Approximate Distributed Top-k Queries[*]

Boaz Patt-Shamir[†] Allon Shafrir
boaz@eng.tau.ac.il shafrir@eng.tau.ac.il
Dept. of Electrical Engineering
Tel Aviv University
Tel Aviv 69978
Israel

March 7, 2007

Abstract

We consider a distributed system where each node keeps a local count for items (similar to elections where nodes are ballot boxes and items are candidates). A top-k query in such a system asks which are the k items whose global count, across all nodes in the system, is the largest. In this paper we present a Monte-Carlo algorithm that outputs, with high probability, a set of k candidates which approximates the top-k items. The algorithm is motivated by sensor networks in that it focuses on reducing the individual communication complexity. In contrast to previous algorithms, the communication complexity depends only on the global scores and not on the partition of scores among nodes. If the number of nodes is large, our algorithm dramatically reduces the communication complexity when compared with deterministic algorithms. We show that the complexity of our algorithm is close to a lower bound on the cell-probe complexity of any non-interactive top-k approximation algorithm. We show that for some natural global distributions (such as the Geometric or Zipf distributions), our algorithm needs only polylogarithmic number of communication bits per node.

keywords: distributed algorithms, aggregate queries, communication complexity, sensor networks, random sampling

1 Introduction

Possibly one of the clearest examples of the difference between "global" and "local" can be seen in elections: each ballot box has a local score for each candidate, but the results we are interested in are the global scores, i.e., how many votes does each candidate have overall. This scenario is representative of the "top-k" query, which is a major target of study in Database Theory. Other examples for the top-k task abound: in peer-to-peer file-sharing networks (such as Gnutella), users may wish to find which are today's most popular downloads; in sensor networks, a sensor may count the number of occurrences of different species of birds, and a user might be interested in the most frequent species observed over the whole instrumented area; in an Internet domain with several gateways, one of the first questions in defending against denial of service (DoS) attacks is which are the most frequent sources of requests over all gateways; and there are many other applications.

[*]An extended abstract of this paper appeared in Proc. 13th Int. Colloquium on Structural Information and Communication Complexity, SIROCCO 2006, Lecture Notes in Computer Science 4056, pp. 319-333.

[†]Corresponding author. Telephone +972 (3) 640-7036, fax +972 (3) 643-6392

In general, a top-k query returns the k items having largest global score in a distributed system, where each item has a set of local scores. The global score of an item is just the sum of its local scores, over all locations. The main difficulty is that the scores may be divided arbitrarily among the different locations. In elections, for example, it may be the case that the most popular global candidate has the lowest positive count in each ballot box. In this case, sampling a single ballot does not help much in determining which are the top-k candidates.

The problem of computing the top-k items in a database is quite well understood when the database consists of a constant number of tables (see [11] for a survey). In this version of the problem, the complexity measure of interest is the number of database *accesses*. However, in a distributed system, a key question in computing top-k queries is how to minimize the *communication complexity* required to provide an answer. This issue is particularly important in sensor networks, where the communication subsystem is by far the largest energy consumer at the nodes. An algorithm which allows us to trade communication for local computation may have a decisive effect on the longevity of node batteries and hence on the usability of the system. This observation has established the measure of *individual* communication complexity as a key performance criterion in sensor networks [15, 17, 24, 7, 20]. The measure of worst-case individual communication cost (used also, for example, in [15]) allows the system designer to estimate the energy required to carry out distributed tasks, in contrast to the more traditional measure of the overall number of messages (see, for example, [1, 16]). Obviously, an upper bound on the worst-case complexity implies an upper bound on the average complexity. In this work, our goal is to minimize the worst-case individual communication complexity of top-k computation. From this point of view, we observe that deterministic algorithms are sensitive to the way scores are partitioned among the different nodes: It may well be the case that the global top-k items are not among the local top-k items in any node, thus forcing essentially the trivial solution, in which all scores are communicated.

1.1 Our results

In this paper we propose a simple and effective way to overcome the problem of adversarial partition of the scores among the nodes. Our algorithm is Monte Carlo (it may err with some arbitrarily small probability), and its results are only approximate: using very little communication, the algorithm can tell, roughly, which items are in the top-k set. We focus on the worst-case individual communication complexity, i.e., our goal is to minimize the maximal number of bits communicated (sent or received) by any single node.

Our basic tool is random sampling. Done in the right way, sampling strips away the difficulties due to geographical distribution of scores which are the main difficulties for deterministic and Las Vegas algorithms. The basic idea is compounded with techniques that allow the algorithm to adapt to the specific input at hand. The performance of the algorithm depends on how popular are the top-k items, and on how "flat" is the global distribution of items. The algorithm learns these parameters adaptively. More concretely, suppose that the least popular item of the top-k appears with frequency p^* in the global input, and suppose that we want a result which is within a factor of $(1+\epsilon)$ from optimal for some $\epsilon > 0$. Then our final algorithm (Algorithm R) guarantees that the communication complexity is always bounded by $O(\frac{1}{p^* \epsilon^2})$ times a polylogarithmic factor. When the distribution of the global scores is "steep," i.e., when the popularity of items decreases relatively quickly, better results can be proved. In particular, if the global scores adhere to the Zipf distribution with parameter $a > 1$ (namely the relative popularity of the i^{th} popular item is proportional to i^{-a}), then the communication complexity drops from $O(\frac{k^a}{\varepsilon^2})$ to $O(\frac{k}{\varepsilon^2})$ (times a polylogarithmic factor). This special case is quite important, as it is widely believed that many natural phenomena (as well as various Internet statistics) are well approximated by the Zipf distribution (see, e.g., [3, 13]).

We note that the communication complexity of our algorithms scales very well compared with previous algorithms [12, 6], since it depends mainly on properties of the underlying global popularity distribution and on the desired approximation accuracy. We stress again, however, that the communication complexity of our algorithms is completely independent of the way scores are divided among nodes. We have run simulations that confirmed the fact that the performance of our algorithm is much better than other known algorithms.

We give some evidence showing that our algorithm is close to optimal. In particular, we analyze a limited class of single-round Monte Carlo algorithms for the top-k problem, and prove a lower bound on the *cell-probe* complexity [23, 14] of algorithms in this model, which is very close to our upper bound in this class.

1.2 Previous work

As mentioned above, the top-k problem received much attention in the context of databases [11], where the goal is to minimize the number of accesses (reads of local counts) by the system. In this model, it appears that one of the strongest results is the Threshold Algorithm (TA) of Fagin, Lotem and Naor [12]. The TA algorithm works as follows. Local counts are scanned top-down in parallel one step at a time, and the sum of the counts of the current local tops is used as a threshold value. The global count of each item encountered is found by looking that item up in all other locations. The top k global results are recorded until there are k items that pass the threshold value. While this does not occur, the next-popular items are looked up and the threshold is updated. Correctness of the TA algorithm follows from the fact that the objects that were not encountered yet cannot have a global count larger than the threshold. Fagin Lotem and Naor argue that the TA algorithm is the best possible among all top-k algorithms that never err (i.e., deterministic or Las Vegas algorithms). Specifically, they prove that TA is 'instance-optimal' for the top-k problem, where instance optimality is a notion reminiscent of competitiveness for on-line algorithms. However, we note that TA may do as many as n accesses times the optimum, where n is the number of nodes in the system (which is unavoidable for any algorithm which doesn't err). While this is fine if n is a small constant (as in the database scenario), the situation is different in the distributed case: n may be very large, and the dominant cost measure is communication rather than accesses.

This viewpoint is developed by Cao and Wang in [6], who propose the TPUT algorithm to reduce latency and save communication in some typical cases. TPUT works in three communication rounds between a designated coordinator node (say, the initiator of the query) and all other nodes. First, all nodes report their local top-k items to the coordinator, which computes the partial counts according to these reports. Let T be the value of the kth largest count in these partial sums (T is clearly a lower bound on the count of the true kth most popular item). In the second round, the coordinator sends T/n to all nodes, where n is the number of nodes. As a response, each node sends to the coordinator all items whose local count is at least T/n. The coordinator now computes the partial sums according to all reports received so far. Let T' be the value of the kth largest count in these partial sums. Then, all items that may have global count larger than T' are counted in all nodes, and the top k of them are reported by the coordinator as the final output. Cao and Want prove that TPUT is instance-optimal if the *local* inputs are generated by Zipf-like distributions. As expected from an algorithm which never errs, the performance of TPUT also depends crucially on the way the scores are partitioned among nodes. Other related work include variations of TA and TPUT optimized for certain network models [18, 25, 5].

From the sensor networks perspective, top-k queries are viewed as a special case of aggregate queries (see, e.g., [17, 24]). Typically, it is assumed that data is routed on a spanning tree, and each node does some aggregation en-route. Simple aggregates (such as counting the number of items, summing numbers etc.) can be done with $O(\log n)$ bits per node. Considine et al. [7] relax the spanning tree assumption to allow messages to be duplicated for these aggregates. In [19], Gibbons et al. present a methodology for robust approximation of aggregates in sensor networks. They also present a sketch of a top-k-approximation algorithm that appears promising, but the algorithm is not fully specified, and no formal statement or analysis is given. Computing the *median* in sensor networks does not fall within the class of simple queries described above. Deterministic and randomized algorithms for median computation are studied in [20, 15]; the randomized algorithm of [20] computes an approximate median with high probability using $O(\log\log n)$ communication per node.

Many recent papers focus on techniques for dynamically monitoring aggregates over time in sensor networks, where the main question is how to efficiently update the results under some assumptions on the way the input changes. Monitoring the k-largest values is studied in [21, 4], monitoring the *median* and *heavy-hitters* is studied in [8] and monitoring the top-k items is studied in [2].

1.3 Paper Organization

The remainder of the paper is organized as follows. Section 2 describes our model, problem definition and a few known results about efficient counting. Section 3 presents our algorithms with formal analysis results. Simulation results are presented in Sect. 4. In Sect. 5, we present our lower bound, and we conclude in Sect. 6. In order to keep the flow of the paper simple, some proofs are deferred to the appendices.

2 Model and preliminaries

2.1 System model

The system is modeled as a communication graph $G(V,E)$ with $n \triangleq |V|$, where each node models a classical RAM machine with access to its local input and to an infinite tape of random bits. A distinguished node $v^* \in V$ is the *root node* and is assumed to have a special write-once output register.

The system executes distributed algorithms according to the standard asynchronous message passing model (see, e.g., [1, 16]). Very briefly, in this model an event (such as arrival of a message to node u) triggers a state-transition (e.g., u computes a response message and inserts it to the link buffer). An execution in our model is considered terminated when the root-node has written the result to the output register.

Let M denote some finite string of bits. We assume that the system contains a message passing infrastructure supporting the following facilities:

- Each node $u \in V$ may send a message M to any other node $v \in V$. This causes each node along some simple path from u to v to send and receive $\Theta\left(|M|\right)$ bits.
- Each node $u \in V$ can broadcast a message M to all other nodes. This causes every node in V to send and receive $\Theta\left(|M|\right)$ bits.

The particular way in which these actions are implemented is immaterial for our purposes. We note, however, that these assumptions can be justified by the existence of a spanning-tree of constant degree for message passing (see, e.g., [15, 17, 24, 7, 20]). We note that our complexity measure ignore the overhead incurred by the routing subsystem, such as message headers. This is because the routing mechanism can be tailored in many different ways which are beyond the scope of this paper.[1]

[1]For example, there may be an identifier for each task (such as the top-k task), and each message can be tagged by the identifier of the task, along with one additional bit saying whether the message is from the root (to be broadcast to everyone) or to the root (in which case its source is immaterial).

2.2 Input model and problem statement

Let $\mathcal{I}$ denote the set of possible *items*. We assume $\mathcal{I}$ is finite. An instance of the problem, denoted by X, is a vector of multisets of $\mathcal{I}$: one multiset, denoted X_v, for each node $v \in V$. We sometimes slightly abuse notation and use X to also denote the multiset $\bigcup_{v \in V} X_v$.

It is convenient to imagine each multiset as a set of *cells*, where each cell contains a single item, so that an item with multiplicity w has w replicas, one in a cell. The *weight* of item i in node v, denoted $w_v(i)$, is the multiplicity of i in X_v. The *weight of a node* v is the total number of cells in v, formally $W_v \triangleq |X_v|$. The *weight of an item* $i \in \mathcal{I}$, is the sum of its multiplicities over all nodes, i.e., $w(i) \triangleq \sum_{v \in V} w_v(i)$. We define the total input size as the total number of input-cells, i.e., $W(X) \triangleq \sum_{v \in V} |X_v|$. The *empirical probability,* or *frequency,* of an item $i \in \mathcal{I}$ in X, is defined by $p_X(i) \triangleq \frac{w(i)}{W}$. When the context is clear we omit the subscript.

Using this notation, we define the top-k set of X as follows.

Definition 2.1 *Let k be a natural number, and suppose that $|\mathcal{I}| \geq k$. The top-k set of X, denoted* $\operatorname{top}(k, X)$*, is a subset of $\mathcal{I}$ of size k containing the items with the maximal weights. Formally,* $\operatorname{top}(k, X)$ *satisfies the following conditions.*

(1) $\operatorname{top}(k, X) \subseteq \mathcal{I}$,

(2) $|\operatorname{top}(k, X)| = k$, *and*

(3) $\forall i \in \operatorname{top}(k, X), \forall j \in \mathcal{I} \setminus \operatorname{top}(k, X): \; p_X(i) \geq p_X(j)$.

Following [12], we extend Definition 2.1 to the concept of approximate top-k sets.

Definition 2.2 *Let $\varepsilon \geq 0$. A set $\operatorname{top}_\varepsilon$ of k items is an ε-approximation of $\operatorname{top}(k, X)$ if and only if for all items $i \in \operatorname{top}_\varepsilon$ and $j \notin \operatorname{top}_\varepsilon$, we have $p_X(j) \leq (1 + \varepsilon) p_X(i)$.*

We will mainly be interested in small values of ε so without loss of generality, we assume henceforth that $\varepsilon \leq 1$.

It turns out that the following quantity has a central role in the complexity of computing top-k (and approximate top-k) queries. For a given input, the *critical frequency* of the instance, denoted $p^*(X, k)$, is the empirical probability of the least popular item in $\operatorname{top}(k, X)$, i.e., given input X and a natural number k, we define

$$p^*(X, k) \triangleq \min \{ p_X(i) \mid i \in \operatorname{top}(k, X) \} \ .$$

Throughout the paper, we assume that instances have n nodes, total weight W, and critical frequency p^*. We denote the set of all such instances by $\mathcal{X}(W, n, p^*)$.

2.3 Complexity measures

We evaluate the performance of certain algorithms using a worst-case measure per node. Specifically, the communication complexity of an algorithm is the maximum, over all inputs and over all nodes, of the total number of bits transmitted and received by a node throughout the execution of the algorithm. Formally, $c_A(X, v)$ denotes the total number of bits transmitted and received by node v, throughout the execution of algorithm A, for the input X; $c_A(X)$ denotes the maximal node-communication of algorithm A on input X, i.e., $c_A(X) \triangleq \max \{ c_A(X, v) \mid v \in V \}$. Finally, given a collection $\mathcal{X}$ of possible inputs, $C_A(\mathcal{X})$ denotes the worst-case communication complexity of algorithm A over all inputs in $\mathcal{X}$, i.e., $C_A(\mathcal{X}) \triangleq \max \{ c_A(X) \mid X \in \mathcal{X} \}$.

Note that our communication complexity measure is individual in the sense that we measure the maximal number of bits communicated by any single node. The motivation for such a measure is that in sensor networks, each node has an individual energy source, and the longevity of the system often depends on the longevity of the weakest sensors (see, e.g., [15]). Furthermore, assuming a spanning tree of bounded degree, we can disregard many aspects of wireless communication and focus on the net communication used by the algorithm.

In some cases, it is more convenient to analyze the *cell probe* complexity of algorithms [23, 14] rather than directly analyze their communication complexity. We define the cell probe complexity of an algorithm as the maximum number of input cells accessed by a single node. Specifically, $\mathrm{cp}_A(X, v)$ denotes the number of input cells accessed by node v, throughout the execution of A for input X; $\mathrm{cp}_A(X)$ denotes the maximal-node cell-probes of algorithm A for input X, i.e., $\mathrm{cp}_A(X) \triangleq \max\{\mathrm{cp}_A(X, v) \mid v \in V\}$. Finally, $\mathrm{CP}_A(\mathcal{X})$ denotes the cell probe complexity of an algorithm A over a set of possible inputs $\mathcal{X}$, namely $\mathrm{CP}_A(\mathcal{X}) \triangleq \max\{\mathrm{cp}_A(X) \mid X \in \mathcal{X}\}$.

2.4 Loglog counting

Let us present a known result which we use. First we define the following concept.

Definition 2.3 *Let Z be a positive number we wish to estimate, let $\varepsilon \geq 0$ and $\sigma \geq 0$ be real numbers. A random variable $\hat{Z}$ is a (ε, σ^2)-estimate of Z if $\frac{1}{Z}|E[\hat{Z}] - Z| \leq \varepsilon$, and $\frac{1}{Z^2}$Var$[\hat{Z}] \leq \sigma^2$.*

Durand and Flajolet [10] prove a result which, specialized to our needs, can be stated as follows.

Fact 2.1 ([10]) *There exists an algorithm A_{loglog} which outputs an (ε, σ^2)-estimate of W with $\varepsilon = 10^{-6}$ and $\sigma = 1$, using $O(\log\log W)$ bits of communication.*

To get bounds that hold with high probability, we iterate Algorithm A_{loglog} and use Bernstein's Inequality (see, e.g., [9]).

```
Algorithm BoundCount ( Input: $\varepsilon, \delta, i$ )
  (1) $M \leftarrow (6/\varepsilon^2)\ln 1/\delta$.
  (2) Broadcast a filtering message indicating that only cells holding item $i$ should be considered in Step 3.
  (3) for $\ell = 1$ to $M$, run $A_{\text{loglog}}$ obtaining an independent estimate $\hat{w}_\ell$ of $w(i)$.
  (4) Output $\hat{w} \triangleq \frac{1}{M}\sum_{\ell=1}^{M} \hat{w}_\ell$.
```

Corollary 2.2 (high-probability estimates) *For $10^{-5} \leq \varepsilon \leq 1$ and $\delta > 0$, the output $\hat{w}$ of Algorithm BoundCount satisfies $\Pr\left\{\frac{1}{w(i)}|\hat{w} - w(i)| < \varepsilon\right\} \geq 1 - \delta$. The individual communication complexity of the algorithm is of order $O\left(\log|\mathcal{I}| + \frac{1}{\varepsilon^2}\log\frac{1}{\delta}\log\log w(i)\right)$. Also, if the algorithm ran M iterations in Step 3, then for any $\zeta > 10^{-5}$, $\Pr\left\{\frac{1}{w(i)}|\hat{w} - w(i)| < \zeta\right\} \geq 1 - \exp\left(-\Omega\left(M\zeta^2\right)\right)$.*

The proof is given in Appendix A.

3 Algorithms

In this section we present our main result, namely a randomized algorithm for computing top-k. The first difficulty introduced by the partition of the scores into multiple nodes is the possible discrepancy between the local scores in different nodes. For example, the most popular item overall may actually be the least popular item in each node. More generally, no local view of scores indicates which are the global top k items. Thus, when the items are adversarially distributed among the nodes, deterministic algorithms will test many irrelevant items, wasting a lot of communication. In our algorithm, the basic idea is to view each cell (representing a unit of weight, or score) as a "vote," and to *sample each vote independently*. Thus, the expected number of sampled votes for candidate i is proportional to the total number of votes candidate i has in the input *regardless of their partition into nodes.* The sampling results provide a good indication which items are globally popular, so that counting can be applied only to these items, thus saving a lot of communication.

Next, we need to determine the sample size. Let p^* denote the frequency of the least popular of the top-k items. Clearly, if we sample once, the sample size should be at least $\Omega(1/p^*)$ (or else the sample will fail to find all top k items). A more refined analysis shows what should be the sample size as a function of p^*, the approximation parameter ε, and the confidence parameter δ. To deal with unknown p^*, we augment the basic sampling algorithm with a technique to find the right sample size. Intuitively, we have a simple test which can prove whether the sample size is sufficiently large; if it isn't, we double the sample size.

Finally, we address the issue of very small p^* values: while $\Omega(1/p^*)$ sample size cannot be avoided for worst-case inputs, a much better bound can be obtained if the popularity of items decreases relatively rapidly. Consider, for example, the case where the global scores are close to the geometric distribution, e.g., when the frequency of the ℓ^{th} popular element is about $2^{-\ell}$. Then we have $p^* = 2^{-k}$, and therefore the sample size should be $\Omega(2^k)$. This cost can be reduced exponentially by utilizing the following simple idea: Whenever a sample is taken, the top item in the sample is by far the most popular (it is expected to have half of the weight in the sample). Therefore it is safe to add the top item to the output list, remove it from further consideration (as if all these items are not present in the system), and take another sample *of the same size*. In the geometric case, this approach has communication cost linear in k. This intuition leads us to our final algorithm, called Algorithm R. In essence, the idea is to iteratively discover the very top items, "shave them off," and to continue recursively with the remainder of the population. The algorithm combines this idea with an additional way to verify that the top-k items have been discovered. We show that Algorithm R is far better than Algorithm S for some common input distributions, such as Zipf distribution.

We start, in Sect. 3.1, by describing the basic algorithm we later use as a building block. Section 3.2 presents Algorithm S which uses adaptive sample size. Section 3.3 presents Algorithm R, which is our main result.

3.1 Algorithm B: Basic Sampling

Consider basic sampling: if the top-k items occupy a constant fraction of the total weight, then a log-size sample is sufficient to detect them for any input size (the logarithm is of the inverse of the error probability).

It is convenient to first define a simple sampling routine and analyze its properties.

```
Algorithm A ( Input: P_SAMPLE )
  (1) The root sends P_SAMPLE to all other nodes.
  (2) Each node sends each cell to the root with probability P_SAMPLE.
```

Lemma 3.1 *There exists a function* $S^*(p^*, \varepsilon, \delta) = \Theta\left(\frac{1}{p^*\,\varepsilon^2} \cdot \ln \frac{1}{p^*\,\delta}\right)$ *, such that for any input* $X \in \mathcal{X}(W, n, p^*)$*, the top-*$k$ *elements of a random sample of size at least* $S^*(p^*, \varepsilon, \delta)$ *is an* ε*-approximation of* $\operatorname{top}(k, X)$ *with probability at least* $1 - \delta$*.*

We give here only an outline of the proof. The full proof is rather technical and is given in Appendix B.

Proof outline: To bound the error-probability of a given sample size, we identify pairs of items i, j such that i must be more popular than j in the sample to get a top-k ε-approximation. By setting a threshold between i and j for every pair and requiring that no item reaches any of its thresholds we get a single condition per item. The choice of thresholds is such that we get tight bounds on error probabilities. Finally, we group items together by their global frequency and use the union bound within each group to bound the probability of errors. Summing the error probabilities of all groups, we get a bound on the overall error probability. Requiring that this probability be below δ, we get our bound for the required sample size. Since our algorithm calculates $S^*(p^*, \varepsilon, \delta)$, we give here its exact definition; we set $S^*(p^*, \varepsilon, \delta) \triangleq \frac{g(\varepsilon)}{p^*} \cdot \left(\ln \frac{1+\varepsilon}{p^*\delta} + 4\right)$, where $g(\varepsilon) \triangleq \frac{(1+\varepsilon)\ \ln(1+\varepsilon)}{\varepsilon\ \ln\left(\frac{\varepsilon}{\ln(1+\varepsilon)}\right)+\ln(1+\varepsilon)-\varepsilon}$.

Algorithm B, presented below, follows directly from the Lemma 3.1. The algorithm first determines the sampling probability $\mathtt{P_{SAMPLE}}$ using the function S^* from Lemma 3.1. Each cell (i.e., unit of weight) is then sent to the root with probability $\mathtt{P_{SAMPLE}}$, and the root outputs the top k items in the sample as an approximation of the top k items in the complete input.

```
Algorithm $B$ ( Input: $W, p^*, k, \varepsilon, \delta$ )
  (1) The root computes $\mathtt{P_{SAMPLE}} \leftarrow 2 \cdot S^*(p^*, \varepsilon, \frac{\delta}{2})/W$.
  (2) Execute Algorithm $A$ with parameter $\mathtt{P_{SAMPLE}}$ to get a sample $\mathcal{S}$.
  (3) Output $\mathrm{top}(k, \mathcal{S})$.
```

When running Algorithm B as described above, each sampled vote is sent to the root separately incurring communication $\log|\mathcal{I}|$. An obvious optimization is to aggregate votes for the same candidate along the way, for example by sending the *count* of votes for each candidate. While such optimization is very worthwhile to implement, it would not help much when the partition of votes to nodes is adversarial. We therefore ignore such optimizations in our upper bounds. Furthermore, we bound the worst-case communication complexity by the size of the sample: this is certainly an upper bound on the number of bits communicated by any node in the system (the worst case occurs at the root node).

Theorem 3.2 *For any instance $X \in \mathcal{X}(W, n, p^*)$ and known p^*, ε, δ and W, Algorithm B outputs a top-k ε-approximation with probability at least $1-\delta$ and communication complexity of order $O\left(\frac{1}{p^*\,\varepsilon^2} \cdot \ln\frac{1}{p^*\delta} \cdot \log|\mathcal{I}|\right)$.*

The proof is rather standard and is given in Appendix B. Note the $1/p^*$ factor: It is unavoidable because if the sample is to contain the top k elements, it should contain the least popular of them, and hence the sample size must be $\Omega(1/p^*)$. A stronger lower-bound is proved in Section 5.

3.2 Algorithm S: Adaptive Sample Size

Algorithm B requires knowing the values of W and p^*. While estimating W is straightforward and cheap (by deterministic or randomized counting, at the cost of $O(\log W)$ or $O(\log\log W)$ communication, respectively), obtaining a lower bound on p^* seems less trivial. We solve this problem as follows.

First, we note that by counting the weight of *any* k items, we obtain a lower bound on p^*: the least popular among any k items is certainly no more popular than the least popular among the top-k items. Second, we note that exact counting is not necessary: we can use high-probability-estimates as described in Corollary 2.2 to get a lower bound on p^* that holds with high probability. However, if we simply use an arbitrary set of k items to bound p^*, that value can be smaller than the true value of p^* by an arbitrary factor, resulting in communication cost that is higher than the bound in Theorem 3.2 by an arbitrary factor.

Our solution, in Algorithm S (Figure 1), combines the ideas described above with an iterative approach that avoids unbounded 'overshoots.' Intuitively, Algorithm S uses a variable $\hat{p}$ as an estimate of p^*. The algorithm starts by assigning $\hat{p} \leftarrow 1$, and repeatedly performs halving of $\hat{p}$ while improving the lower bound on p^*. The process continues until $\hat{p}$ is smaller than the lower bound. The algorithm is formally presented in Figure 1.

```
Algorithm $S$ ( Input: $k, \varepsilon, \delta$ )
  (1) $\hat{W} \leftarrow$ BoundCount($\varepsilon = 1, \delta/5$, 'ALL')
  (2) $\hat{p} \leftarrow 1$ ($\hat{p}$ is the current estimate of $p^*$)
  (3) Repeat:
        (3a) $\hat{p} \leftarrow \frac{\hat{p}}{2}$
        (3b) Execute Algorithm $B$ with parameters $(\hat{W}/2, \hat{p}, \varepsilon, \delta/5)$, to get a candidate top-$k$-set $T$.
        (3c) For each item $i \in T$,
                 $\hat{w}(i) \leftarrow$ BoundCount($\varepsilon = 1, \delta/5, i$)
      Until $\hat{p} < \frac{1}{8} \min \left\{ \hat{w}(i)/\hat{W} \mid i \in T \right\}$.
  (4) Output the set $T$ computed at Step 3b of the last iteration.
```

Figure 1: Algorithm S

Theorem 3.3 *For any input $X \in \mathcal{X}(W, n, p^*)$, with probability at least $1 - \delta$, Algorithm S outputs a top-k ε-approximation with communication complexity*

$$C_S = O\left(\frac{1}{p^* \, \varepsilon^2} \cdot \log \frac{1}{p^* \, \delta} \cdot \log |\mathcal{I}| \; + \; k \log \frac{1}{p^*} \log \frac{1}{\delta} \log \log W \right) .$$

The proof is deferred to Appendix C. The general idea in the proof is that the high-probability estimates ensure that we run approximately $\log \frac{1}{p^*}$ iterations. As a result, the last iteration is similar to an execution of Algorithm B with the correct parameters and its communication complexity is as specified in Theorem 3.2.

Note that when $|\mathcal{I}| \geq \log W$ and $k < O\left(1/(-p^* \log p^*)\right)$, the communication complexity of Algorithm S is within a constant factor of the complexity of Algorithm B.

3.3 Algorithm R: Sample & Remove

As already mentioned above, it appears that the $1/p^*$ factor is unavoidable when we want all the top-k items to be included in the sample (because p^* is the empirical probability of the k^{th} popular item). However, when the popularity of the popular items is far from uniform, one can do much better, as mentioned for the geometric distribution example in the beginning of this section: the $1/p^*$ factor in the communication complexity can be replaced by a factor of k. Algorithm R achieves this improvement for both geometric and Zipf distributions. In a nutshell, the idea is still to cut the estimate of p^* by half in each iteration; however, while Algorithm S achieves this by doubling the sample size, Algorithm R tries also to reduce the size of the relevant population.

In more detail, the algorithm works as follows (see Figure 2). In each iteration, the algorithm samples with probability $\mathrm{P}_{\texttt{SAMPLE}}$, obtained from the current estimate $\hat{p}$ of p^*. Then, in Step 3f, the algorithm counts (approximately) some of the top items in the sample: the number of items is such that the cost of counting equals the cost of sampling. The counted items (recorded in Q) are not considered part of the input anymore (their identities are broadcast to all nodes in Step 3h). In Step 3c, $\hat{p}$ is adjusted so that its value *in the full input* is halved (but since the input is smaller now, $\hat{p}$ is multiplied by a factor larger than $1/2$). The loop is executed until one of the stopping rules specified in Predicate `safe` is met. These rules use a lower bound on p^* computed by function p_{lo}: it is a high-probability lower bound on the k^{th} smallest value among all values counted so far. Using p_{lo}, the stopping rules are defined as follows. First, we can stop if the value of $\hat{p}$ w.r.t. the full input is smaller than the lower bound on p^*. This test is done in Step 1 of Predicate `safe`. Second, we bound the probability that an "important" item did not arrive at the top of the current sample and therefore was not counted.

```
**Algorithm** $R$ ( Input: $k, \varepsilon, \delta$ )
  (1) $Q \leftarrow \emptyset$ ($Q$ holds all items whose weights were already estimates)
  (2) $\hat{p} \leftarrow 1, \ell \leftarrow 0$. ($\hat{p}$ is the current estimate of $p^*$, $\ell$ is the iteration index)
  (3) Repeat
      (3a) $\hat{W} \leftarrow$ BoundCount($\varepsilon = 1, \delta/7$, 'ALL').
      (3b) $\ell \leftarrow \ell + 1$ ; $\hat{W}[\ell] \leftarrow \hat{W}$.
      (3c) $\hat{p} \leftarrow \hat{p} \cdot \frac{1}{2} \cdot \hat{W}[\ell - 1] / \hat{W[\ell]}$.
      (3d) $\mathtt{P_{SAMPLE}} \leftarrow 2S^*(\hat{p}, \varepsilon/4, \delta/7) \,/\, \hat{W}$.
      (3e) Execute Algorithm $A$ with parameter $\mathtt{P_{SAMPLE}}$ to get sample $\mathcal{S}$.
      (3f) $\eta \leftarrow (\delta/7) \cdot (1/ - \log p_{lo}(\mathrm{top}(k, Q), \hat{W}))$;
           $T^{\mathcal{S}} \leftarrow \mathrm{top}(|\mathcal{S}| \log |\mathcal{I}| \,/ (\log |\mathcal{I}| + \log\log \hat{W}),\ \mathcal{S})$.
      (3g) For each $i \in T^{\mathcal{S}}$, $\hat{w}(i) \leftarrow$ BoundCount($\varepsilon = 1, \eta, i$).
      (3h) $Q \leftarrow Q \cup T^{\mathcal{S}}$; remove elements of $T^{\mathcal{S}}$ from the input.
      Until safe($\hat{W}, Q, \mathcal{S}, \hat{p}, \varepsilon, \delta$).
  (4) If $|T^{\mathcal{S}}| < k$, then for each $i \in \mathrm{top}(k, \mathcal{S})$, do $\hat{w}(i) \leftarrow$ BoundCount($\varepsilon = 1, \eta, i$) and add $i$ to $Q$.
  (5) $Q \leftarrow \left\{ i \in Q \mid \hat{w}(i)/\hat{W} > p_{lo}(\mathrm{top}(k, Q), \hat{W}) \right\}$
  (6) For each $i \in Q$, $\hat{w}(i) \leftarrow$ BoundCount($\varepsilon/4, \frac{\delta}{5|Q|}, i$).
  (7) Output the top $k$ items in $Q$ according to the new estimates.

**Function** $p_{lo}$ ( Input: $Q, W$ )
  (1) Return $\frac{1}{4W} \min \{\hat{w}(i) \mid i \in Q\}$.

**Predicate safe** ( Input: $W, Q, \mathcal{S}, \hat{p}, \varepsilon, \delta$ )
  (1) If $\hat{p} \cdot (W[\ell]/W[1]) < p_{lo}(\mathrm{top}(k, Q), W)$ return TRUE.
  (2) $r_{hi} \leftarrow p_{lo}(\mathcal{S} \cap Q, W)$
  (3) $q \leftarrow (1 + \varepsilon/2) \cdot p_{lo}(\mathrm{top}(k, Q), W)$; $\alpha \leftarrow q/r_{hi}$.
  (4) If $\alpha > 1$ AND $\exp\left(-|\mathcal{S}| q \left(\frac{\alpha - 1}{\alpha}\right)^2\right) < \frac{q\delta}{5}$ then return TRUE else return FALSE.
```

Figure 2: Algorithm R

More specifically, we bound the probability that an item with frequency q, where q is sufficiently larger than p^*, was not counted, while an item whose frequency is r_{hi} was counted. If the probability of this event is low, we know that with-high-probability, the top items counted by the algorithm contain an ε-approximation to the top-k set (Step 4). This test is useless in some cases (say, the uniform distribution), but it is effective in some distributions (such as Zipf).

Theorem 3.4 shows some general bounds on the complexity of Algorithm R. As expected, it shows that for some inputs, Algorithm R has very little advantage over Algorithm S. Theorem 3.9, however, presents an alternative analysis which, when applied to some natural input distributions, attains much better bounds than those of Theorem 3.4.

Theorem 3.4 *For any input $X \in \mathcal{X}(W, n, p^*)$, with probability at least $1 - \delta$, Algorithm R outputs a top-k ε-approximation with communication complexity $C_R = O\left(\frac{1}{p^* \varepsilon^2} \log \frac{1}{\delta p^*} (\log |\mathcal{I}| + \log\log W)\right)$.*

Again, when $|\mathcal{I}| < \log W$, the communication complexity of Algorithm R is within a constant factor of the complexity of Algorithm B.

Proof of Theorem 3.4: The proof outline is as follows. The proof separates the algorithm to the iterations phase and the final phase. First we show that if by the end of the iterations, the set Q contains an approximation of $\mathrm{top}(k, X)$ then the output (chosen by high-probability estimates) is correct with high probability. Next we show that Q satisfies this condition with high probability; this is shown separately for two scenarios according to the stopping condition by which the iterations terminated. For the purpose of bounding communication we show that in the worst case, the iterations perform similarly to Algorithm S. Throughout the proof we use $\delta' \triangleq \delta/7$ to express error probabilities and $L' \triangleq 5 + \log \frac{1}{p^*}$ to express a bound on the number of iterations.

Lemma 3.5 *If at the end of Step 4 of the algorithm,* $\mathrm{top}(k, Q)$ *is a top-*k $\frac{\varepsilon}{4}$*-approximation then with probability at least* $1 - \delta'$*, the output of Algorithm* R *is a top-*k ε*-approximation and the communication cost of Step 6 is of order* $O\left(\frac{1}{p^*} \log \frac{1}{p^*} \log |\mathcal{I}| + \frac{1}{p^* \varepsilon^2} \log \frac{1}{p^* \delta} \log \log W\right)$

Proof: Let j be the least popular item in $\mathrm{top}(k, Q)$. If all obtained estimates satisfy the requested accuracy then in the worst case scenario:

- There exists an item $i' \notin Q$ whose weight is $w(i') = (1 + \varepsilon/4)w(j)$.
- There exists an item $j' \in Q$ such that $w(j) = w(j') \cdot (1 + \varepsilon/2)$.
- The high-probability estimate of $w(j)$ was low by relative error of $\varepsilon/4$.
- The high-probability estimate of $w(j')$ was high by relative error of $\varepsilon/4$.

In this case R may output j' and not i', and the approximation error of the output would still be below ε. A worse approximation factor is possible only if one of the high-probability estimates obtained in Step 6 is wrong. The error probability passed to BoundCount in this line gives the result of the lemma. Since we removed from Q all items having weights below $w^*/4$, the size of Q is bound by $O\left(\frac{1}{p^*}\right)$ and by Corollary 2.2 we get the communication cost bound. ▌

It remains to show that with high probability, the conditions of Lemma 3.5 hold. We show this separately for each of the two stopping conditions. We start with a lemma about the number of iterations.

Lemma 3.6 *With probability at least* $1 - 3\delta'$*, Algorithm* R *runs at most* L' *iterations before it stops.*

Proof: Recall that $L' \triangleq \log \frac{1}{p^*} + 5$ and let E_3 denote the event where Algorithm R reached iteration $L' + 1$. We assume that Algorithm R did reach iteration L' and consider the state at the end of this iteration. Note also that the stopping condition at Step 1 of `safe` is the same as the stopping condition of Algorithm S. Denote by p' the left hand side of this condition, i.e. $p' \triangleq \hat{p} \cdot (\hat{W}[\ell]/\hat{W}[1])$. Note that by Step 3c of the algorithm, p' is halved on each iteration so on iteration L', $p' \leq \frac{1}{32} p^*$. The right hand side of the condition is actually the frequency-bound of the item j having the k-highest estimate from items discovered so far. If Algorithm R continued to the next iteration, then $\{p' \geq p(j)_{lo}\}$, and therefore either of the two following events occurred on iteration L':

- Event $E_3' : p(j) \leq \frac{1}{2}p^*$
- Event $E_3'' : p(j) \geq \frac{1}{2}p^* \text{ AND} p' \geq p(j)_{lo}$.

First consider the probability of E_3' on iteration L'. Clearly there is some item $j' \in \mathrm{top}(k, X)$ which is either missing from the set of counted items or got an estimate lower than that of j. Let e' denote the first case and let e'' denote the second. Regarding e', Note that the last sample size was calculated by $\hat{p}$ which is at least 32 times larger than the frequency of j' in the modified input. By Lemma 3.1, Pr $\{e'\} < \delta'$. Since e'' means an error in one of the high-probability-estimates, its probability is also smaller than δ' and we have Pr $\{E_3'\} < 2\delta'$. E_3'' is the event where $p(j)$ is high enough but $p(j)_{lo}$ is too low. Since on iteration L', we have $p' < \frac{p^*}{32}$, E_3'' requires that $\left\{\frac{1}{16}p(j) > p(j)_{lo}\right\}$. Recall that $p(j)_{lo} \triangleq \frac{\hat{w}(j)}{2W_{hi}}$ where $\hat{w}(j)$ was obtained by Algorithm BoundCount using

$\varepsilon = 1$. By Corollary 2.2 the probability of the last event is less than δ' and therefore $\Pr\{E_3''\} < \delta'$. Since E_3 requires that either E_3' or E_3'' occur on iteration L' we have that $\Pr\{E_3\} < 3\delta'$ ▮

Lemma 3.7 *If Algorithm R terminated by the stopping condition in Step 1 of* `safe`*, then by the end of Step 4 of Algorithm R, $\mathrm{top}(k,Q)$ was a top-k $\frac{\varepsilon}{4}$-approximation with probability at least $1-5\delta'$.*

Proof: Let j' denote the least popular item in $\mathrm{top}(k,Q)$ after the last iteration of the algorithm. If $\mathrm{top}(k,Q)$ does not satisfy the result then there exists an item $i' \notin Q$ such that $w(i') > (1+\varepsilon/4)\cdot w(j')$. Now, let j be the least popular item in $\mathrm{top}(k,\mathcal{S})$ on the last iteration; clearly $w(j) < w(j')$ and it follows that $\mathrm{top}(k,\mathcal{S})$ is not a $\frac{\varepsilon}{4}$-approximation of the top-k-set in the modified input. Let X' denote the modified input at the beginning of the last iteration. Using Lemma 3.1 we get that if $\mathrm{top}(k,Q)$ doesn't satisfy the result then either we used $\hat{p} > p^*(X',k)$ or we had a sampling error that occurs with probability δ'. The event that we stopped even though $\hat{p} > p^*$ requires that the k-highest estimate in Q is off by a factor of over 2. If this is a different estimate on each iteration that it may occur with probability at most η on each iteration. By Lemma 3.6, with probability $1-\delta'$, there are at most L' iterations. For $\eta \le \delta'/L'$ we get an overall error probability of at most $5\delta'$. ▮

Lemma 3.8 *If Algorithm R terminated by the stopping condition in Step 4 of* `safe`*, then by the end of Step 4 of Algorithm R, $\mathrm{top}(k,Q)$ was a top-k $\frac{\varepsilon}{4}$-approximation with probability at least $1-\delta'$.*

Proof: Let j' denote the least popular item in $\mathrm{top}(k,Q)$ after the last iteration of the algorithm. If $\mathrm{top}(k,Q)$ does not satisfy the result (i.e. it is *not* a top-k $\frac{\varepsilon}{4}$-approximation) then there exists an item $i' \notin Q$ such that $w(i') > (1+\varepsilon/4)\cdot w(j')$. Letting j be the least popular item in $T^{\mathcal{S}}$ we note that if $i' \notin Q$ then j is more popular than i' in the last sample. For a specific such i', this occurs with probability at most $\exp\left(-\frac{1}{8}S\cdot p(i)\cdot\Theta\left(\left(\frac{\alpha-1}{\alpha}\right)^2\right)\right)$ by Lemma B.2. Since the condition demands that this probability be $p(i')\delta'$ and there are no more than $1/p(i')$ such items, we get the result. ▮

Combining the last 3 lemmas we get that with probability at least $1-5\delta'$, the conditions of Lemma 3.5 hold and the algorithms runs at most $\Theta\left(\log\frac{1}{p^*}\right)$ iterations. It follows that Algorithm R is correct with probability at least $1-6\delta'$.

As for communication complexity, first note that Step 3f of the algorithm carefully calculates the number of items to be counted so that the communication of counting doesn't exceed that of sampling. Secondly, note that for the $\ell' th$ iteration the largest communication cost is performed when $\hat{p}$ is minimal. The series of iterations-communication-costs is thus bound by the series where $\hat{p}$ is halved on each iteration. In this case the sum is bound by twice the communication cost of the last iteration. assuming the algorithm ran $O\left(\log\frac{1}{p^*}\right)$ iterations, the last iteration has $\hat{p} = \Omega(p^*)$ and the communication cost of the last iteration is bound by $O\left(S^*(p^*,\varepsilon/4,\delta'\, log\frac{1}{p^*})\log|\mathcal{I}|\right)$ with probability at least $1-\delta'$. By adding the communication cost of Step 6 as specified in Lemma 3.5 we get the result of the theorem. ▮

Next we refine the analysis to depend more closely on the global distribution (rather than only on p^*). We use the following definition.

Definition 3.1 *Fix the input distribution. For any integer $\ell \ge 0$, π_ℓ is the probability that a random item has global frequency at most $2^{-\ell}$. Formally:* $\pi_\ell(X) \triangleq \sum_{i:\,p_X(i)\le 2^{-\ell}} p_X(i)$.

In our main result below, we get rid of the direct dependence on $1/p^*$ as follows.

Theorem 3.9 (Main result) *Let $X \in \mathcal{X}(W,n,p^*)$. Then with probability at least $1-\delta$, Algorithm R outputs a* *top-k ε-approximation of X while using $O\left(\frac{1}{\varepsilon^2}\log\frac{1}{p^*\,\delta}\,\left(\log|\mathcal{I}|+\log\log W\right)\cdot\sum_{\ell=1}^{\log\frac{1}{p^*}} 2^\ell \pi_\ell\right)$ communication bits per node where π_ℓ is as defined above.*

Proof: The proof is a generalization of the "shaving" concept mentioned in the beginning of this chapter. It starts by showing a direct relation between the iterations of the algorithm and the frequency of the items not yet discovered. Secondly, it uses this relation and π_ℓ to get tight bounds on the communication of an iteration. From a similar bound on the number of items "discovered" on each iteration, we bound the size of the set Q and hence the communication of the final counting phase. The last part of the proof combines these bounds to get the result.

The proof uses the following notation. Let $\hat{p}[\ell]$ and $R[\ell]$ denote the values of $\hat{p}$ and of $\frac{W[\ell]}{W[1]}$ used in the ℓ^{th} iteration, and let $c[\ell]$ denote the individual-communication used throughout the ℓ^{th} iteration. As before we use $\delta' \triangleq \delta/7$.

Lemma 3.10 *The individual-communication done in the ℓ^{th} iteration of Algorithm R is of order $O\left(\frac{1}{\hat{p}[\ell]}\frac{1}{\varepsilon^2}\log\frac{1}{p^*\,\delta}\log|\mathcal{I}|\right)$.*

Proof: Note that Step 3f of the algorithm carefully calculates the number of items to be counted so that the communication of counting doesn't exceed that of sampling. The communication of each iteration is therefore determined by the sample size and is of order $O\left(S^*(\hat{p}[\ell],\varepsilon/4,\delta')\right)\cdot\log|\mathcal{I}|$. The result follows using the bound for S^* as given in Lemma 3.1. ▮

Lemma 3.11 *Define $\ell' \triangleq \ell + \left\lceil \log\left(1+\frac{\log\log W}{\log|\mathcal{I}|}\right)\right\rceil$ and $\delta'' \triangleq \delta'^{(1/\varepsilon^2)}$. By the end of iteration ℓ',with probability at least δ'' , the algorithm has removed from the input all items having $p_X(i) > 2^{-\ell}$ (in the original input).*

Proof: Let X' denote the modified input at the beginning of iteration ℓ' and let Y denote the set of items appearing in X' that had frequency $p_X(i) > 2^{-\ell}$ in the original input. Note that $\hat{p}[\ell'] = 2^{-\ell'}/R[\ell']$. Note also that any item $i \in Y$ has frequency $p_{X'}(i) > 2^{-\ell}/R[\ell'] \geq 2\hat{p}[\ell']$ in the modified input and thus the number of such items is bound by $|Y| < 2^\ell R[\ell']$. The sample size required to "discover" all items in Y with probability $1-\delta''$ is $S^*(\hat{p}[\ell'],1,\delta'')$. Since the sample size used, $S^*(\hat{p}[\ell'],\varepsilon/4,\delta')$, is larger than that, all items having $p_X(i) > 2^{-\ell}$ are in the top-items of the sample with probability at least $1-\delta''$. At last note that since the sample size includes a factor of $\frac{1}{\hat{p}[\ell']} = \left(1+\frac{\log\log W}{\log|\mathcal{I}|}\right)2^\ell R[\ell']$, Step 3f chooses more than $2^\ell R[\ell']$ items and it follows that all items in Y are selected and are later removed from the input. ▮

Lemma 3.12 *On iteration number $\ell'+1$ we have $\hat{p}[\ell'+1] \leq 2^{-\ell'-1}/\pi_\ell$ with probability at least $1-\delta''$.*

Proof: By Lemma 3.11, before the beginning of iteration $\ell'+1$, all items $i \in \mathcal{I}$ having $p_X(i) > 2^{-\ell}$ have been removed with high probability. It follows that $R[\ell'+1] \leq \pi_\ell$ and the result is obtained by definition of $\hat{p}$. ▮

Using the above lemmas we go back to the proof of Theorem 3.9. Let $c'(X)$ denote the communication used throughout the iterations of Algorithm R and let $c''(X)$ denote the communication used after the last iteration (in Step 6). Clearly $c_R(X) = c'(X) + c''(X)$. Let L denote the number of iterations performed by the algorithm on this execution and recall that by Lemma 3.6 L is bound by $(5+\log(1/p^*))$ with high probability. For $\Delta \triangleq 1 + \log\left(1+\frac{\log\log W}{\log|\mathcal{I}|}\right)$ we get

$$c'(X) \;=\; \sum_{\ell=1}^{L} c[\ell] \;<\; \frac{1}{\varepsilon^2}\log\frac{1}{p^*\,\delta}\log|\mathcal{I}|\sum_{\ell=1}^{L}\frac{1}{\hat{p}[\ell]}$$

$$< \quad \frac{1}{\varepsilon^2}\log\frac{1}{p^*\,\delta}\log|\mathcal{I}|\sum_{\ell=1}^{L}2^{\ell}\pi_{\ell-\Delta}$$

where the first inequality is by Lemma 3.10 and the second is by Lemma 3.12. By shifting the indices and using $2^{\Delta}\ \log|\mathcal{I}| = 2\left(\log|\mathcal{I}| + \log\log W\right)$ we get

$$c'(X) < \frac{1}{\varepsilon^2}\log\frac{1}{p^*\,\delta}\left(\log|\mathcal{I}| + \log\log W\right)\left(2 + 2\sum_{\ell=1}^{L}2^{\ell}\pi_{\ell}\right) \tag{1}$$

In order to bound $c''(X)$ we bound $|Q|$ in Step 6. Since at this point, all items in Q have frequency more than some $p' = \Omega\left(p^*\right)$, a trivial upper bound is $|Q| = \frac{1}{p'} = O\left(\frac{1}{p^*}\right)$. Another bound is obtained as follows. Partition Q into subsets Q_ℓ for $1 \le \ell \le \log(1/p')$, where $Q_\ell \triangleq \left\{i \in Q \mid 2^{-\ell+1} < p_X(i) \le 2^{-\ell}\right\}$. Then $|Q| = \sum_{\ell=1}^{\log(1/p')}|Q_\ell| < \sum_{\ell=1}^{\log(1/p')}\dfrac{\pi_{\ell-1} - \pi_\ell}{2^{-\ell}}$, and hence

$$\begin{aligned} c''(X) &= O\left(|Q|\log|\mathcal{I}| \ + \ |Q|\frac{1}{\varepsilon^2}\log\frac{|Q|}{\delta}\log\log W\right) \\ &= O\left(\sum_{\ell=1}^{\log(1/p^*)}2^{\ell}(\pi_{\ell-1} - \pi_\ell)\left(\log|\mathcal{I}| \ + \ \frac{1}{\varepsilon^2}\log\frac{1}{p^*\,\delta}\log\log W\right)\right) . \end{aligned} \tag{2}$$

Combining Eq. (1) and Eq. (2), we conclude that

$$c_R(X) \ = \ O\left(\frac{1}{\varepsilon^2}\log\frac{1}{p^*\,\delta}\left(\log|\mathcal{I}| + \log\log W\right)\sum_{\ell=1}^{\log\frac{1}{p^*}}2^{\ell}\,\pi_\ell\right) . \quad ▮$$

Using Theorem 3.9, we can prove the following corollaries for specific distributions.

Corollary 3.13 *Let X be an instance where item frequencies are geometrically distributed, i.e., $p_X(i) = (1-\lambda)\lambda^{i-1}$ for some constant $0 < \lambda < 1$. Then Algorithm R, when run on input X with parameters $\delta, \varepsilon > 0$ has individual-communication $c_R(X) = O\left(\frac{k}{\varepsilon^2}\log\frac{1}{p^*\,\delta}\left(\log|\mathcal{I}| + \log\log W\right)\right)$.*

Proof: It is straightforward to see that for such input, $\pi_\ell = \frac{2^{-\ell}\lambda}{1-\lambda}$. Next we calculate the sum used in Theorem 3.9. Let $L = \log\frac{1}{p^*}$. Then $\sum_{\ell=1}^{L}2^{\ell}\pi_\ell \ = \ \sum_{\ell=1}^{L}\dfrac{\lambda}{1-\lambda} \ = \ \dfrac{L\lambda}{1-\lambda}$. Since $L = \log\frac{\lambda}{1-\lambda} + k\log\frac{1}{\lambda}$, for constant λ, $L = \Theta\left(k\right)$ and we obtain $\sum_{\ell=1}^{L}2^{\ell}\pi_\ell \ = \ \Theta\left(k\right)$. The result follows immediately by Theorem 3.9. ▮

For the important case of Zipf distribution, we have the following corollary.

Corollary 3.14 *Let X be an instance where item frequencies have Zipf distribution with parameter $a > 1$, i.e., $p_X(i) \triangleq \dfrac{i^{-a}}{h}$, where $a > 1$ and h is the normalization factor. Then Algorithm R, when run on input X with parameters $\delta, \varepsilon > 0$ has individual-communication satisfying $c_R(X) = O\left(\frac{a+1}{a-1}\cdot\frac{k}{\varepsilon^2}\cdot\log\frac{1}{p^*\,\delta}\cdot\left(\log|\mathcal{I}| + \log\log W\right)\right)$.*

Proof: We first find the items summed up in π_ℓ for a given ℓ. Requiring $p(i) \le 2^{-\ell}$ yields $i \ge \left(\frac{2^\ell}{h}\right)^{1/a}$. Next we approximate π_ℓ by $1 - F((2^\ell/h)^{1/a})$ where $F(z) \triangleq \Pr\left\{Z \le z\right\}$ is the probability density function of the

Pareto distribution, which is essentially the continuous version of the Zipf distribution. The approximation is correct up to a constant factor if we use the same exponent $(-a)$ for the Pareto distribution. We obtain that for $a > 1$, $\pi_\ell \approx \frac{2^{-\ell}}{a-1}\left(\frac{2^\ell}{h}\right)^{1/a}$. Let L denote the value of $\log\frac{1}{p^*} = \log\frac{k^{-a}}{h}$. Then

$$\sum_{\ell=1}^{L} 2^\ell \pi_\ell \;=\; \sum_{\ell=1}^{L} 2^{\ell/a} \;=\; \frac{h^{-1/a}}{a-1}\cdot 2^{1/a}\cdot\frac{2^{6/a}h^{1/a}k\;-\;1}{2^{1/a}-1}$$
$$<\; 2^{6/a}\cdot\frac{3(a+1)}{2(a-1)}\cdot k \;=\; O\left(k\cdot\frac{a+1}{a-1}\right)\;,$$

and the result follows from Theorem 3.9. ∎

The results in this section analyze the behavior of R for asymptotically large inputs. As described in the next section, we have tested our algorithms on realistic-size inputs to study the effect of the hidden constants. We have also compared R with other known algorithms from the literature. We remark that Algorithm R turns out to have significantly superior performance.

3.4 A note on average complexity

The main target of the analysis in this section up to now was to bound the worst-case communication complexity of any nodes. In fact, our algorithms are such that the worst case always occurs at the root, which collects all information. Clearly, the average-case complexity is also a very interesting measure; however, the average complexity depends on the topology of the network, whose analysis is beyond the scope of this paper. To get some intuition, consider the following two extreme cases. In one scenario, n nodes are arranged on a line, with the root node being one of the endpoints, and in the other scenario, n nodes are arranged in a complete binary tree of height h (assuming that $n = 2^{h+1} - 1$). To keep things simple, let us roughly analyze basic sampling (Algorithm B), which is the fundamental building block of all our algorithms. Every time a sample of size s is inspected, the root receives s randomly chosen items.

In the line topology, these items are spread uniformly over the line, and thus the number of items that go through a node at distance i from the root is expected to be $s(1-\frac{i}{n})$. It follows that on average, the number of message per node is $s/2$, i.e., the same order of magnitude as the worst case.

Consider now the complete binary tree scenario. Suppose that the tree is of height h. Of the s items in the sample, we expect about $s/2$ items to originate at the leaves, and each of them incurs h transmissions until it gets to the root; about $s/4$ items originate at the next level up, each traveling $h-1$ hops to the root. In general, we have about $s/2^{i+1}$ messages originating at height i, traveling $h-i$ hops each. It follows that the expected total number of message transmissions is

$$\sum_{i=0}^{h}\frac{s}{2^{i+1}}\cdot(h-i)\;<\;s\cdot h\sum_{i=1}^{\infty}2^{-i}\;=\;s\cdot h\,.$$

(Note that the upper bound is quite tight, because $sh/2$ messages are already due to the items at the leaves alone.) This means that the average communication cost is $sh/n = O(s\log n/n)$, i.e., about a $\frac{\log n}{n}$ fraction of the worst case. Note that usually, $s \ll \frac{n}{\log n}$, in which case the average communication complexity is $o(1)$ message.

Our conclusion is that the difference between the worst and average case cost may be negligible or huge, even if we restrict ourselves to bounded degree trees.

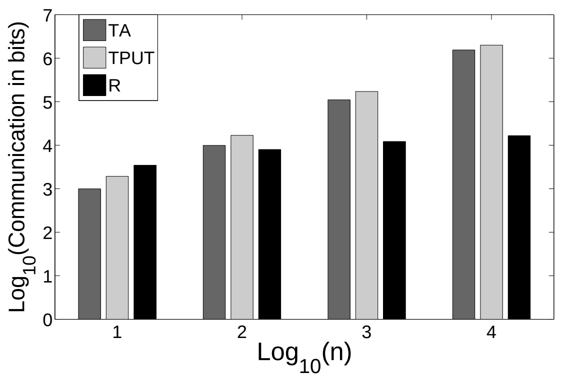


Figure 3: *Communication costs of algorithm R compared to TA and TPUT for random partition of scores in various system scales.*

4 Simulations results

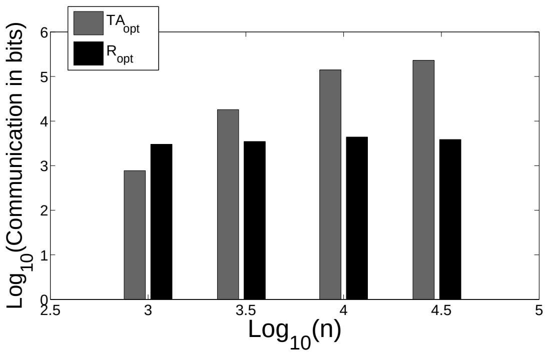
To evaluate the performance of our algorithm in more realistic scenarios, we ran simulations examining the actual number of bits communicated throughout the execution. We compared our algorithm R with the best known algorithms, namely TA [12] and TPUT [6]. We compared the performance of the algorithms when the input is partitioned randomly and when it is partitioned adversarially. Finally, we evaluated the performance of R for various values of p^* (popularity of the k^{th} most popular item) and various values of ϵ (the approximation parameter). Informally, we find that Algorithm R offers a dramatic improvement over TA and TPUT unless the system is very small and the distribution is particularly favorable to TA and TPUT. Moreover, for typical inputs, R behaves better than predicted by our analytical bounds.

4.1 Simulated instances

Topology. Since we are mainly interested in the worst-case communication complexity, and since the worst case occurs at the root node under all algorithms we consider, the topology we simulate is simply the star topology, where the root is connected to all other nodes.

Inputs. The main input to the problem is the distribution of items to nodes. We used randomly generated inputs of various sizes, and our main focus was on inputs generated by Zipf distribution with exponent between 0.8 and 3; all shown charts used Zipf distribution with exponent 1.5 (different exponent values showed qualitatively similar behavior). Typical other parameters were $k = 5$, $\varepsilon = 0.5$ and $\delta = 0.05$. The total number of votes (W) was either 10^6 or 10^7 and the varying parameter is the number of nodes (n). The number of candidate-items was equal to the number of nodes ($|\mathcal{I}| = n$).

Output. All simulations measured the worst-case individual communication (in bits). This is simply the number of bits communicated by the root node.



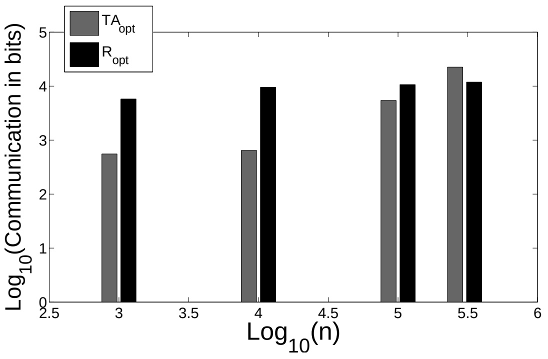


Figure 4: *Communication costs of optimized algorithms for various system scales. Left: with an adversarial partition of scores among nodes. Right: with a random partition of scores among nodes.*

4.2 Comparison with existing algorithms

In Figure 3 we compare algorithms R, TA and TPUT. We used the approximation version of the TA algorithm. We note that TPUT has no approximation version, but in all simulations we ran, TPUT appeared less efficient than TA, even for high accuracy such as $\varepsilon = 0.01$.

Results for adversarial and random partition of scores were similar. As shown in Figure 3, the performance of TA and TPUT for a random partition is rather poor for large systems and Algorithm R is superior even for a system of 100 nodes. R is overwhelmingly superior (by more than two orders of magnitude) for $n > 10000$.

Next, we modified the deterministic algorithms by using loglog counting and allowing them to err. We also assumed the existence of a bounded degree spanning tree, and used unlimited memory at each node (which allows simple aggregation while traversing the tree). These modifications resulted in drastic improvement in TA performance: see Figure 4. The optimized version of TPUT is not shown because it performed far worse than the optimized version of TA. Note that when the input is adversarially partitioned, R beats TA for $n > 2000$. But even when the input is partitioned randomly (which is close to the best case of TA), R performs better for n large enough, where large enough means $n > 200000$ in this case.

4.3 The effect of accuracy and critical frequency on performance

We examined the performance of Algorithms R and algorithm S while varying ε and p^* (for n fixed). The results are given in Figure 5. Obviously, Algorithm R was always superior to Algorithm S. The results for p^* (Figure 5, left) show that Algorithm R is less sensitive to p^* when the input is generated by Zipf distribution. We note that the flat left segment in the graph of S is due to the fact that for small values of p^*, the sample of Algorithm S consists of the entire input.

The effect of the approximation factor ε (seen in Figure 5, right) on the communication of both algorithms is proportional to $\left(\frac{1+\varepsilon}{\varepsilon}\right)^2$, as suggested by our upper bounds. Algorithm R is more efficient due to its improved termination criterion.

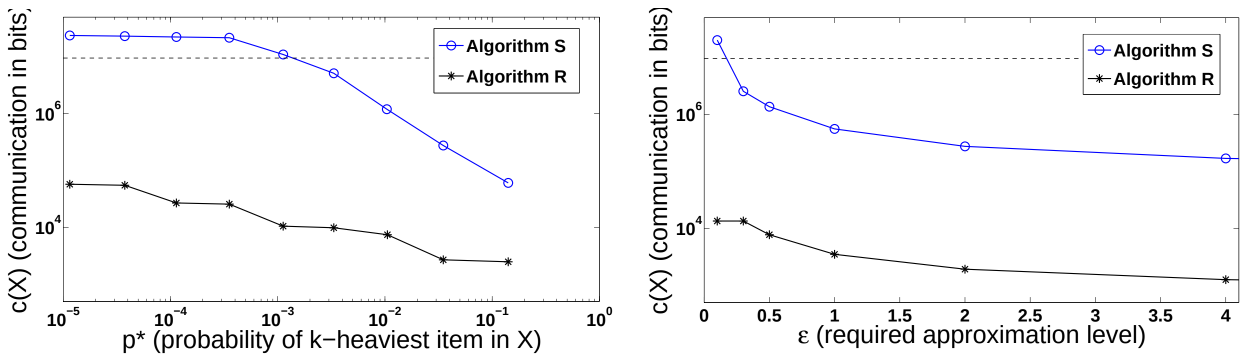


Figure 5: *Communication costs of algorithms S and R. Left: as a function of p^*. Right: as a function of ε. The dashed line marks the communication cost of reading the entire input.*

5 A lower bound on the total cell-probe complexity

In this section we give a lower bound on the *total cell probe complexity* [23, 14] of randomized algorithms for the top-k problem. For a restricted settings, we can deduce a lower bound on the individual communication complexity of such randomized algorithms.

Recall that the cell probe complexity of an algorithm was defined as the total number of input cells accessed by a single node for the worst-case input. Note that the set of all probed cells can be viewed as a sample whose size is the total number of cell-probes.

Theorem 5.1 *For some natural k, let $\varepsilon > 0$, $0 < p^* \le \frac{1}{2k}$, $0 < \delta < 0.014$, and $W = \Omega\left(\frac{1}{\delta(p^*)^2} \cdot \frac{(1+\varepsilon)^2}{\varepsilon^4}\right)$ and let A be any Monte Carlo algorithm. If A outputs a top-k ε-approximation with probability at least $1-\delta$ for any input in $\mathcal{X}(W, n, p^*)$, then its total cell-probe complexity satisfies $E\left[\,\mathrm{CP}_A^T\,\right] = \Omega\left(\frac{1}{p^*\,\varepsilon^2} \cdot \log\frac{1}{\delta}\right)$.*

The proof is deferred to the end of this section.

Star topology and "smart dust" systems. Theorem 5.1 demonstrates the optimality of Algorithm B with respect to the *total* cell-probe complexity but not the *individual* cell-probe complexity. Consider a setting where the system has star topology, i.e., all nodes are connected to a root node, and suppose that each node holds a single cell. This model is a reasonable abstraction of the setting in sensor networks using *passive communication*. In these systems, the only communication is between a powered base-station and a sensor, forming a star topology (as opposed to the common spanning tree). When compared with *active communication* techniques, passive communication is much more energy efficient and suitable for very small devices, such as "smart dust" systems (see e.g., [22]). Now, for these systems, Theorem 5.1 says that any algorithm that satisfies the conditions of the theorem has individual communication complexity at least $\Omega\left(\frac{1}{p^*\,\varepsilon^2} \cdot \log\frac{1}{\delta}\right)$.

We stress that while our model assumes certain routing capabilities, our algorithms only use a broadcast-convergecast scheme rooted by the root node. Such communication is suitable for the star topology.

Proof of Theorem 5.1: The main steps of the proof are as follows. We use Yao's Minimax Principle to show the lower bound on Monte Carlo algorithms. To this end, we prove a lower bound on the expected sample size used by any deterministic algorithm when the input is drawn at random from a certain distribution of our choice. In our distribution, the gap between the popularity of a top-k set and any other item is more than ε, so that the correct output for any ε-approximation algorithm is precisely the top-k set. Furthermore, the local view of each node is just a random subset of the items. We then prove that any algorithm which does not output the top-k set of its sample is doomed to err with probability at least $1/2$. Finally, we show that the top-k set of the sample is the top-k set of the input with probability at least $1-\delta$ only if the sample is large enough.

Consider some input distribution D. For the sake of brevity we will use $X \in D$ to denote inputs in the support of D (i.e. $X : \Pr_D\{X\} > 0$). Let $\mathcal{A}(D,\varepsilon,\delta)$ be the set of all algorithms which output a top-k ε-approximation with probability at least $1-\delta$ for a random input X drawn by D. To formalize this, we define the proximity, $\rho(\mathcal{Q})$ of a set $\mathcal{Q}$ to $\mathrm{top}(k,X)$; $\rho(\mathcal{Q})$ is defined as the minimal ε for which $\mathcal{Q}$ is a top-k ε-approximation, i.e.,

$$\rho(\mathcal{Q}) \triangleq \frac{\max\{p(j) \mid j \in \mathcal{I}\backslash\mathcal{Q}\}}{\min\{p(i) \mid i \in \mathcal{Q}\}} - 1\,.$$

The class $\mathcal{A}(D,\varepsilon,\delta)$ is then defined as

$$\mathcal{A}(D,\varepsilon,\delta) \triangleq \left\{A \,:\, \Pr_{X\sim D}\{\rho(A(X)) \le \varepsilon\} \ge 1-\delta\right\}\,.$$

Let $\mathcal{A}'$ denote the set of all deterministic algorithms in $\mathcal{A}(D,\varepsilon,\delta)$ and assume that we have a lower bound b of the form: $E_{X\sim D}\left[\mathrm{cp}_A^T(X)\right] \ge b$, for any deterministic algorithm $A \in \mathcal{A}'$. Using Yao's Minimax Principle, we say that for any random algorithm A_R satisfying $A_R \in \mathcal{A}(D,\varepsilon,\frac{\delta}{2})$ the total cell-probe complexity satisfies $E\left[\mathrm{CP}_{A_R}^T\right] \ge \max\left\{E\left[\mathrm{cp}_{A_R}(X)\right] \mid X \in D\right\} \ge \dfrac{b}{2}$. Lemma 5.2 that follows gives such a lower bound for deterministic algorithms with $b = \Omega\left(\dfrac{1}{p^*}\cdot\dfrac{(1+\varepsilon)}{\varepsilon^2}\cdot\log\dfrac{1}{2\delta}\right)$ giving the result of the theorem. ∎

Lemma 5.2 *Let k be a natural number. Let $\varepsilon > 0$, $0 < p^* \le \frac{1}{2k}$, $0 < \delta < 0.014$, $W = \widetilde{\Omega}\left(\frac{1}{\delta p^{*2}}\cdot\frac{(1+\varepsilon)^2}{\varepsilon^4}\right)$. Then there exists an input distribution D with support$(D) \subseteq \mathcal{X}(W,n,p^*)$ such that for any deterministic algorithm A we have that if A outputs a top-k ε-approximation on an input X chosen randomly from D with probability at least $1-\delta$, then $E_{X\sim D}\left[\mathrm{cp}_A^T(X)\right] = \Omega\left(\frac{1}{p^*}\cdot\frac{(1+\varepsilon)}{\varepsilon^2}\cdot\log\frac{1}{2\delta}\right)$.*

Proof: For the given values $W, k, p^*, \varepsilon > 0 \quad (0 < p^* \le \frac{1}{2k})$, construct the input distribution D as follows. Let $U(X)$ be the uniform distribution above the permutations of X and define $q \triangleq \frac{p^*}{\alpha}$ for some $\alpha > 1+\varepsilon$.

- Randomly choose a permutation of items $\vec{I} \sim U(\mathcal{I})$. We use i to denote both the item and its index in $\vec{I}$.
- Given the item-indices determined by $\vec{I}$, fix the aggregate weight of each item by:

$$w(i) \quad = \quad \begin{cases} Wp^* & \text{for } 1 \le i \le k \\ Wq \quad (= \frac{Wp^*}{\alpha}) & \text{for } k < i \le k + \frac{1-p^*k}{q} \\ 0 & \text{otherwise} \end{cases}$$

- Let X_0 be an arbitrary vector where each item i has exactly $w(i)$ cells and let $\langle v_1,\ldots,v_n\rangle$ be some ordering of the nodes. To obtain the random input X, take a random permutation of X_0 and partition it evenly among the nodes. i.e. Let $X_0' \sim U(X_0)$ and let $X_{v_\ell} \triangleq \left\{X_0'[\frac{\ell-1}{n}W],\ldots,X_0'[\frac{\ell}{n}W]\right\}$.

Note that for any input X in the support of D, only the top-k set is a top-k ε-approximation. Note also that since we use a random permutation of the items, the probabilities are symmetric with respect to item-identities.

Let A be any deterministic algorithm which outputs a top-k ε-approximation with probability at least $1-\delta$ for a random input $X \sim D$. Let the sample of X, $\mathcal{S}_A(X)$ denote the cells actually probed throughout the execution of A on input X. Since A is deterministic, $\mathcal{S}$ is a deterministic function of X and the output of A

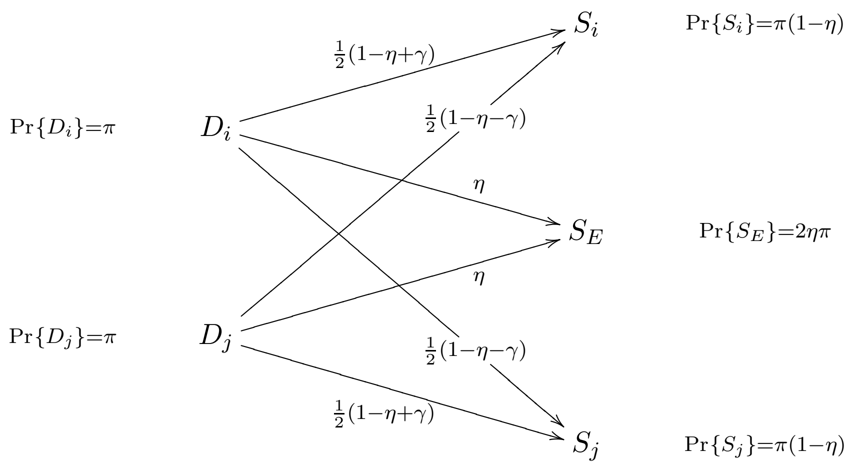


Figure 6: Transition probabilities during the sampling process.

is a deterministic function of $\mathcal{S}$. We can therefore denote by $A(\mathcal{S})$ the output of A when $\mathcal{S}_A(X) = \mathcal{S}$. We first argue in that if A does not output the top k items in the sample $\mathrm{top}(k, \mathcal{S})$, then A cannot be correct with probability more than $1/2$. We then show that when A outputs the top k items in the sample, the sample has to be large to get small error probability. Combining the two arguments we get result of the lemma.

Lemma 5.3 *Let A be any deterministic top-k-approximation algorithm. For a random input X drawn by D, if the resulting sample $\mathcal{S} = \mathcal{S}_A(X)$ is such that $A(\mathcal{S}) \neq \mathrm{top}(k, \mathcal{S})$, then A errs with probability at least $\frac{1}{2}$.*

Proof: For a given sample, let $\vec{w}^{\mathcal{S}} = \left(w^{\mathcal{S}}(i_1) \geq w^{\mathcal{S}}(i_2) \geq \ldots \geq w^{\mathcal{S}}(i_{|I|})\right)$ be the sorted vector of weights seen in the sample and let $\vec{I}^{\mathcal{S}} = \left(i_1, i_2, \ldots, i_{|I|}\right)$ denote the items matching the entries of $\vec{w}^{\mathcal{S}}$. We say that A has some deterministic function $J = f(\vec{w}), J = \{j_1, \ldots, j_k\}$ such that A outputs the set $\left\{\vec{I}^{\mathcal{S}}[j_1], \ldots, \vec{I}^{\mathcal{S}}[j_k]\right\}$ for any input X. For example, the algorithm which always outputs $\mathrm{top}(k, \mathcal{S})$, has $J = \{1, 2, \ldots, k\}$. Let $\vec{w}$ be a weights vector for which A outputs a set Y such that $Y \neq \mathrm{top}(k,^{\mathcal{S}}\,)$. We now show that $\Pr\left\{A \text{ errs} \mid \vec{w}^{\mathcal{S}} = \vec{w}\right\} > \frac{1}{2}$.

Fix such a $\vec{w}$. Let $i \in \mathrm{top}(k, \mathcal{S}) \setminus Y$ and let $j \in Y \setminus \mathrm{top}(k, \mathcal{S})$. Since there is only one correct answer in our input distribution, Y is a correct answer if and only if $\mathrm{top}(k, \mathcal{S})$ is a wrong answer in which case there exists such a pair (i, j) which also proves $\mathrm{top}(k, \mathcal{S})$ to be a wrong answer:

$$\begin{aligned}(i,j) \quad : \quad & i \in \mathrm{top}(k,\mathcal{S})\backslash Y, \quad j \in Y\backslash\mathrm{top}(k,\mathcal{S}), \\ & p_{\mathcal{S}}(j) \leq p_{\mathcal{S}}(i), \quad p_X(j) > (1+\varepsilon)p_X(i) > p_X(i)\end{aligned}$$

Define $\vec{I'}$ to be the instance generated from $\vec{I}$ by switching the names of items i and j. Call $\vec{I'}$ the ij-mirror of $\vec{I}$. Consider the set $\mathcal{X}$ of all possible inputs which, for the sampling function of A, result in either $(\vec{w}, \vec{I})$ or its ij-mirror $(\vec{w}, \vec{I'})$. Let D_i denote the set of such inputs where $p_X(i) > p_X(j)$ and let D_j denote the inputs where $p_X(i) < p_X(j)$. Further, let S_i denote the set of possible samples where $p_{\mathcal{S}}(i) > p_{\mathcal{S}}(j)$, and let S_j denote the samples where $p_{\mathcal{S}}(i) \geq p_{\mathcal{S}}(j)$ and by S_E the samples where $p_{\mathcal{S}}(i) = p_{\mathcal{S}}(j)$.

Figure 5 illustrates the relations of the weights of i, j during the sampling process. Let π denote the probability of D_i. By symmetry we have $\Pr\{D_j\} = \pi$ too. It is straightforward to see that $\Pr\{S_i \mid D_i\} > \Pr\{S_j \mid D_i\}$, and similarly that $\Pr\{S_j \mid D_j\} > \Pr\{S_i \mid D_j\}$. Define: $\gamma \triangleq \Pr\{S_i \mid D_i\} - \Pr\{S_j \mid D_i\}$, and $\eta \triangleq \Pr\{S_E \mid D_i\} = \Pr\{S_E \mid D_j\}$. With this notation, we have the following identities.

$$\begin{aligned}
\Pr\{S_i \mid D_i\} &= \Pr\{S_j \mid D_j\} = \tfrac{1}{2}(1-\eta+\gamma) \\
\Pr\{S_j \mid D_i\} &= \Pr\{S_j \mid D_i\} = \tfrac{1}{2}(1-\eta-\gamma) \\
\Pr\{S_i\} &= \Pr\{S_j\} = \pi(1-\eta) \\
\Pr\{S_E\} &= 2\eta\pi\ ,
\end{aligned}$$

and therefore, since $\gamma > 0$, we obtain

$$\begin{aligned}
\Pr\{Y \text{ is correct} \mid \vec{w}\} &< \Pr\{D_j \mid S_i \cup S_E\} \qquad (3)\\
&= \frac{\Pr\{D_j\}\Pr\{S_i \cup S_E \mid D_j\}}{\Pr\{S_i\} + \Pr\{S_E\}} \\
&= \frac{\pi\left(\frac{1}{2}(1-\eta-\gamma) \quad + \quad \eta\right)}{\pi(1-\eta) \quad + \quad 2\eta\pi} \\
&= \frac{1}{2}\left(\frac{1+\eta-\gamma}{1+\eta}\right) < \frac{1}{2}\ . \qquad \blacksquare
\end{aligned}$$

Lemma 5.4 *Let A be any deterministic algorithm which probes a deterministic set of S distinct cells and outputs the top-k-set computed by this sample. For a random input X drawn by D, if A has error probability $\delta < \frac{1}{3e^3}$ and if the input size satisfies $W = \tilde{\Omega}\left(\frac{1}{\delta p^{*2}} \cdot \frac{(1+\varepsilon)^2}{\varepsilon^4}\right)$ then S satisfies*

$$S = \Omega\left(\frac{1}{p^*} \cdot \frac{\alpha}{(\alpha-1)^2} \cdot \log\frac{1}{4\delta}\right)\ .$$

Proof: Recall that the input X is a random permutation of X_0 and consider the cells of X_0 probed by A. Actually, an execution of A may probe any one of the possible ordered-sets, of X_0-cells (where each set contains S distinct X_0-cells). Note that all those ordered-sets are equally likely to be used by A. Since A is deterministic we can define the collection G of such ordered-sets for which A is correct. Since A has error probability δ, the size of G is a fraction of $(1-\delta)$ from all possible ordered-sets of S distinct X_0-cells.

Now, let A_r be the algorithms that selects at random a sample of S cells, with replacement, probes those cells and outputs the top-k set of the sample. Note that $A_r(S)$ is the same as A except it takes a random sample with replacement instead of a deterministic set of distinct cells. It follows that whenever A_r happens to probe a set of X_0-cells from G, A_r outputs a correct answer just as A would. Note also that for A_r too, all ordered-sets of S distinct X_0 cells are equally probable. Let $\mathcal{S}$ denote the ordered-set of X_0-cells sampled by A_r. Then

$$\Pr\{\text{there are repetitions in } \mathcal{S}\} \le \sum_{\ell=1}^{S} \frac{\ell-1}{W} < \frac{S^2}{2W}\ .$$

Since $\Pr\{\mathcal{S} \in G \mid \text{no repetitions in } \mathcal{S}\} = 1-\delta$, we have that $\Pr\{\mathcal{S} \in G\} \ge (1-\delta)\left(1 - \frac{S^2}{2W}\right)$, and therefore

$$\Pr\{A_r \text{ errs}\} \le 1 - \Pr\{\mathcal{S} \in G\} \le \delta + \frac{S^2}{2W}(1-\delta) < \delta + \frac{S^2}{2W}\ ,$$

which implies that for $S \le \sqrt{2W\delta}$, $\Pr\{A_r \text{ errs}\} < 2\delta$. Lemma 5.5 that follows gives a bound on the sample size used by A_r. Using the relation between the error probabilities of A and A_r, we deduce that the number of cells probed by A must satisfy $S = \Omega\left(\frac{1}{p^*} \cdot \frac{\alpha}{(1-\alpha)^2} \cdot \ln\frac{1}{4\delta}\right)$ $\blacksquare$

Lemma 5.5 *For $\delta < \frac{1}{3e^3}$, there exists a value $S_{min} = \Theta\left(\frac{1}{p^*} \cdot \frac{\alpha}{(1-\alpha)^2} \cdot \ln \frac{1}{2\delta}\right)$ such that if A_r is ran on an input chosen randomly by D and it uses a sample smaller than S_{min}, then A_r errs with probability at least δ.*

Proof: For a specific execution of A_r we have a specific input X, and a specific sample $\mathcal{S}$ of cells probed by A_r. For this input, let $i \in \mathrm{top}(k, X)$ and some $j \in \mathcal{I} \setminus \mathrm{top}(k, X)$ and define the *advantage within the sample* of i over j to be $d_{ij}(\mathcal{S}) \triangleq w^{\mathcal{S}}(i) - w^{\mathcal{S}}(j)$. For a permutation $Z = (z_1, \ldots, z_S)$ of $\mathcal{S}$, let $\Pr\{Z \mid X\}$ denote the probability that Z is the sequence (ordered multiset) of cells probed during the execution of A_r on input X. Clearly, $\Pr\{Z \mid X\} = \prod_{\ell=1}^{S} p_X(z_\ell)$. Note that all permutations of $\mathcal{S}$ are equally probable.

We now bound the error probability of A_r. We distinguish between three cases, depending on the value of d_{ij}. If $d_{ij} < 0$ then $w^{\mathcal{S}}(i) < w^{\mathcal{S}}(j)$. Since $i \in \mathrm{top}(k, X)$ and $j \notin \mathrm{top}(k, X)$, it is obvious that is this case $\mathrm{top}(k, \mathcal{S}) \neq \mathrm{top}(k, \mathcal{S})$ and hence A_r always errs in this case.

Next, consider a specific input X' and a specific ordered-set Z' sampled by A_r such that $d_{ij}(Z') = 0$. Suppose that A_r outputs i and not j (which is correct)). Consider the ij-mirror input $X'' \triangleq M_{ij}(X')$. By definition of D, $\Pr\{X'\} = \Pr\{X''\}$. Since for $Z'' \triangleq M_{ij}(Z')$, we have $\Pr\{Z' \mid X'\} = \Pr\{Z'' \mid X''\}$ and since $d_{ij} = 0$, we get $\Pr\{Z' \mid X''\} = \Pr\{Z'' \mid X''\}$ and therefore $\Pr\{Z', X''\} = \Pr\{Z', X''\}$. Once the sample is fixed, A_r is deterministic so if the input is X'' and the probed cells are the same Z', A_r outputs the same answer as it did for X', Z' but in this case it would be wrong. For each such case where $d_{ij} = 0$ and A_r is correct there exists at least one mirror-case where A_r errs. We get that $\Pr\{\text{A errs} \mid d_{ij}(\mathcal{S}) = 0\} \geq \frac{1}{2}$.

Finally, we bound the overall error probability of A_r by the probability that $d_{ij} < 0$. Let π denote the probability $\Pr\{d_{ij} > 0\}$. We have that

$$\Pr\{A_r \text{ errs}\} \;\geq\; \Pr\{d_{ij} < 0\} + \frac{1}{2}\Pr\{d_{ij} = 0\} \;\geq\; \frac{1}{2}\Pr\{d_{ij} \leq 0\} \;=\; \frac{1-\pi}{2} .$$

Note that if $\pi \leq \frac{1}{2}$ we have $\Pr\{A_r \text{ errs}\} \geq \frac{1}{4}$, and we are done. For the remainder of the proof, assume $\pi > \frac{1}{2}$. For any unordered sample $\mathcal{S} = (\nu_1, \ldots, \nu_S)$, let $S' \triangleq M_{ij}(\mathcal{S})$ be the mirror sample. Clearly $d_{ij}(\mathcal{S}') = -d_{ij}(\mathcal{S})$. Next we show the relation between the probability of a sample $\mathcal{S}$ and the probability of its ij-mirror, $\mathcal{S}'$. Then by bounding the probability of a set of samples all having $d_{ij} > 0$, we get a bound for the probability of the mirror-samples having $d_{ij} < 0$.

Let $\mathrm{perms}(\mathcal{S})$ denote the number of distinct permutations of $\mathcal{S}$. Clearly we have $\mathrm{perms}(\mathcal{S}) = \mathrm{perms}(\mathcal{S}')$. For brevity, let $a \triangleq w^{\mathcal{S}}(i)$ and $b \triangleq w^{\mathcal{S}}(j)$. Clearly $d_{ij} = a - b$. Furthermore, For a specific input X we have

$$\begin{aligned}
\Pr\{\mathcal{S} \mid X\} &= \mathrm{perms}(\mathcal{S}) \cdot \prod_{\ell=1}^{S} p_X(v_\ell) \\
&= \mathrm{perms}(\mathcal{S}) \cdot p_X(i)^a \cdot p_X(j)^b \cdot \prod_{\ell \in \{1..S\}, v_\ell \notin \{i,j\}} p_X(v_\ell) p_X(v_\ell) . \\
\text{Similarly,} \quad \Pr\left\{\mathcal{S}' \mid X\right\} &= \mathrm{perms}(\mathcal{S}') \cdot \prod_{\ell=1}^{S} p_X(v'_\ell) \\
&= \mathrm{perms}(\mathcal{S}) \cdot p_X(i)^b \cdot p_X(j)^a \cdot \prod_{\ell \in \{1..S\}, v_\ell \notin \{i,j\}} p_X(v_\ell) , \\
\text{yielding,} \quad \Pr\left\{\mathcal{S}' \mid X\right\} &= \Pr\{\mathcal{S} \mid X\} \cdot p_X(i)^{b-a} \cdot p_X(j)^{a-b} \\
&= \Pr\{\mathcal{S} \mid X\} \cdot \left(\frac{1}{\alpha}\right)^{d_{ij}} .
\end{aligned}$$

We now lower bound the probability of receiving a sample having $d_{ij} > 0$. Since $\mathcal{S} = (\nu_1, \ldots, \nu_S)$ is a random sample of X, each cell ν_ℓ can be considered a random variable distributed by the empirical item-probabilities in X. This way, d_{ij} is a random variable satisfying $d_{ij}(\mathcal{S}) = \sum_{\ell=1}^{S} d'(\nu_\ell)$ where $d'(\nu_\ell)$ are i.i.d. random variables with

$$\Pr\left\{d'(\nu_\ell) = d' \mid X\right\} = \begin{cases} p_X(i) & \text{if } d' = 1 \\ p_X(j) & \text{if } d' = -1 \\ 1 - p_X(i) - p_X(j) & \text{if } d' = 0 \end{cases}$$

Since this gives the same distribution for any X produced during our input distribution, we can eliminate the conditioning. The expectation and variance of $d(\nu_\ell)$ and $d_{ij}(\mathcal{S})$ are:

$$\begin{array}{rlrl} E\left[\, d'(\nu_\ell) \,\right] & = q(\alpha - 1) & \mu_d & = E\left[\, d_{ij} \,\right] = S(\alpha - 1)q \\ \text{Var}\left(d'(\nu_\ell)\right) & = q\alpha + q - (q\alpha - q)^2 & {\sigma_d}^2 & = \text{Var}\left(d_{ij}\right) = S(q\alpha + q - (q\alpha - q)^2) \\ & & {\sigma_d}^2 & < E\left[\, (d_{ij})^2 \,\right] = S(\alpha + 1)q \end{array}$$

We also note that by definition $-S \le d_{ij} \le S$. By Chebyshev inequality, if we pick a random sample of size S, we get:

$$\Pr\left\{0 < d_{ij} < 7\mu_d\right\} \;=\; \pi - \Pr\left\{d_{ij} - \mu_d \ge 6\mu_d\right\} \;>\; \frac{1}{2} - \frac{{\sigma_d}^2}{6{\mu_d}^2} \;>\; \frac{1}{2} - \frac{(\alpha + 1)}{Sq(\alpha - 1)^2 6^2} \;.$$

Since the transformation from $\mathcal{S}$ to $\mathcal{S}'$ is one-to-one, we can apply it on the set of n distinct samples and get another set of n distinct mirror-samples. Replacing i and j in all samples for which $0 < d_{ij} < 7\mu_d$ we get the set of all possible samples satisfying $-7\mu_d < d_{ij} < 0$. The probability of the transformed set satisfies:

$$\begin{array}{rcl} \Pr\left\{-7\mu_d < d_{ij} < 0\right\} & > & \Pr\left\{0 < d_{ij} < 7\mu_d\right\} \cdot \left(\dfrac{1}{\alpha}\right)^{7\mu_d} \\ & > & \left(\dfrac{1}{2} - \dfrac{(\alpha + 1)}{Sq(\alpha - 1)^2 6^2}\right)\left(\dfrac{1}{\alpha}\right)^{7\mu_d} \;. \end{array}$$

For $S \ge S_1$, where $S_1 \triangleq \frac{(\alpha+1)}{6q(\alpha-1)^2}$, we can say that

$$\begin{array}{rcl} \Pr\left\{-7\mu_d < d_{ij} < 0\right\} & > & \dfrac{1}{3}\left(\dfrac{1}{\alpha}\right)^{7\mu_d} \\ & = & \dfrac{1}{3}\exp\left(-7\ln\alpha\mu_d\right) \\ & > & \dfrac{1}{3}\exp\left(-7(\alpha - 1)\mu_d\right) \\ & = & \dfrac{1}{3}\exp\left(-7Sq(\alpha - 1)^2\right) \;. \end{array}$$

And for $S_1 \le S < S_2$ where $S_2 \triangleq \frac{1}{2q} \cdot \frac{1}{(\alpha-1)^2} \cdot \ln\frac{1}{3\delta}$, we get

$$\begin{array}{rl} \forall X \in D, & \Pr\left\{A_r \text{ errs} \mid X\right\} \\ = & \Pr\left\{S : \text{A errs for } S \mid X\right\} \\ = & \Pr\left\{S : \exists(i,j), i \in \text{top}(k, X), j \notin \text{top}(k, X), d_{ij}(\mathcal{S}) < 0 \mid X\right\} \\ > & \Pr\left\{\mathcal{S} : -7\mu_d < d_{ij}(\mathcal{S}) < 0 \mid X\right\} \quad \text{for a single pair } (i,j) \\ > & \delta \;. \end{array} \tag{4}$$

The above applies for cases where $S_1 < S_2$ which is satisfied for $\delta < \frac{1}{3e^3}$, $\alpha < \frac{11}{7}$. We have shown a specific sample size S_2 for which A_r has an error probability larger than δ. We argue that there is no value $S' < S_2$ for which A_r has a smaller error probability. If there were, than we could define another algorithm A'_r, which takes a random sample of size S_2 and outputs the top-k-set by the first S' probed cells only thus achieving error probability smaller then δ. Lemma 5.3 covers all cases where the output of A'_r is different from that of $A_r(S_2)$ so A'_r cannot have such error probability - a contradiction. This completes the proof for $\delta < \frac{1}{3e^3}, \quad \alpha < \frac{11}{7}$.

As for other values of α, we note that in this case the result of the lemma is for sample size of order $\Theta\left(\frac{1}{p^*}\ln\frac{1}{\delta}\right)$ for such small samples there is a good chance that one of the top-k items would be completely missing from the sample. Due to page limitations we omit the proof of this. ∎

We now go back to the proof of Lemma 5.2. Lemma 5.3 has shown that for a random $X \sim D$, whenever A outputs something different than the top-k-set of probed-cells, it is probably wrong. This means that an algorithm A that satisfies the conditions of the lemma must output the top-k set of probed-cells most of the time. Lemma 5.4 showed the for the latter case, A must probe at least $S = \Omega\left(\frac{1}{p^*}\cdot\frac{\alpha}{(\alpha-1)^2}\cdot\log\frac{1}{4\delta}\right)$ to have error probability under δ. Since this lower bound is concave in δ, there is no better option than probing the same number of cells for all inputs.

Even if we assume that A probes zero cells whenever it is wrong, we get that for the conditions of the lemma and a random input $X \sim D$, if the error probability of A is δ, then the expected number of probed cells is of the same order: $E_{X\sim D}\left[\,\mathrm{cp}(X)\,\right] = \Omega\left(\frac{1}{p^*}\cdot\frac{\alpha}{(\alpha-1)^2}\cdot\log\frac{1}{4\delta}\right)$. ∎

6 Conclusions and future work

In this paper we have proposed algorithms solving the top-k problem by adaptive sampling. The communication complexity of our algorithms does not depend on the way the input is partitioned in the network: only the global statistics affect the complexity. Our final algorithm performs particularly well when the global statistics are far from flat. We have tested our algorithm by simulation and found empirical support for our analytical claims. Although our study is mainly theoretical, simulation results indicate that our algorithm is rather practical and can be very useful in real-life scenarios. Future work may extend our algorithm to specific models, where spatial and temporal dependence among different sensors holds (e.g., nearby nodes have similar readings).

From the theoretical viewpoint, we think that it is very interesting to extend our lower bound to the case of interactive algorithms. We conjecture that our Algorithm R is nearly optimal in this more general model.

7 Acknowledgments

This research was supported in part by Israel Ministry of Science and Technology contract 3-941.

References

[1] H. Attiya and J. Welch. *Distributed Algorithms*. McGraw-Hill Publishing Company, UK, 1998.

[2] B. Babcock and C. Olston. Distributed top-k monitoring. In *Proc. 2003 ACM SIGMOD*, pages 28–39, 2003.

[3] P. Bak. *How Nature Works: The science of self-organized criticality*. Springer-Verlag, New York, 1996.

[4] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. Progressive distributed top k retrieval in peer-to-peer networks. In *Proc. 21st Int. Conf. on Data Engineering*, pages 174–185, 2005.

[5] N. Bruno, L. Gravano, and A. Marian. Evaluating top-k queries over web-accessible databases. In *Proc. 18th Int. Conf. on Data Engineering*, 2002.

[6] P. Cao and Z. Wang. Efficient top-k query calculation in distributed networks. In *Proc. 23rd Ann. ACM Symp. on Principles of Distributed Computing*, pages 206–215, 2004.

[7] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proc. 20th Int. Conf. on Data Engineering*, pages 449–460, Apr. 2004.

[8] G. Cormode, M. N. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *Proc. 2005 ACM SIGMOD*, pages 25–36, 2005.

[9] P. Dagum, R. M. Karp, M. Luby, and S. Ross. An optimal algorithm for Monte Carlo estimation. *SIAM J. Comput.*, 29(5):1484–1496, 2000.

[10] M. Durand and P. Flajolet. Loglog counting of large cardinalities (extended abstract). In *Algorithms: ESA 11th Ann. European Symp.*, pages 605–617, 2003.

[11] R. Fagin. Combining fuzzy information: an overview. *SIGMOD Record*, 31(2):109–118, 2002.

[12] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proc. 20th ACM Symp. on Principles of Database Systems*, 2001.

[13] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proc. SIGCOMM '99*, pages 251–262, New York, NY, USA, 1999. ACM Press.

[14] M. Fredman and M. Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, May 1989.

[15] M. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *Proc. 23rd ACM Symp. on Principles of Database Systems*, pages 275–285, 2004.

[16] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, San Mateo, CA, 1995.

[17] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proc. 2003 ACM SIGMOD*, pages 491–502, 2003.

[18] S. Michel, P. Triantafillou, and G. Weikum. Klee: A framework for distributed top-k query algorithms. In *Proc. 31st Int. Conf. on Very Large Data Bases*, pages 637–648, 2005.

[19] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys '04: Proc. 2nd international conference on Embedded networked sensor systems*, pages 250–262, 2004.

[20] B. Patt-Shamir. A note on efficient aggregate queries in sensor networks. In *Proc. 23rd Ann. ACM Symp. on Principles of Distributed Computing*, pages 283–289, 2004.

[21] A. Silberstein, R. Braynard, C. Ellis, K. Munagala, and J. Yang. A sampling-based approach to optimizing top-k queries in sensor networks. In *Proc. 22nd Int. Conf. on Data Engineering*, 2006.

[22] B. Warneke. Miniaturizing sensor networks with mems. In M. Ilyas and I. Mahgoub, editors, *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, pages 5–1 – 5–19. CRC Press, 2004.

[23] A. C.-C. Yao. Should tables be sorted? *J. ACM*, 28(3):615–628, 1981.

[24] Y. Yao and J. Gehrke. The Cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record*, 31(3):9–18, Sept. 2002.

[25] D. Zeinalipour-Yazti, Z. Vagena, D. Gunopulos, V. Kalogeraki, V. Tsotras, M. Vlachos, N. Koudas, and D. Srivastava. The threshold join algorithm for top-k queries in distributed sensor networks. In *Proc. 2nd Int. Workshop on Data Management for Sensor Networks*, pages 61–66, 2005.

APPENDICES

A Proofs for Algorithm BoundCount

Corollary 2.2 *For given $10^{-5} \le \varepsilon \le 1$ and $\delta > 0$, Algorithm BoundCount outputs an estimate $\hat{w}$ of $w(i)$ such that*

$$\Pr\left\{\frac{1}{w(i)}\left|\hat{w} - w(i)\right| < \varepsilon\right\} \ge 1 - \delta.$$

The communication complexity of Algorithm BoundCount is

$$O\left(\log|\mathcal{I}| + \frac{1}{\varepsilon^2}\log\frac{1}{\delta}\log\log w(i)\right).$$

Also, if the algorithm ran M iterations in Step 3, then for any $\zeta > 10^{-5}$,

$$\Pr\left\{\frac{1}{w(i)}\left|\hat{w} - w(i)\right| < \zeta\right\} \ge 1 - \exp\left(-\Omega\left(M\zeta^2\right)\right).$$

Proof: For each $\hat{w}_\ell$, consider the random variable $z_\ell \triangleq \hat{w}_\ell / w(i)$. z_ℓ is a (useless) estimate of $z \triangleq E\left[\,\hat{w}_\ell\,\right]/w(i)$ and by Fact 2.1, it satisfies $\left|E\left[\,z_\ell\,\right] - 1\right| < 10^{-6}$, and $\mathrm{Var}\left(z_\ell\right) = O(1)$. Now consider the random variable $\hat{z} \triangleq \hat{w}/w(i)$ where $\hat{w}$ is the output of the algorithm. Note that $\hat{z}$ is actually the average of the M independent variables $\{z_1, \ldots, z_M\}$. we get from Bernstein's Inequality that $\Pr\left\{|\hat{z} - 1| < 10^{-6} + \varepsilon/2\right\} > 1 - \exp\left(-M\varepsilon^2/6\right)$. This translates directly to the average of the M original estimates, and the result of the corollary follows. ∎

B Proofs for Algorithm B

Lemma 3.1 *There exists a function $S^*(p^*, \varepsilon, \delta) = \Theta\left(\frac{1}{p^*\,\varepsilon^2} \cdot \ln\frac{1}{p^*\,\delta}\right)$, such that for any input $X \in \mathcal{X}(W, n, p^*)$, the top-$k$ elements of a random sample of size at least $S^*(p^*, \varepsilon, \delta)$ is an ε-approximation of $\mathrm{top}(k, X)$ with probability at least $1 - \delta$.*

Proof: We say that a pair (i, j) of items is *important* if $i \in \mathrm{top}(k, X)$, $j \notin \mathrm{top}(k, X)$, and $p_X(i) > (1 + \varepsilon) \cdot p_X(j)$.

A pair (i, j) of items is said to be *poorly-swapped* in $\mathcal{S}$ if it is important AND $p_{\mathcal{S}}(i) \le p_{\mathcal{S}}(j)$. Note that $\mathrm{top}(k, \mathcal{S})$ is a top-k ε-approximation if and only if there are no poorly-swapped pairs in $\mathcal{S}$.

For each item $i \in \mathrm{top}(k, X)$ we define a threshold value t_i satisfying $\frac{p_X(i)}{1+\varepsilon} < t_i < p_X(i)$ and for each item $j \notin \mathrm{top}(k, X)$ we define a threshold value t'_j by $t'_j \triangleq \min\left\{t_i \mid (i, j) \text{ is important}\right\}$. We denote by E_i the event of receiving a sample $\mathcal{S}$ where $p_{\mathcal{S}}(i) \le t_i$. This event is also referred to as "i violates its threshold". We denote by E'_j the event of receiving a sample $\mathcal{S}$ where $p_{\mathcal{S}}(j) \ge t'_j$. This event is also referred to as "j violates its threshold".

From the above properties of the thresholds, if no $i \in \mathrm{top}(k, X)$ violates its threshold t_i, and no $j \notin \mathrm{top}(k, X)$ violates its threshold t'_j, then there are no poorly-swapped pairs in $\mathcal{S}$ and thus no error.

We now partition the items into 3 disjoint sets according to their empirical probabilities; we will analyze the violation-probabilities in each set separately.

- G^1 is the top-k set, i.e. $G^1 \triangleq \{i \in \mathrm{top}(k,X)\}$
- $G^2 \triangleq \left\{j \notin \mathrm{top}(k,X) \mid \frac{p^*}{(1+\varepsilon)e^2} \le p_X(j) \le p^*\right\}$
- $G^3 \triangleq \left\{j \notin \mathrm{top}(k,X) \mid p_X(j) < \frac{p^*}{(1+\varepsilon)e^2}\right\}$

We denote by E^1 the event that one or more items in G^1 violate their threshold. E^2 and E^3 are defined similarly for G^2, G^3. Given the sample-size S, we bound the probabilities of these 3 events; for this we require the following two lemmas on the sample distribution.

Lemma B.1 *Let X be a given input and let i be some item from $\mathcal{I}$ having empirical probability $p_X(i)$. If $\mathcal{S}$ is a random sample of size S obtained by Algorithm A then*

$$\begin{aligned}\forall\varepsilon>0: \quad \Pr\{p_{\mathcal{S}}(i) \ge (1+\varepsilon)p_X(i)\} &\le \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}\right)^{Sp_X(i)}\\ \forall\varepsilon<0: \quad \Pr\{p_{\mathcal{S}}(i) \le (1+\varepsilon)p_X(i)\} &\le \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}\right)^{Sp_X(i)}\end{aligned}$$

Proof: The proof is by Chernoff bound. Since $p_{\mathcal{S}}(i) \triangleq \frac{w^{\mathcal{S}}(i)}{S}$, bounding the tail probability of $w^{\mathcal{S}}(i)$ is equivalent. Consider the set $\{\nu_1,\ldots,\nu_{w(i)}\}$ of all cells holding i in the input and let x_ℓ be an indicator random variable having value 1 if the cell ν_ℓ was sampled. Clearly x_ℓ are i.i.d random variables and the sampled-weight of i satisfies $w^{\mathcal{S}}(i) = \sum_{\ell=1}^{w(i)} \nu_\ell$. Since the expectation of the sampled-weight is $Sp_X(i)$, we get the result by Chernoff Bound. ▌

Lemma B.2 *For any input and for any $i,j \in \mathcal{I}$ such that $p_X(i) > p_X(j) > 0$, let $\alpha \triangleq \frac{p_X(i)}{p_X(j)}$ and define t_{ij} as*

$$t_{ij} \triangleq p_X(i)\frac{\alpha-1}{\alpha\ln\alpha} \qquad \textit{(we later show that } p_X(j) < t_{ij} < p_X(i)\textit{)}$$

If $\mathcal{S}$ is a random sample of size S obtained by Algorithm A for this input then:

$$\begin{aligned}\Pr\{p_{\mathcal{S}}(i) \le t_{ij}\} &\le \exp\left(-\frac{1}{8}S\cdot p_X(i)\cdot\Theta\left(\left(\frac{\alpha-1}{\alpha}\right)^2\right)\right)\\ \Pr\{p_{\mathcal{S}}(j) \ge t_{ij}\} &\le \exp\left(-\frac{1}{8}S\cdot p_X(i)\cdot\Theta\left(\left(\frac{\alpha-1}{\alpha}\right)^2\right)\right)\end{aligned}$$

Note also that t_{ij} is monotone decreasing with α and so are the 'violation' probabilities.

Proof: Let $f(\alpha) \triangleq \frac{\alpha-1}{\ln\alpha}$, $t \triangleq t_{ij} = f(\alpha)\cdot p_X(j)$, $e_1 \triangleq \{p_{\mathcal{S}}(i) \le t_{ij}\}$ and $e_2 \triangleq \{p_{\mathcal{S}}(j) \ge t_{ij}\}$. Applying the tail-probability bounds from Lemma B.1 for e_1 and e_2 gives

$$\Pr\{p_{\mathcal{S}}(i) \le t\} < (a_1)^{Sp_X(i)} \tag{5}$$

where

$$a_1 \triangleq \frac{e^{-\delta_1}}{(1-\delta_1)^{(1-\delta_1)}} \quad , \quad \delta_1 \triangleq \frac{p_X(i)-t}{p_X(i)} \tag{6}$$

and

$$\Pr\{p_{\mathcal{S}}(j) \ge t\} < (a_2)^{Sp_X(j)} \tag{7}$$

where

$$a_2 \triangleq \frac{e^{\delta_2}}{(1+\delta_2)^{(1+\delta_2)}} \quad , \quad \delta_2 \triangleq \frac{t - p_X(j)}{p_X(j)} \tag{8}$$

For $t = f(\alpha)p_X(j)$, we get from Eq. (6) and Eq. (8) that $\delta_1 = 1 - \frac{f}{\alpha}$ and $\delta_2 = f - 1$ giving

$$a_2 \ = \ a_1^{\alpha} \ = \ \frac{e^{(f-1)}}{(f)^{(f)}} \ = \ \exp\left(-(f \ln f + 1 - f)\right) \tag{9}$$

Using Taylor expansion around $\alpha = 1$ we can show that for $\alpha > 1$, $1 < f(\alpha) < 1 + \frac{1}{2}(\alpha - 1) < \alpha$ and therefore $p_X(j) < t < p_X(i)$. From Eq. (5), Eq. (7) and from Eq. (9), we get that $\Pr\{e_1\} < a_2^{Sp_X(j)}$ and $\Pr\{e_2\} < a_2^{Sp_X(j)}$ while

$$\begin{aligned} a_2^{Sp_X(j)} &= \exp\left(-Sp_X(j) \cdot (f \ln f + 1 - f)\right) \\ &= \exp\left(-S\frac{p_X(i)}{\alpha}(f \ln f + 1 - f)\right) \end{aligned} \tag{10}$$

The function $h(\alpha) = \frac{1}{\alpha}(f \ln f + 1 - f)$ is monotone increasing for any $\alpha > 1$. Let $\hat{h}(\alpha) \triangleq \frac{(\alpha-1)^2}{8\alpha^2}$, then $h(\alpha) = \Theta(\hat{h}(\alpha))$. The result of the lemma follows from Eq. (10) by replacing $h(\alpha)$ with $\Theta(\hat{h}(\alpha))$. ▮

Continuing with the proof of Lemma 3.1 we now define for each $i \in \text{top}(k, X)$ the exact choice of its threshold value as $t_i \triangleq p_X(i)\frac{\varepsilon}{(1+\varepsilon)\ln(1+\varepsilon)}$; this is equal to t_{ij} defined in Lemma B.2 for $\alpha = 1 + \varepsilon$. Given $i \in \text{top}(k, \mathcal{S})$, recall that for any j such that (i, j) is an important pair, we have $\frac{p_X(i)}{p_X(j)} > 1 + \varepsilon$. Therefore by Lemma B.2, $p_X(j) < t_i < p_X(i)$ and $\Pr\{E_i\} < \exp\left(-\frac{1}{8}Sp_X(i) \cdot \Theta\left(\left(\frac{\varepsilon}{1+\varepsilon}\right)^2\right)\right)$. Note also that for some $j \notin \text{top}(k, X)$, if we let i be the item that has $t'_j = t_i$ then $\Pr\left\{E'_j\right\} < \exp\left(-\frac{1}{8}Sp_X(i) \cdot \Theta\left(\left(\frac{\varepsilon}{1+\varepsilon}\right)^2\right)\right)$. Note that $p_X(i) \geq p^*$ for any $i \in \text{top}(k, X)$. Note also that the size of G^1 is k and the size of G^2 is bound by $(1 + \varepsilon)\, e^2 \,/\, p^*$. Clearly $p^* \leq \frac{1}{k}$ so using the union bound we obtain

$$\Pr\left\{E^1 \text{OR} E^2\right\} < \frac{10 \cdot (1+\varepsilon)}{p^*} \cdot \exp\left(-\frac{1}{8}Sp^*\,\Theta\left(\left(\frac{\varepsilon}{1+\varepsilon}\right)^2\right)\right) .$$

The following lemma bounds the probability of E^3.

Lemma B.3 *For E^3 as defined above, if the sample size satisfies $S \geq 2(1+\varepsilon)\ln(1+\varepsilon)/\varepsilon p^*$, then*

$$\Pr\left\{E^3\right\} < \frac{2e^2(1+\varepsilon)}{p^*} \cdot \exp\left(-\frac{1}{8}Sp^*\,\Omega\left(\left(\frac{\varepsilon}{1+\varepsilon}\right)^2\right)\right)$$

Proof: Let $f(\alpha)$ and $h(\alpha)$ be defined as in the proof of Lemma B.2,
and let: $f \triangleq f(1+\varepsilon) > 1$, $t \triangleq \frac{p^* f}{1+\varepsilon}$. For any item j in G^3, we have that $t'_j > t$. Therefore defining E''_j as the event of receiving $p_S(j) \geq t$, it is clear that $\Pr\left\{E'_j\right\} < \Pr\left\{E''_j\right\}$.

We now further partition G^3: for any integer $\ell \geq 0$ we define the subset G^3_ℓ as

$$G^3_\ell \triangleq \left\{j \notin \text{top}(k, X) : t \cdot e^{-3-\ell} \leq p_X(j) < t \cdot e^{-2-\ell}\right\}$$

For every item $j \in G_\ell^3$, let:

$$\begin{aligned} a &\triangleq \frac{t}{p_X(j)} \\ I &\triangleq \frac{a}{f(1+\varepsilon)} = \frac{p^*}{(1+\varepsilon)\cdot p_X(j)} \quad \Rightarrow \quad e^{2+\ell} < I \le e^{3+\ell} \end{aligned}$$

Using the tail bounds from Lemma B.1 we get

$$\begin{aligned} \Pr\left\{E_j''\right\} &< \left(\frac{e^{a-1}}{a^a}\right)^{Sp_X(j)} \\ &= \exp\left(-(a\ln a - a + 1)Sp_X(j)\right) \\ &= \exp\left(-Sp^*\frac{1}{(1+\varepsilon)I}\cdot(If(\ln f + \ln I) - If + 1)\right) \\ &= \exp\left(\frac{-Sp^*}{1+\varepsilon}\cdot\left((f\ln f - f + 1) \;+\; (f\ln I - \frac{I-1}{I})\right)\right) \\ &< \exp\left(\frac{-Sp^*}{1+\varepsilon}\cdot((1+\varepsilon)\,h(1+\varepsilon) \;+\; (f(\ell+2)-1))\right) \\ &< \exp\left(-Sp^*\,h(1+\varepsilon)\right)\cdot\exp\left(\frac{-Sp^*}{1+\varepsilon}\cdot f(\ell+1)\right) \end{aligned}$$

By definition of G_ℓ^3 we have that $p_X(j) > p^*/(1+\varepsilon)e^{\ell+3}$. We can therefore bound the size of G_ℓ^3 by $\left|G_\ell^3\right| < (1+\varepsilon)e^{\ell+3}/p^*$.

Let E_ℓ^3 denote the event that one or more items in G_ℓ^3 violate the threshold t, $E_\ell^3 \triangleq \bigcup_{j\in G_\ell^3} E_j''$. By the union-bound we get

$$\begin{aligned} \Pr\left\{E^3\right\} &\le \sum_{\ell=0}^{\infty}\Pr\left\{E_\ell^3\right\} \\ \Pr\left\{E_\ell^3\right\} &< \frac{e^{\ell+1}e^2(1+\varepsilon)}{p^*}\cdot\exp\left(\frac{-Sfp^*}{1+\varepsilon}(\ell+1)\right)\cdot\exp\left(-Sp^*\,h(1+\varepsilon)\right) \\ &= \frac{e^2(1+\varepsilon)}{p^*}\cdot\exp\left(-(\ell+1)(-1+\frac{Sfp^*}{1+\varepsilon})\right)\cdot\exp\left(-Sp^*\,h(1+\varepsilon)\right) \end{aligned}$$

Using $S \ge 2(1+\varepsilon)/fp^*$ we get that since $Sfp^*/(1+\varepsilon) > 2$,

$$\begin{aligned} \Pr\left\{E_\ell^3\right\} &< \frac{e^2(1+\varepsilon)}{p^*}\exp\left(-(\ell+1)\cdot\frac{Sfp^*}{2(1+\varepsilon)}\right)\cdot\exp\left(-Sp^*\,h(1+\varepsilon)\right) \\ &= \frac{e^2(1+\varepsilon)}{p^*}\exp\left(-Sp^*\,h(1+\varepsilon) - \frac{Sp^*\,f}{2(1+\varepsilon)}\right)\cdot\exp\left(-\ell\cdot\frac{Sfp^*}{2(1+\varepsilon)}\right) \end{aligned}$$

Note that the bounds we got for E_ℓ^3 form a descending geometric-series. By summing them up and using the fact that $Sfp^*/(1+\varepsilon) \ge 2$ again we get

$$\begin{aligned} \Pr\left\{E^3\right\} &< \frac{2e^2(1+\varepsilon)}{p^*}\cdot\exp\left(-Sp^*\left(h(1+\varepsilon)+\frac{f}{2(1+\varepsilon)}\right)\right) \\ &< \frac{2e^2(1+\varepsilon)}{p^*}\cdot\exp\left(-Sp^*\left(h(1+\varepsilon)+\frac{1}{2(1+\varepsilon)}\right)\right) \qquad \blacksquare \end{aligned}$$

To conclude the proof of Lemma 3.1, we combine the result of Lemma B.3 with the result we got for $\left\{E^1 \text{OR} E^2\right\}$. We get that for sample size satisfying $S \geq 2(1+\varepsilon)\ln(1+\varepsilon)/\varepsilon p^*$, the overall error probability satisfies

$$\Pr\left\{E^1 \text{OR} E^2 \text{OR} E^3\right\} < \frac{4e^2(1+\varepsilon)}{p^*} \cdot \exp\left(-Sp^*\, h(1+\varepsilon)\right)$$

Define $S^*(p^*,\varepsilon,\delta) \triangleq \dfrac{1}{p^*\, h(1+\varepsilon)} \cdot \left(\ln \dfrac{1+\varepsilon}{p^*\delta} + 4\right)$. For sample size at least $S^*(p^*,\varepsilon,\delta)$ we get an error probability smaller than δ. The result of the lemma is a simplified expression for $\varepsilon = O(1)$. ▮

Theorem 3.2 *Let $X \in \mathcal{X}(W,n,p^*)$ be an instance. Provided that p^*, ε, δ and W are known, Algorithm B outputs a top-k ε-approximation with probability at least $1-\delta$ and communication complexity of order $O\left(\frac{1}{p^*\,\varepsilon^2} \cdot \ln \frac{1}{p^*\delta} \cdot \log|\mathcal{I}|\right)$.*

Proof: First, note that the communication complexity of Algorithm B is exactly the sample size $|\mathcal{S}|$ times $O(\log|\mathcal{I}|)$ (the number of bits required to encode each item in the sample). Let S be a random variable whose value is the sample size. Clearly, $E[\,S\,] = W \cdot \mathtt{P}_{\mathtt{SAMPLE}}$, and hence, by Step 1 of the algorithm, we have $E[\,S\,] = 2S^*$. Now, the probability of error is bounded by $\Pr\{E_1 \text{OR} E_2\}$, where E_1 denotes the event where the output is incorrect even though the sample was large enough ($|\mathcal{S}| \geq S^*$), and E_2 denotes the event where the sample was too small, i.e. $|S| < S^*$. By Lemma 3.1, $\Pr\{E_1\} \leq \frac{\delta}{2}$. Using the Chernoff bound we obtain that $\Pr\{E_2\} < \exp\left(-\frac{1}{4}S^*\right)$ and since $S^* > 8\ln\frac{e^4}{\delta}$ we get $\Pr\{E_2\} < \frac{\delta}{e^4}$. Therefore, the algorithm is correct with probability at least $1 - \Pr\{E_1\} - \Pr\{E_2\} > 1 - 3\delta/4$. To bound the communication complexity, let E_3 denote the event that $|S| > 4S^*$. Using the Chernoff bound once again, we obtain that $\Pr\{E_3\} < \exp\left(-\frac{1}{3}S^*\right) < \frac{\delta}{e^4}$. Therefore, the sample size is bounded by $4S^* = O\left(\frac{1}{p^*} \cdot \frac{1}{\varepsilon^2} \cdot \ln\frac{1}{p^*\delta}\right)$ with probability at least $1 - \frac{\delta}{e^4}$ and all results hold with probability higher than $1-\delta$. ▮

C Proofs for Algorithm S

Theorem 3.3 *For any input $X \in \mathcal{X}(W,n,p^*)$, with probability at least $1-\delta$, Algorithm S outputs a top-k ε-approximation with communication complexity of order*

$$C_S = O\left(\frac{1}{p^*\,\varepsilon^2} \cdot \log\frac{1}{p^*\,\delta} \cdot \log|\mathcal{I}| \;+\; k\log\frac{1}{p^*}\log\frac{1}{\delta}\log\log W\right) .$$

Proof: The proof outline is as follows. The proof starts by showing that with high probability, the algorithm runs approximately $\log\frac{1}{p^*}$ iterations. As a result, the last iteration is similar to an execution of Algorithm B with the correct parameters and therefore outputs a top-k ε-approximation with high probability. The communication bound follows from the fact that the sample size is more than doubled on each iteration. The communication is therefore dominated by the last iteration.

The proof uses the following notational convention. Given a high probability estimate $\hat{Z}$ of Z that has accuracy $\varepsilon = 1$, Z_{lo} denotes $\hat{Z}/2$ and Z_{hi} denoted $2\hat{Z}$. Z_{lo} and Z_{hi} are uses as bounds on Z that hold with high probability. The proof also uses $\delta' \triangleq \delta/5$ to express error probabilities.

First we show why is Algorithm S correct. We look at two possible 'bad' events:

- Let E_1 be the event where Algorithm S stopped too early because of an erroneous bound, i.e. on some iteration we stopped even though we used $\hat{p} > p^*$.

- Let E_2 be the event where the last iteration used $\hat{p} \le p^*$ but the output of Algorithm B was not a top-k ε-approximation.

We can bound the probability that the output is not a top-k ε-approximation by $\Pr\{E_1\} + \Pr\{E_2\}$. Since we pass δ' as error probability to Algorithm B, from Section 3.1 it is easy to see that $\Pr\{E_2\} < \delta'$.

Before we bound $\Pr\{E_1\}$ we give the following lemma.

Lemma C.1 *With probability at least* $1 - 2\delta'$, *Algorithm* S *runs at least* $\log \frac{1}{p^*}$ *iterations before it stops.*

Proof: Let $L = \log \frac{1}{p^*}$ and let E_1' denote the event where the algorithm stopped during the first L iterations. We let $(I_0, I_1 \ldots)$ denote the first $L+1$ iterations in a reversed order. Specifically I_0 denotes the earliest iteration where $\hat{p} \le p^*$ and I_ℓ is the iteration which is ℓ iterations before that. On iteration I_ℓ, we have $\hat{p} > 2^{\ell-1} p^*$. Recall the stopping rule of Algorithm S is:

$$\text{‘Until } 2\hat{p} < \min\{w(i)_{lo}/W_{hi} \mid i \in T\}\text{’}.$$

Letting p^*_{lo} denote the current value of $\min\{w(i)_{lo}/W_{hi} \mid i \in T\}$ we obtain

$$\Pr\{E_1'\} \le \sum_{\ell=1}^{L} \Pr\left\{p^*_{lo} > 2^\ell\, p^*\right\} .$$

We now consider the iteration I_ℓ. Let j be the item having the smallest weight, $w(j)$, among the k items in T and define $p(j)_{lo} \triangleq w(j)_{lo}/W_{hi}$. We have

$$\begin{aligned} p(j) &\le p^* \\ p(j)_{lo} &\ge p^*_{lo} \\ \Pr\left\{p^*_{lo} > 2^\ell \cdot p^*\right\} &\ge \Pr\left\{p(j)_{lo} > 2^\ell \cdot p(j)\right\} . \end{aligned}$$

Since $w(j)_{lo}$ was obtained using Algorithm BoundCount with $\varepsilon = 1$ and error probability δ', the algorithm used $M = 6\ln(1/\delta')$ and by Corollary 2.2,

$$\Pr\left\{p(j)_{lo} > 2^\ell \cdot p(j)\right\} < \exp\left(-(2^\ell - 1)^2 \ln \delta'\right) = \delta'^{(2^\ell - 1)} .$$

For $\delta' < \frac{1}{2}$ and we get $\Pr\{E_1'\} < \sum_{\ell=1}^{L} \delta'^{(2^\ell-1)} < 2\delta'$. ∎

Since after $\log \frac{1}{p^*}$ iterations, $\hat{p}$ is smaller than p^*, a direct result of Lemma C.1 is that $\Pr\{E_1\} < 2\delta'$ and the total error probability of the algorithm is bound by $3\delta'$.

In order to bound communication complexity we give another lemma about the number of iterations.

Lemma C.2 *With probability at least* $1 - 2\delta'$, *Algorithm* S *runs at most* $\log \frac{1}{p^*} + 5$ *iterations before it stops.*

Proof: Let $L' \triangleq \log \frac{1}{p^*} + 5$ and let E_3 denote the event where Algorithm S reached iteration $L' + 1$. We bound the probability of E_3. We assume that Algorithm S did reach iteration L' and consider the state at the end of this iteration. Recall that proceeding to the next iteration occurs if $2\hat{p} < p^*_{lo}$, where p^*_{lo} is the minimum of k high-probability bounds for the empirical probability of k distinct items. Let j be the item having the lowest empirical probability among the k items inspected on iteration L', then if Algorithm S continued to the next iteration, then $\{2\hat{p} > p(j)_{lo}\}$, and therefore either of the two following events occurred on iteration L':

- Event $E_3' : p(j) < \frac{1}{2}p^*$
- Event $E_3'' : p(j) \geq \frac{1}{2}p^*$ AND $2\hat{p} > p(j)_{lo}$.

Consider the probabilities of E_3' and E_3'' on iteration L'. Note that E_3' is the event where the top-k of the sample is not a $1 - approximation$ of the true top-k set. Since we assume $\varepsilon \leq 1$ and since $\mathtt{P_{SAMPLE}}$ is large enough at this stage, $\Pr\{E_3'\} \leq \delta'$ by Lemma 3.1. E_3'' is the event where $p(j)$ is high enough but the high-probability bound is too low. Since on iteration L', we have $2\hat{p} < \frac{p^*}{16}$, E_3'' requires that $\left\{\frac{1}{8}p(j) > p(j)_{lo}\right\}$. Recall that $p(j)_{lo} \triangleq \frac{\hat{w}(j)}{2W_{hi}}$ where $\hat{w}(j)$ was obtained by Algorithm BoundCount using $\varepsilon = 1$. By Corollary 2.2 the probability of the last event is less than δ' and therefore $\Pr\{E_3''\} < \delta'$. Since E_3 requires that either E_3' or E_3'' occur on iteration L' we have that $\Pr\{E_3\} < 2\delta'$ ▮

We now have that with probability at least $1 - 4\delta'$ the output is correct and Algorithm S ran between $\log \frac{1}{p^*}$ and $\log \frac{1}{p^*} + 5$ iterations. The rest of the proof deals with communication complexity assuming Algorithm S performed at most L' iterations.

We use Chernoff bound for the number of cell probes. Let $S(\ell)$ denote the sample size used on iteration ℓ and let $\text{CP} = \sum S(\ell)$ denote the number of cell-probes throughout Algorithm S. We consider a set of all indicator random variables used for sampling by all iterations (one random variable per input cell per iteration); CP is the sum of these indicators. To estimate the expectation we use the fact that $\mathtt{P_{SAMPLE}}$ is at least doubled on each iteration and so is the expected sample size. We can therefore bound the sum of expectations by by twice the expected sample size on the last iteration.

$$E[\,\text{CP}\,] = \sum E[\,S(\ell)\,] < 2S(L') = \Theta\left(S^*(W, n, p^*)\right)$$

Using Chernoff bound we have $\Pr\{\text{CP} > 2E[\,\text{CP}\,]\} < e^{-\frac{1}{3}S^*}$. Since S^* is defined as $S^*(p^*, \varepsilon, \delta) \triangleq \dfrac{g(\varepsilon)}{p^*} \cdot \left(\ln \dfrac{1+\varepsilon}{p^*\delta} + 4\right)$ and since for $\varepsilon \leq 1$, $g(\varepsilon) > 8$, we get

$$S^* > 8 \ln \frac{e^4}{\delta'} \quad \text{yielding} \quad \Pr\{\text{CP} > 2E[\,\text{CP}\,]\} < \delta' \,.$$

We have that with probability at least $1 - 5\delta'$ the cell probe complexity is of order $\Theta\left(S^*(W, n, p^*)\right)$ and the resulting number of communicated bits can be bound by $O\left(S^*(W, n, p^*) \cdot \log|\mathcal{I}|\right)$

Finally we bound the communication complexity needed to obtain all high-probability bounds on each iteration by $\Theta\left(k \log|\mathcal{I}| + k \log \frac{1}{\delta'} \log\log W\right)$. If we bound k by $\frac{1}{p^*}$ and bound the number of iterations by L' we get that the communication complexity of all obtained bounds on all iterations is of order

$$\Theta\left(\frac{1}{p^*} \log \frac{1}{p^*} \log|\mathcal{I}| + k \log \frac{1}{p^*} \log \frac{1}{\delta'} \log\log W\right) \,.$$

Adding the two communication complexities and setting $\delta' \triangleq \delta/5$ we get that all of the above holds with probability at least $1 - \delta$ giving the result of the theorem. ▮