Approximate Top-k Queries in Sensor Networks

A thesis submitted toward the degree of Master of Science in Electrical and Electronic Engineering

by

Allon Shafrir

March, 2006
Approximate Top-k Queries in Sensor Networks

A thesis submitted toward the degree of Master of Science in Electrical and Electronic Engineering

by

Allon Shafrir

This research work was carried out at the Department of Electrical Engineering - Systems, under the supervision of Dr. Boaz Patt-Shamir

March, 2006
Acknowledgments

I would like to extend my gratitude to my supervisor, Dr. Boaz Patt-Shamir, for his guidance, ideas and help that made this work possible. Boaz has been an outstanding guide throughout the research and set an admirable example in his ambitions for accuracy and clarity.

I would also like to thank my wife Sharon for her patience and support over the evenings she had to spend without me.

This research was supported in part by the Israel Ministry of Science and Technology.
Abstract

We consider a distributed system where each node keeps a local count for items (similar to elections where nodes are ballots 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. This thesis presents 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.
# Contents

1 Introduction ................................................. 1
   1.1 Outline ............................................... 2

2 Background .................................................. 4
   2.1 Sensor Networks ....................................... 4
   2.2 The top-$k$ problem ................................... 6

3 Model and Problem Definition ............................. 9
   3.1 Model .................................................. 9
   3.2 Problem definition .................................... 10
   3.3 Complexity measures .................................. 11

4 Preliminaries ................................................. 13

5 Algorithms .................................................... 16
   5.1 Basic sampling ........................................ 17
   5.2 Algorithm $S$: adaptive sample size .................. 20
   5.3 Algorithm $R$: sample & remove ....................... 25

6 Simulations results .......................................... 37
6.1 Comparison with existing algorithms . . . . . . . . . . . . . . . . . . 38
6.2 The effect of accuracy and critical frequency on performance . . . 39

7 A lower bound on the total cell-probe complexity 41

8 Conclusions 52

A Additional Proofs 57

B Random sampling 63

C List of notation 66
List of Figures

5.1 Algorithm $S$ .............................................. 21
5.2 Algorithm $R$ .............................................. 36
6.1 Communication costs of algorithms $R$, TA, and TPUT .............. 38
6.2 Communication costs of optimized algorithms ........................... 39
6.3 Communication costs of algorithms $S$ and $R$ ........................... 39
7.1 Transition probabilities during the sampling process. ................. 46
Chapter 1

Introduction

Possibly one of the clearest examples of the difference between “global” and “local” can be seen in elections: each ballot has a local score for each candidate, but the result we care about is the global scores, i.e., how many votes does each candidate have overall. A top-$k$ query in this case is “Which are the $k$ globally most popular candidates?” 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 main difficulty is that the scores may be divided arbitrarily among the different locations. In elections, for example, it may be that the most popular candidate globally is the least popular in each ballot. 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 [12] 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, 18, 27, 8, 22].

In this work, we propose a simple but effective approach for top-\(k\) computation in large systems. Our algorithm is Monte Carlo (i.e., it is randomized and it may err with some arbitrarily small probability), and its results are only approximate: using very little communication, the algorithm can tell which items are in the top-\(k\) set, roughly.

We study the individual communication complexity of our algorithm and give some evidence that may indicate that our algorithm is close to optimal. In particular, we study a limited class of single-round Monte Carlo algorithms for the top-\(k\) problem, which may be appropriate for the “smart dust” type of sensor networks [25]. We prove a lower bound on the cell-probe complexity [26, 14] of algorithms in this model, which is very close to our upper bound in this class.

1.1 Outline

The work is organized as follows.

Chapter 2 provides some intuitive background on Sensor Networks and on the top-\(k\) problem. We describe previous work on these subjects and explain the motivation for our study.

Chapter 3 presents our model and notations. Based on the formal definition of the system and input model, it also defines the top-\(k\) problem and the complexity measures we use to evaluate algorithms.

In Chapter 4 we presents a few known techniques for counting and summation in large systems. We derive an efficient Monte Carlo algorithm for obtaining bounds in our model and give related results.

In Chapter 5 we describe our algorithm. The algorithm is described in a bottom-up fashion where analysis of simple algorithms provides us with reasoning
for construction of better algorithms. This chapter also presents analysis results for our final algorithm.

Chapter 6 describes simulations we ran to evaluate the actual communication used by our algorithm. We ran two types of simulations. The first type simulates our algorithm with various inputs trying to understand the effectiveness of used techniques. The second type simply presents the performance of our algorithm compared with existing algorithms for typical inputs; results demonstrate the improvement our algorithm can offer in large systems.

Chapter 7 presents an analysis of Single-Round Monte Carlo algorithms for the top-\(k\) problem. The analysis yields a lower bound on the cell probe complexity of such algorithms. This lower bound may indicate that our algorithm is close to optimal for the ‘smart dust’ type of sensor networks.

Finally, Chapter 8 summarizes the results obtained in this study and suggests possible future research.
Chapter 2

Background

2.1 Sensor Networks

By sensor we refer to a tiny autonomous computer with sensing and communication capabilities. A Sensor Network consists of many cheap sensors which are densely deployed in some environment. Each sensor in the network collects measurements from its close environment and communicates with some of his peers. The study of sensor networks is primarily driven by the Smart Dust vision, where sensors are almost as small and as numerous as dust particles. Naturally, sensor networks enable large-scale measurements in environments where this task is otherwise impossible. Applications that can benefit from these capabilities include:

• Collecting military intelligence.

• Monitoring of phenomena for environmental purposes such as:
  • Detecting forest fires or water contamination in an early stage.
  • Studying wildlife and seismic activities.
  • Precision agriculture.

• Health monitoring.

• Home and Office automation.
When deploying a sensor network, an obvious goal is to keep the network operational for as long as possible. On the other hand, to approach the smart dust vision one needs to develop autonomous sensors that are very small. In contemporary sensors, battery size is a major factor and minimization requires reduced power consumption of each sensor. It so happens that communication is by far the largest energy consumer in contemporary sensors; sending or even receiving one bit is equivalent to approximately 1,000 cpu-operations in terms of energy cost. It is therefore critical to reduce the amount of communication done by each sensor. The term individual communication refers to this critical quantity; it signifies the maximum number of bits sent or received by any single node.

2.1.1 Sensor nets as distributed databases

Essentially, a sensor network is a distributed system collecting data. It is distinguished from other systems by two properties:

1. The number of nodes, denoted by \( n \), is very large.

2. The high importance of individual communication.

The purpose of the system is to answer queries based on collected data. A typical query includes some filtering of relevant data and the calculation of an aggregate (e.g. sum, max, median, average). The input for such queries can be modeled as pairs of the form (item, score) where each node may hold data about many items and the scores of one item may be spread over many nodes. A naive approach for processing such queries would be to send all collected data to a central node and process the query locally at the center. With this approach, the amount of communication performed by the central node (or its close neighbors) would be proportional to the total amount of data; the central node is likely to run out of battery before a handful of queries are answered. Filtering is actually very easy to do with minimal communication; if we broadcast the criterion to all nodes,
each node can filter his local data accordingly. Efficient processing of aggregates on the other hand, is not as trivial.

Efficient aggregation is a central issue in the study of sensor networks. A typical assumption (see, e.g., [18, 27]) is that data is routed on a spanning tree, and each node does some aggregation en-route. Under this assumption, Simple aggregates (such as counting the number of items, summing numbers etc.) can be done with $O(\log W)$ communication per node where $W$ is the total input size.

Considine et al. [8] relax the spanning tree assumption to allow messages to be duplicated for these aggregates. In [20], 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.

Efficient computation of 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 [22, 15]; the randomized algorithm of [22] 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 [24, 4], monitoring the median and heavy-hitters is studied in [9] and monitoring the top-$k$ items is studied in [3].

This work presents a method for efficient aggregation for the top-$k$ problem.

## 2.2 The top-$k$ problem

Whenever we have a data collection consisting of many (item, score) pairs, a natural derivative is calculating the global score of an item (being the sum of all its scores). A top-$k$ query requests the retrieval of the $k$ items whose global score
is maximal.

Following are some examples for the top-$k$ task.

- 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.

- Distributed-denial-of-service (DDoS) attacks are a major concern on the Internet. Contemporary detection methods of such attacks use many agents, each looking for abnormal patterns in its local traffic. Finding the globally-largest-anomalies efficiently is important in such systems.

As already suggested, the main difficulty in calculating top-$k$ is the possible discrepancy between the local scores in different nodes. The performance of deterministic algorithms is 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.

Probably the most celebrated result for the database version of the top-$k$ problem is the Threshold Algorithm (TA) by Fagin, Lotem and Naor [13]. In that paper, they define a notion of ‘instance-optimality,’ a notion reminiscent of competitiveness for on-line algorithms, and prove that TA is ‘instance-optimal’ for the top-$k$ problem when compared with any algorithm that never errs (i.e., deterministic or Las Vegas algorithms). In [13] they also define the approximation metric we use here, and present a version of TA which is instance-optimal among top-$k$ approximation algorithms that never err. However, TA may do as many as $n$ accesses times the optimum, where $n$ is the number of nodes in the system (they also show that an $\Omega(n)$ factor blowup is unavoidable for any algorithm which doesn’t
err). In [13], $n$ is considered to be a constant, but obviously, when the database is distributed and the critical cost measure is communication rather than accesses, TA may not be the ultimate solution.

This viewpoint is developed by Cao and Wang in [7], where the TPUT algorithm is proposed as a heuristic which saves communication in some typical cases. In particular, it is proved 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 depends crucially on the way the scores are partitioned among nodes.

Other related works include variations of TA and TPUT optimized for certain network models [19, 28, 5].

Having described the context of this work we proceed with some formal definitions in the next chapter.
Chapter 3

Model and Problem Definition

This chapter includes most of the definitions we use. It formally defines our system model, the problem we deal with and the measures we use to assess algorithms.

3.1 Model

3.1.1 System model

The system is modeled as a communication graph $G(V, E)$ with $n \overset{\text{def}}{=} |V|$, 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 defined to be 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., [2, 17]). 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 that includes routines 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 $\Theta(|M|)$ 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 $\Theta(|M|)$ 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, 18, 27, 8, 22]).

### 3.1.2 Input model

We consider collected data of the form (item, local-weight) and choose to represent weights by multiplicities. Let $\mathcal{I}$ denote the set of possible items and 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 $i$ has $i$ 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 $W_v \defeq |X_v|$. The weight of an item $i \in \mathcal{I}$, is the sum of its multiplicities over all nodes, i.e., $w(i) \defeq \sum_{v \in V} w_v(i)$. We define the total input size as the total number of input-cells, i.e., $W(X) \defeq \sum_{v \in V} |X_v|$.

### 3.2 Problem definition

The empirical probability, or frequency, of an item $i \in \mathcal{I}$ in $X$, is defined by $p_X(i) \defeq \frac{w(i)}{W}$. When the context is clear we omit the subscript.
Using this notation, we define the top-$k$ set of $X$ as follows.

**Definition 3.1** Let $k$ be a natural number, and suppose that $|\mathcal{I}| \geq k$. The top-$k$ set of $X$, denoted $\text{top}(k, X)$, is a subset of $\mathcal{I}$ of size $k$ containing the items with the maximal weight. Formally, $\text{top}(k, X)$ satisfies the following conditions.

1. $\text{top}(k, X) \subseteq \mathcal{I}$,
2. $|\text{top}(k, X)| = k$, and
3. $\forall i \in \text{top}(k, X), \forall j \in \mathcal{I} \setminus \text{top}(k, X) : p_X(i) \geq p_X(j)$.

Following [13], we extend Definition 3.1 to the concept of approximate top-$k$ sets.

**Definition 3.2** Let $\varepsilon \geq 0$. An $\varepsilon$-approximation of the top-$k$ set of $X$ is a set $\text{top}_\varepsilon$ of $k$ items, such that for all items $i \in \text{top}_\varepsilon$ and $j \notin \text{top}_\varepsilon$, we have $p_X(j) \leq (1 + \varepsilon)p_X(i)$.

We will mainly be interested in small values of $\varepsilon$ 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 $p^*_X$, is the empirical probability of the least popular item in $\text{top}(k, X)$, i.e., given input $X$ and a natural number $k$, we define

$$p^*(X, k) \overset{\text{def}}{=} \min \{ p_X(i) \mid i \in \text{top}(k, X) \}.$$ 

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 $\mathcal{X}(W, n, p^*)$.

### 3.3 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., $c_A(X) \overset{\text{def}}{=} \max \{c_A(X, v) \mid v \in V\}$. 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., $C_A(\mathcal{X}) \overset{\text{def}}{=} \max \{c_A(X) \mid X \in \mathcal{X}\}$.

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 many cases, it is more convenient to analyze the cell probe complexity of algorithms [26, 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, $c_{pA}(X, v)$ denotes the number of input cells accessed by node $v$, throughout the execution of $A$ for input $X$; $c_{pA}(X)$ denotes the maximal-node cell-probes of algorithm $A$ for input $X$, i.e., $c_{pA}(X) \overset{\text{def}}{=} \max \{c_{pA}(X, v) \mid v \in V\}$. Finally, $CP_A(\mathcal{X})$ denotes the cell probe complexity of an algorithm $A$ over a set of possible inputs $\mathcal{X}$, namely $CP_A(\mathcal{X}) \overset{\text{def}}{=} \max \{c_{pA}(X) \mid X \in \mathcal{X}\}$. 

12
Chapter 4

Preliminaries

This section presents a few simple techniques for counting and related results which we use in our work.

Computing the sum, minimum and maximum of values across nodes are possibly the simplest tasks in our model. The broadcast-convergecast protocol [23] with the natural corresponding aggregation rules gives the following trivial result. For each node $v$, let $z_v$ be a local value held by $v$ (in our applications, $z_v$ will be the local weight of some item).

**Fact 4.1** Let $Z_{\text{sum}} \overset{\text{def}}{=} \sum_{v \in V} z_v$. There exist an algorithm that computes $Z_{\text{sum}}$ with individual communication complexity $O(\log Z_{\text{sum}})$. Similarly, let $Z_{\text{max}} \overset{\text{def}}{=} \max \{z_v \mid v \in V\}$, and $Z_{\text{min}} \overset{\text{def}}{=} \min \{z_v \mid v \in V\}$. Then there exist similar algorithms that compute $Z_{\text{max}}$ or $Z_{\text{min}}$ with communication complexity $O(\log Z_{\text{max}})$.

In simple terms, this means that the input size, $W$, in our model can be calculated with individual communication complexity $O(\log W)$. By first broadcasting a filtering criterion, the weight, $w(i)$, of a given item can be calculated with individual communication complexity $O(\log |I| + \log w(i))$. A much more interesting fact is that summation can be approximated with exponentially smaller communication complexity. Intuitively, the basis for the best approximate counting algorithm is the following phenomenon: the maximum of $w$ geometrically distributed variables with parameter $1/2$ is, with constant probability, $\log_2 w$ (see
[16, 1, 11] for in-depth study). So suppose we wish to compute the total weight of an item \(i\). If for each input-cell holding item \(i\) we sample an independent geometric random variable with parameter \(1/2\) (say, by counting random bits until the first “1” occurs), then, with constant probability, the maximum of all these samples is about \(\log_2 w\). The nice thing is that the number of bits required to encode \(\log_2 w\) is \(\log_2 \log_2 w\). Therefore, by computing the maximum of these samples (each sample is \(O(\log \log w)\) bits long), we get a rough estimate of the total weight while incurring only \(O(\log \log w)\) communication complexity. This estimate can be refined by repeating the experiments and averaging. By a result of Durand and Flajolet [11], the input size, \(W\), can be estimated up to a constant relative deviation (e.g. \(±10\%\)) with individual communication complexity \(O(\log \log W)\). To state this result formally, we use the following concept.

**Definition 4.1** 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 \((\varepsilon, \sigma^2)\)-estimate of \(Z\) if

\[
\frac{1}{Z} |E[\hat{Z}] - Z| \leq \varepsilon, \text{ and } \frac{1}{Z^2} \text{Var}[\hat{Z}] \leq \sigma^2.
\]

Specializing the result of [11] to our needs, it can be stated as follows.

**Fact 4.2 ([11])** There exists a randomized algorithm \(A_{\log \log}\) with communication complexity \(O(\log \log W)\), whose output is an \((\varepsilon, \sigma^2)\)-estimate of the input size, \(W\), for \(\varepsilon = 10^{-6}\) and \(\sigma = 1\).

To get bounds that hold with high probability, we use Bernstein’s Inequality (see, e.g., [10]). Bernstein’s Inequality says that for any natural number \(M\), with probability \(1 - e^{-\Omega(M)}\), the difference between \(W\) and the average of \(M\) independent values of \(\hat{W}\) is at most \(1/\sqrt{M}\). Hence we define the following algorithm.
Algorithm BoundCount
Input: $\varepsilon, \delta, i$
1. $M \leftarrow \left(\frac{6}{\varepsilon^2}\right) \ln \left(\frac{1}{\delta}\right)$.
2. Broadcast the filtering criterion $i$, letting each node filter its local input.
3. for $\ell = 1$ to $M$, run $A_{\log\log}$ obtaining an independent estimate $\hat{w}_\ell$ of $w(i)$.
4. Output $\hat{w} \overset{\text{def}}{=} \frac{1}{M} \sum_{\ell=1}^{M} \hat{w}_\ell$

**Corollary 4.3 (high-probability estimates)** For given $10^{-5} \leq \varepsilon \leq 1$ and $\delta > 0$, Algorithm BoundCount outputs an estimate $\hat{w}$ of $w(i)$ such that

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

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 $\zeta > 10^{-5}$,

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

**Proof:** For each $\hat{w}_\ell$, consider the random variable $z_\ell \overset{\text{def}}{=} \hat{w}_\ell / w(i)$. $z_\ell$ is a (useless) estimate of $z \overset{\text{def}}{=} E [\hat{w}_\ell] / w(i)$ and by Fact 4.2, it satisfies $|E [z_\ell] - 1| < 10^{-6}$, and $\text{Var} (z_\ell) = O(1)$. Now consider the random variable $\hat{z} \overset{\text{def}}{=} \hat{w} / w(i)$ 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 $\Pr \{|\hat{z} - 1| < 10^{-6} + \varepsilon/2\} > 1 - \exp \left( -M \varepsilon^2 / 6 \right)$. This translates directly to the average of the $M$ original estimates, and the result of the corollary follows. \hfill \qed
Chapter 5

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 weights into multiple nodes is the possible discrepancy between the local weights 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 weights 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) 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 proportional to $O(1/p^*)$ (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 $\varepsilon$, and the confidence parameter $\delta$. 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, i.e., the frequency of the \( \ell \)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, and take another sample of the same size. It is straightforward to see that for the geometric instance described above, such a recursive algorithm will have 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 Section 5.1, describing basic algorithms we later use as a building block. Section 5.2 presents Algorithm \( S \) which uses adaptive sample size. Section 5.3 presents Algorithm \( R \), which is our main result.

### 5.1 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 our sampling routine and analyze its properties.

```
Algorithm A

Input: P

(1) The root node broadcasts P to all other nodes.
(2) Each node performs a random experiment for each local-cell such that the cell is
sent to the root with probability P.
(3) The root collects all received cells into a multiset S which is the returned sample.
```

We use $S$ denotes the obtained sample size, and consider the sampled-weight
of an item, $w^S(i)$, which is the multiplicity of $i$ within the sample $S$. The sampled-frequency of $i$ is denoted by $p_S(i) \overset{\text{def}}{=} w^S(i)/S$. Assume we wish to take a single
sample and output the top-$k$ set by the sample, i.e. $\text{top}(k, S)$. The first question
to be answered is what should be the sample size. By using Chernoff Bound, we
can show that with high probability, the sampled-frequency of an item $i$ is within
factor $1 + \varepsilon$ from its global-frequency. The probability that this does not happen is
proportional to $\exp(-S p_X(i) \varepsilon^2)$. By considering $|\mathcal{I}|$ possible items we get that
a sample of size $O\left(\frac{1}{p^* \varepsilon^2} \cdot \log \frac{|\mathcal{I}|}{\delta}\right)$ is sufficient to provide a top-$k$ $\varepsilon$-approximation
with probability $1 - \delta$ ($p^*$ is the critical frequency of the top-$k$ set). The following
lemma gives a tighter bound where $\log \frac{|\mathcal{I}|}{\delta}$ is replaced by $\log \frac{1}{p^* \delta}$. The additional
complexity of our proof is justified for common applications where a constant
approximation level and constant error probability are used but the number of
items ($|\mathcal{I}|$) is proportional to the system size.

**Lemma 5.1** 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 a top-$k$ $\varepsilon$-approximation of $X$ 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 A.

**Proof outline:** To bound the error-probability of a given sample size, we iden-
tify pairs of items $i, j$ such that $i$ must be more popular than $j$ in the sample to
get a top-$k$ $\varepsilon$-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 $\delta$, we get our bound for the required sample size. We comment that while the proof assumes that the sample was obtained by Algorithm $A$, the same result holds for any algorithm which performs uniform sampling (with replacement or without replacement). Appendix $B$ elaborates on this matter and extends the proof of Lemma 5.1 accordingly.

Algorithm $B$, presented below, follows directly from the Lemma 5.1. The algorithm first determines the sampling probability $P_{\text{SAMPLE}}$ using the function $S^*$. Each cell (i.e., unit of weight) is then sent to the root with probability $P_{\text{SAMPLE}}$, 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 $P_{\text{SAMPLE}} \leftarrow 2 \cdot S^*(p^*, \varepsilon, \delta)/W$.
2. Execute Algorithm $A$ with parameter $P_{\text{SAMPLE}}$ to get a sample $S$.
3. Output $\text{top}(k, S)$.

When running Algorithm $B$ as described above, each sampled vote is sent to the root separately incurring communication $\log |I|$. 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.

**Theorem 5.2** Let $X \in X(W, n, p^*)$ be an instance. Provided that $p^*, \varepsilon, \delta$ and $W$ are known, Algorithm $B$ outputs a top-$k$ $\varepsilon$-approximation with probability at least
Note the $1/p^*$ factor in the result of Theorem 5.2: 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 7.

**Proof of Theorem 5.2:** First, note that the communication complexity of Algorithm $B$ is exactly the sample size $|S|$ times $O(\log |I|)$ (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, $E[S] = W \cdot p_{\text{SAMPLE}}$, and hence, by Step 1 of the algorithm, we have $E[S] = 2S^*$. Now, the probability of error is bounded by $\Pr\{E_1 \cup E_2\}$, where $E_1$ denotes the event where the output is incorrect even though the sample was large enough ($|S| \geq S^*$), and $E_2$ denotes the event where the sample was too small, i.e. $|S| < S^*$. By Lemma 5.1, $\Pr\{E_1\} \leq \frac{\delta}{2}$. 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 $4S^* = O\left(\frac{1}{p^*} \cdot \frac{1}{e^2} \cdot \ln \frac{1}{p^*} \cdot \log |I|\right)$ with probability at least $1 - \frac{\delta}{e^4}$ and all results hold with probability higher than $1 - \delta$.

5.2 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
Algorithm $S$

Input: $k, \varepsilon, \delta$

(1) $\hat{W} \leftarrow \text{BoundCount}(\varepsilon = 1, \delta/5, \text{‘ALL’})$

(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 $i \in T$,

\[ \hat{w}(i) \leftarrow \text{BoundCount}(\varepsilon = 1, \delta/5, i) \]

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 5.1: Algorithm $S$

bound on $p^*$: the least popular among any $k$ items is certainly no more popular than the least popular among the top-$k$ items. Secondly we note that exact counting is not necessary: we can use high-probability-estimates as described in Corollary 4.3 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 5.2 by an arbitrary factor.

Our solution, in Algorithm $S$ below, combines the idea 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 5.1.

**Theorem 5.3** For any input $X \in \mathcal{X}(W, n, p^*)$, with probability at least $1 - \delta$, Algorithm $S$ outputs a top-$k$ $\varepsilon$-approximation with communication complexity

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

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

**Proof of Theorem 5.3:** 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-$k\varepsilon$-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 used as bounds on $Z$ that hold with high probability. The proof also uses $\delta' \overset{\text{def}}{=} \delta/5$ 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 $\hat{p} \leq p^*$ but the output of Algorithm $B$ was not a top-$k\varepsilon$-approximation.

We can bound the probability that the output is not a top-$k\varepsilon$-approximation by $\Pr\{E_1\} + \Pr\{E_2\}$. Since we pass $\delta'$ as error probability to Algorithm $B$, from Section 5.1 it is easy to see that $\Pr\{E_2\} < \delta'$.

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

**Lemma 5.4** 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 \( \hat{p} \leq p^* \) and \( I_{\ell} \) is the iteration which is \( \ell \) iterations before that. On iteration \( I_{\ell} \), 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

\[
\Pr \{ E'_1 \} \leq \sum_{\ell=1}^{L} \Pr \{ p^*_{lo} > 2^\ell p^* \}. 
\]

We now consider the iteration \( I_{\ell} \). Let \( j \) be the item having the smallest weight, \( w(j) \), among the \( k \) items in \( T \) and define \( p(j)_{lo} \overset{\text{def}}{=} w(j)_{lo}/W_{hi} \). We have

\[
p(j) \leq p^* \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad p(j)_{lo} \geq p^*_{lo} 
\]

\[
\Pr \{ p^*_{lo} > 2^\ell \cdot p^* \} \geq \Pr \{ p(j)_{lo} > 2^\ell \cdot p(j) \}. 
\]

Since \( w(j)_{lo} \) was obtained using Algorithm BoundCount with \( \varepsilon = 1 \) and error probability \( \delta' \), the algorithm used \( M = 6 \ln(1/\delta') \) and by Corollary 4.3,

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

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 5.4 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 5.5** With probability at least \( 1 - 2\delta' \), Algorithm \( S \) runs at most \( \log \frac{1}{p^*} + 5 \) iterations before it stops.
**Proof:** Let $L' \overset{\text{def}}{=} \log \frac{1}{p^*} + 5$ 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 $2\hat{p} < p^*_l$, where $p^*_l$ 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 $\{2\hat{p} > p(j)_l\}$, 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 $2\hat{p} > p(j)_l$.

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 $1 - \epsilon$ approximation of the true top-$k$ set. Since we assume $\epsilon \leq 1$ and since $p_{\text{SAMPLE}}$ is large enough at this stage, $\Pr\{E'_3\} \leq \delta'$ by Lemma 5.1. $E''_3$ 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 $\left\{\frac{1}{8}p(j) > p(j)_l\right\}$. Recall that $p(j)_l \overset{\text{def}}{=} \frac{\hat{w}(j)}{2^{\lceil \text{hi} \rceil}}$ where $\hat{w}(j)$ was obtained by Algorithm BoundCount using $\epsilon = 1$. By Corollary 4.3 the probability of the last event is less than $\delta'$ and therefore $\Pr\{E''_3\} < \delta'$. Since $E_3$ requires that either $E'_3$ or $E''_3$ occur on iteration $L'$ we have that $\Pr\{E_3\} < 2\delta'$. $\Box$

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 $\ell$ and let $CP = \sum S(\ell)$ 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 \( p_{\text{sample}} \) 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.

\[
E[CP] = \sum E[S(\ell)] < 2S(L') = \Theta(S^*(W, n, p^*))
\]

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

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

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

Finally we bound the communication complexity needed to obtain all high-probability bounds on each iteration by \( \Theta(k \log |\mathcal{I}| + k \log \frac{1}{\delta} \log \log W) \). 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 \( \delta' \defeq \delta/5 \) we get that all of the above holds with probability at least \( 1 - \delta \) giving the result of the theorem.

5.3 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 frequency of the \( k^{\text{th}} \) most popular item). However, when the frequency 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 on 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 5.2). On each iteration, the algorithm samples with probability $p_{\text{SAMPLE}}$, obtained from the current estimate $\hat{p}$ of $p^*$. After sampling, the algorithm counts (approximately) the top items in the sample (Step 3f), the number of items is such that the cost of counting is about the same as the cost of sampling. The counted items (recorded in $Q$) are not considered part of the input anymore. Since the input size shrinks, $\hat{p}$ need not necessarily be cut by a factor of 2: say, if the input size was reduced by a factor of 2 then $\hat{p}$ can remain the same. More generally, $\hat{p}$ is adjusted so that its value in the full input is halved (Step 3c).

The loop is executed until one of the stopping rules specified in Predicate $\text{safe}$ is met. These rules use a lower bound on $p^*$ computed by function $p_{\text{lo}}$: the bound is a high-probability lower bound on the $k^{th}$ highest value among all values counted so far (in the set $Q$). Using $p_{\text{lo}}$, the stopping rules are defined as follows.

First, similarly to Algorithm $S$, we can stop if the normalized value of $\hat{p}$ (i.e., the value of $\hat{p}$ w.r.t. the full input) is smaller than the bound on $p^*$. This test is done in Step 1. Second, we bound the probability that an “important” item did not arrive at the top of the current sample and therefore was not counted. 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 the top items counted by the algorithm contain an $\varepsilon$-approximation to the top-$k$ set (Step 4). We remark that even though this test is useless in some cases (say, the uniform
distribution), its effect on some real-life distributions (such as Zipf) is significant, as we show later.

### 5.3.1 Analysis

Theorem 5.6 shows some general bounds on the communication complexity of Algorithm $R$. As expected, it shows that for some inputs, Algorithm $R$ has very little advantage over Algorithm $S$. Theorem 5.11, however, presents an alternative analysis which, when applied to some natural input distributions, attains much better bounds than those of Theorem 5.6.

**Theorem 5.6** For any input $X \in \mathcal{X}(W, n, p^*)$, with probability at least $1 - \delta$, Algorithm $R$ outputs a top-$k$ $\epsilon$-approximation while the communication complexity satisfies $C_R = O\left(\frac{1}{p^* \epsilon^2} \log \frac{1}{\delta p^*} (\log |I| + \log \log W)\right)$.

Again note that when $|I| < \log W$, the communication complexity of Algorithm $R$ is within a constant factor of the complexity of Algorithm $B$. The proof of Theorem 5.6 is given in the appendix.

**Proof of Theorem 5.6**: 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 $\text{top}(k, X)$ 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 $\delta' \overset{\text{def}}{=} \delta/7$ to express error probabilities and $L' \overset{\text{def}}{=} 5 + \log \frac{1}{p^*}$ to express a bound on the number of iterations.

**Lemma 5.7** If at the end of Step 4 of the algorithm, $\text{top}(k, Q)$ is a top-$k$ $\frac{\epsilon}{4}$-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

\[ O\left(\frac{1}{p^*} \log \frac{1}{p^*} \log |I| + \frac{1}{p^* \varepsilon^2} \log \frac{1}{p^* \delta} \log \log W\right) \]

**Proof:** Let \( j \) be the least popular item in \( \text{top}(k, Q) \). 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 \( \varepsilon \). 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 4.3 we get the communication cost bound.

It remains to show that with high probability, the conditions of Lemma 5.7 hold. We show this separately for each of the two stopping conditions. We start with a lemma about the number of iterations.

**Lemma 5.8** With probability at least \( 1 - 3\delta' \), Algorithm \( R \) runs at most \( L' \) iterations before it stops.

**Proof:** The proof is analogous to the proof of Lemma 5.5. Recall that \( L' \equiv \log \frac{1}{p^*} + 5 \) 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 \( \text{safef} \) is the same as the stopping condition of Algorithm \( S \). Denote by \( p' \) the left hand side of this condition, i.e. \( p' \equiv \hat{p} \cdot (\hat{W}[\ell]/\hat{W}[1]) \). 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 \( E'_3 : p(j) \leq \frac{1}{32} p^* \)
- Event \( E'_{3'} : p(j) \geq \frac{1}{2} p^* \) AND \( p' \geq p(j)_{lo} \).

First consider the probability of \( E'_3 \) on iteration \( L' \). Clearly there is some item \( j' \in \text{top}(k, X) \) 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 5.1, \( \Pr \{ e' \} < \delta' \). Since \( e'' \) means an error in one of the high-probability-estimates, its probability is also smaller than \( \delta' \) 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 \( \{ \frac{1}{16} p(j) > p(j)_{lo} \} \). Recall that \( p(j)_{lo} \overset{\text{def}}{=} \frac{\hat{w}(j)}{2W_{hi}} \) where \( \hat{w}(j) \) was obtained by Algorithm BoundCount using \( \varepsilon = 1 \). By Corollary 4.3 the probability of the last event is less than \( \delta' \) and therefore \( \Pr \{ E''_3 \} < \delta' \). Since \( E_3 \) requires that either \( E'_3 \) or \( E''_3 \) occur on iteration \( L' \) we have that \( \Pr \{ E_3 \} < 3\delta' \).

\[ \square \]

**Lemma 5.9** If Algorithm \( R \) terminated by the stopping condition in Step 1 of safe, then by the end of Step 4 of Algorithm \( R \), \( \text{top}(k, Q) \) was a top-\( k \) \( \frac{\varepsilon}{4} \)-approximation with probability at least \( 1 - 5\delta' \).

**Proof:** Let \( j' \) denote the least popular item in \( \text{top}(k, Q) \) after the last iteration of the algorithm. If \( \text{top}(k, Q) \) 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 \( \text{top}(k, S) \) on the last iteration; clearly \( w(j) < w(j') \) and it follows that \( \text{top}(k, S) \)
is not a $\frac{\epsilon}{4}$-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 5.1 we get that if $\text{top}(k, Q)$ doesn’t satisfy the result then either we used $\hat{p} > p^*(X', k)$ or we had a sampling error that occurs with probability $\delta'$. 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 $\eta$ on each iteration. By Lemma 5.8, with probability $1 - \delta'$, there are at most $L'$ iterations. For $\eta \leq \delta' / L'$ we get an overall error probability of at most $5\delta'$.

Lemma 5.10 If Algorithm $R$ terminated by the stopping condition in Step 4 of safe, then by the end of Step 4 of Algorithm $R$, $\text{top}(k, Q)$ was a top-$k$ $\frac{\epsilon}{4}$-approximation with probability at least $1 - \delta'$.

Proof: Let $j'$ denote the least popular item in $\text{top}(k, Q)$ after the last iteration of the algorithm. If $\text{top}(k, Q)$ does not satisfy the result (i.e. it is not a top-$k$ $\frac{\epsilon}{4}$-approximation) then there exists an item $i' \notin Q$ such that $w(i') \geq (1 + \epsilon/4) \cdot w(j')$. Letting $j$ be the least popular item in $T^S$ 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 $\exp\left(-\frac{1}{8} S \cdot p(i) \cdot \Theta\left(\frac{(n-1)}{\alpha}\right)^2\right)$ by Lemma A.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 5.7 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 \( O \left( S^*(p^*, \varepsilon/4, \delta' \log \frac{1}{p^*}) \log |I| \right) \) with probability at least \( 1 - \delta' \). By adding the communication cost of Step 6 as specified in Lemma 5.7 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 5.1** Fix the input distribution. For any integer \( \ell \geq 0 \), \( \pi_\ell \) is the probability that a random item has global frequency at most \( 2^{-\ell} \). Formally: \( \pi_\ell(X) \overset{\text{def}}{=} \sum_{i: p_X(i) \leq 2^{-\ell}} p_X(i) \).

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

**Theorem 5.11 (Main result)** For an execution of Algorithm \( R \) with an input \( X \in \mathcal{X}(W, n, p^*) \) and parameters \( \delta > 0, \varepsilon > 0 \), with probability at least \( 1 - \delta \), the output is a top-\( k \) \( \varepsilon \)-approximation with individual communication complexity

\[
c_R(X) = O \left( \frac{1}{\varepsilon^2} \log \frac{1}{p^*} \delta \left( \log |I| + \log \log W \right) \cdot \sum_{\ell=1}^{\log \frac{1}{p^*}} 2^\ell \pi_\ell \right),
\]

**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 \( \pi_\ell \) 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 \( \ell \)th iteration, and let \( c[\ell] \) denote the individual-
communication used throughout the $\ell^{th}$ iteration. As before we use $\delta' \equiv \delta/7$ to express error probabilities.

**Lemma 5.12** The communication done on the $\ell^{th}$ iteration of Algorithm $R$ is of order $O\left(\frac{1}{\hat{p}[\ell]} \frac{1}{2^\ell} \log \frac{1}{\hat{p}^2} \log |\mathcal{I}| \right)$.

**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 $O\left(S^*(\hat{p}[\ell], \varepsilon/4, \delta') \cdot \log |\mathcal{I}| \right)$. The result follows using the bound for $S^*$ as given in Lemma 5.1.

**Lemma 5.13** Define $\ell' \equiv \ell + \left\lfloor \log \left(1 + \frac{\log \log W}{\log |\mathcal{I}|} \right) \right\rfloor$ and $\delta'' \equiv \delta'^{(1/2^\ell)}$. By the end of iteration $\ell'$, with probability at least $\delta''$, 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 $\ell'$ 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 $p_X(i) > 2^{-\ell} / R[\ell'] > \hat{p}[\ell']$ in the modified input and thus the number of such items is bound by $|Y| < 2^\ell R[\ell']$. It follows that a sample of size $S^*(\hat{p}[\ell'], \varepsilon/4, \delta')$ ensures that with probability at least $\delta''$, all items having $p_X(i) > 2^{-\ell}$ are in the top-items of the sample. 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 5.14** On iteration number $\ell' + 1$ we have $\hat{p}[\ell' + 1] \leq 2^{-\ell'}/\pi_\ell$ with probability at least $1 - \delta''$. 

32
**Proof:** By Lemma 5.13, 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 \( \hat{\rho} \).

Using the above lemmas we go back to the proof of Theorem 5.11. 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 \( L \) is bound by \((5 + \log(1/p^*))\) with high probability. For \( \Delta \overset{\text{def}}{=} 1 + \log \left( 1 + \frac{\log \log W}{\log |\mathcal{I}|} \right) \) we get

\[
\frac{1}{\varepsilon^2} \log \frac{1}{p^* \delta} \log |\mathcal{I}| \sum_{\ell=1}^L 2^\ell \pi_{\ell-\Delta}.
\]

By shifting the indices and using \( 2^\Delta \log |\mathcal{I}| = 2 (\log |\mathcal{I}| + \log \log W) \) we get

\[
c'(X) < \frac{1}{\varepsilon^2} \log \frac{1}{p^* \delta} \left( \log |\mathcal{I}| + \log \log W \right) \left( 2 + 2 \sum_{\ell=1}^L 2^\ell \pi_{\ell} \right) \tag{5.1}
\]

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 \( p' = \Omega(p^*) \), 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 \( Q_\ell \) for \( 1 \leq \ell \leq \log(1/p') \), where \( Q_\ell \overset{\text{def}}{=} \{ i \in Q \mid 2^{-\ell+1} \leq p_X(i) \leq 2^{-\ell} \} \). Then

\[
|Q| = \sum_{\ell=1}^{\log(1/p')} |Q_\ell| < \sum_{\ell=1}^{\log(1/p')} \frac{\pi_{\ell-1} - \pi_\ell}{2^{-\ell}},
\]

and hence

\[
c''(X) = O \left( |Q| \log |\mathcal{I}| + |Q| \frac{1}{\varepsilon^2} \log \frac{|Q|}{\delta} \log \log W \right) \tag{5.2}
\]

\[
\quad = O \left( \sum_{\ell=1}^{\log(1/p')} 2^\ell (\pi_{\ell-1} - \pi_\ell) \left( \log |\mathcal{I}| + \frac{1}{\varepsilon^2} \log \frac{1}{p^* \delta} \log \log W \right) \right).
\]

Combining Eq. (5.1) and Eq. (5.2), we conclude that

\[
c_R(X) = O \left( \frac{1}{\varepsilon^2} \log \frac{1}{p^* \delta} \left( \log |\mathcal{I}| + \log \log W \right) \sum_{\ell=1}^{\log(1/p')} 2^\ell \pi_\ell \right) \quad \blacksquare
\]

33
Using Theorem 5.11, we can prove the following corollaries for specific distributions.

**Corollary 5.15** Let $X$ be an instance where the item-frequencies have a geometric distribution, i.e. $p_X(i) = (1 - \lambda)\lambda^{i-1}$, where $\lambda$ is a constant satisfying $0 < \lambda < 1$. Then Algorithm $R$, when run on input $X$ with parameters $\delta, \varepsilon > 0$ has communication complexity $c_R(X) = O\left(\frac{k}{\epsilon^2} \log \frac{1}{p^*\delta} \left(\log |I| + \log \log W\right)\right)$.

**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 5.11. Let $L = \log \frac{1}{p^*}$. Then $\sum_{\ell=1}^{L} 2^\ell \pi_\ell = \sum_{\ell=1}^{L} \frac{\lambda}{1 - \lambda} = \frac{L\lambda}{1 - \lambda}$. Since $L = \log \frac{\lambda}{1 - \lambda} + k \log \frac{1}{\lambda}$, for constant $\lambda$, $L = \Theta(k)$. We obtain $\sum_{\ell=1}^{L} 2^\ell \pi_\ell = \Theta(k)$. The result follows immediately by Theorem 5.11. \[\square\]

Note that applying Theorem 5.6 to the geometric instance would give a similar bound except that instead of factor $k$ we would get factor $1/\lambda^k$. This exponential difference demonstrates the importance of Theorem 5.11 which reflects the advantages of Algorithm $R$ over Algorithm $S$.

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

**Corollary 5.16** Let $X$ be an instance where item-frequencies adhere to Zipf distribution with parameter $a > 1$, i.e. $p_X(i) \overset{\text{def}}{=} \frac{i^{-a}}{h}$, where $h$ is the normalization factor. Algorithm $R$, when run on input $X$ with parameters $\delta, \varepsilon > 0$ has communication complexity $c_R(X) = O\left(\frac{a + 1}{a - 1} \cdot \frac{k}{\epsilon^2} \cdot \frac{1}{p^*\delta} \left(\log |I| + \log \log W\right)\right)$.

**Proof:** We first find the items summed up in $\pi_\ell$ for a given $\ell$. Requiring $p(i) \leq 2^{-\ell}$ yields $i \geq \left(\frac{2^\ell}{h}\right)^{1/a}$. Next we approximate $\pi_\ell$ by $1 - F((2^\ell/h)^{1/a})$ where $F(z) \overset{\text{def}}{=} \Pr\{Z \leq z\}$ 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.
distribution. We obtain that for \( a > 1 \), \( \pi_\ell \approx \frac{1}{a-1} \left( \frac{a}{h} \right)^{1/a} \). Let \( L \) denote the value of \( \log \frac{1}{p^*} = \log \frac{k-a}{h} \). Then

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

and the result follows from Theorem 5.11.

As before, we compare the result of Corollary 5.16 with the result of Theorem 5.6 for the Zipf instance. In this case the ratio between the bounds is polynomial: instead of factor \( k \), Theorem 5.6 gives factor \( k^a \). While this difference is not as dramatic as the exponential difference seen above, it is still important when \( k \) is large and the distribution is Zipf with exponent far from 1 (e.g. \( a = 2 \)).

The results in this section analyze the behavior of \( R \) for asymptotically large inputs. As the next chapter describes, we have also tested our algorithms on realistic-size inputs to study the effect of the hidden constants. Having compared Algorithm \( R \) with other known algorithms from the literature we remark that it turns out to have significantly superior performance both asymptotically and for realistic inputs.
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) $\hat{W} \leftarrow \text{BoundCount}(\varepsilon = 1, \delta/7, \text{`ALL'})$.
   - (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) $\text{P}_{\text{SAMPLE}} \leftarrow 2S^*(\hat{p}/4, \delta/7)$.
   - (3e) Execute Algorithm $A$ with parameter $\text{P}_{\text{SAMPLE}}$ to get sample $S$.
   - (3f) $\eta \leftarrow (\delta/7) \cdot (1/\log p_{\text{lo}}(\text{top}(k, Q), \hat{W}))$.
   - (3g) $T_S \leftarrow \text{top}(\log |I| / (\log |I| + \log \log \hat{W}), S)$. For each $i \in T_S$, $\hat{w}(i) \leftarrow \text{BoundCount}(\varepsilon = 1, \eta, i)$.
   - (3h) $Q \leftarrow Q \cup T_S$; remove elements of $T_S$ from the input.
4. Until $\text{safe}(\hat{W}, Q, S, \hat{p}, \varepsilon, \delta)$.

5. If $|T_S| < k$, then for each $i \in \text{top}(k, S)$, do $\hat{w}(i) \leftarrow \text{BoundCount}(\varepsilon = 1, \eta, i)$ and add $i$ to $Q$.

6. $Q \leftarrow \left\{ i \in Q \mid \hat{w}(i)/\hat{W} > p_{\text{lo}}(\text{top}(k, Q), \hat{W}) \right\}$

7. For each $i \in Q$, $\hat{w}(i) \leftarrow \text{BoundCount}(\varepsilon/4, \delta/5|Q|, i)$.

8. Output the top $k$ items in $Q$ according to the new estimates.

Function $p_{\text{lo}}$

Input: $Q, W$

1. Return $\frac{1}{4W} \min \{ \hat{w}(i) \mid i \in Q \}$.

Predicate $\text{safe}$

Input: $W, Q, S, \hat{p}, \varepsilon, \delta$

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

Figure 5.2: Algorithm $R$
Chapter 6

Simulations results

To evaluate the performance of our algorithm on 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 [13] and TPUT [7]. 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 $\varepsilon$ (the approximation parameter). Informally, we find that Algorithm $R$ offers a dramatic improvement over TA and TPUT unless the system is small and the distribution is favorable to TA and TPUT. Moreover, for typical inputs, $R$ behaves better than predicted by our analytical results.

**Simulated instances.** We used randomly generated inputs of various sizes, and our main focus was on inputs generated according to 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 $\delta = 0.05$.

All simulations measured the individual communication (in bits) used by the root-node; this is justified by the fact that in all simulated algorithms, the root node performs more communication than any other node.
6.1 Comparison with existing algorithms

To compare algorithms $R$, TA and TPUT we ran simulations using instances of size between $10^6$ and $10^7$ ($10^6 \leq W \leq 10^7$). The varying parameter was the number of nodes ($n$) and the number of candidate items was equal to the number of nodes ($|I| = n$). The simulation of TA followed the approximation version of TA as described in [13] where the approximation parameter $\theta$ corresponds to $1 + \varepsilon$ in our notation. Noting that TPUT has no approximation version, we had to simulate the exact algorithm. We comment however that in all simulations we ran, TPUT appeared less efficient than TA, even for high accuracy such as $\varepsilon = 0.01$. Following the recommendation in [7], we used $\alpha = 0.5$ for the pruning factor of TPUT.

Results for adversarial and random partition of scores were similar. As shown in Figure 6.1, 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.
Figure 6.2: 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.

Figure 6.3: Communication costs of algorithms $S$ and $R$. Left: as a function of $p^*$. Right: as a function of $\varepsilon$. The communication cost of reading the entire input is $W \log |I|$, ($W$ is the input size and $|I|$ is the number of items).

while traversing the tree). These modifications resulted in drastic improvement in TA performance: see Figure 6.2. 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.

### 6.2 The effect of accuracy and critical frequency on performance

We also examined the performance of algorithms $R$ and $S$ while varying $\varepsilon$ and $p^*$. The results are given in Figure 6.3. The results for $p^*$ (Figure 6.3, left) show that Algorithm $R$ is less sensitive to $p^*$ when the frequencies adhere to Zipf distri-
bution. This improvement is directly related to the effectiveness of the “shaving” technique used by Algorithm \( R \). We note that the flat left segment in the graph of Algorithm \( S \) is due to the fact that for small values of \( p^* \), Algorithm \( S \) obtains a sample of the entire input.

The effect of the approximation factor \( \varepsilon \) (seen in Figure 6.3, right) on the communication of both algorithms is proportional to \((1 + \varepsilon)^2\), as suggested by our upper bounds. Algorithm \( R \) behaves much better due to its better termination criterion. Specifically, recall that the second stopping rule in Predicate \texttt{safe} considers a hypothetical item whose probability is \( q = (1 + \varepsilon/2)p^* \) and bounds the probability that it was not included in \( T^S \) on the last sample. This bound depends on 3 parameters: \( S \) (the sample size), \( q \) (the hypothetical item frequency), and \( \alpha \) (the ratio between \( q \) and \( r \) where \( r \) is the smallest empirical probability in \( T^S \)). If \( \alpha \) was equal to \( 1 + \varepsilon \) we would expect the same \((1 + \varepsilon)^2\) behavior. However, as \( \varepsilon \) approaches 0, and the algorithm performs more iterations, even very unpopular items are included in \( T^S \). This causes \( \alpha \) to increase rather than to approach 1 and the strong effect of \( \varepsilon \) diminishes.

A common observation in all performed simulations is that Algorithm \( R \) was consistently better than Algorithm \( S \). In fact, the only case where the algorithms have similar performance is the “step” distribution where \( k \) items have probability \( p^* \), and all other items have probability \( p^*/(1 + \varepsilon) \). In this case, both Algorithm \( S \) and Algorithm \( R \) sample the entire input before terminating.
Chapter 7

A lower bound on the total cell-probe complexity

In this section we give a lower bound on the total number of cells probed by 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.

Let $c_{p_A}^T(X)$ denote the total number of cell-probes throughout the execution of algorithm $A$ on an instance $X$, i.e., $c_{p_A}^T(X) \overset{\text{def}}{=} \sum_{v \in V} c_{p_A}(X, v)$. The total cell-probe complexity of $A$ is the total number of cells probed throughout the execution of $A$ with the worst instance, i.e. $C_{p_A}^T(\mathcal{X}) \overset{\text{def}}{=} \max \{ c_{p_A}^T(X) \mid X \in \mathcal{X} \}$. Note that the set of all probed cells can be viewed as a sample whose size is the total number of cell-probes.

Next we give our lower bound. The proof is given at the end of this chapter. We note that our lower bound resembles well known lower bounds on Monte Carlo estimation (see [6, 10]). The most familiar bound asserts that to estimate the expectation of a random variable with accuracy $\varepsilon$ and error probability $\delta$, one requires $\Omega \left( \frac{1}{\varepsilon^2} \log(1/\delta) \right)$ independent samples [6]. We comment however that these bounds do not translate to a bound on top-$k$ approximation since top-$k$ queries require only an identification of the top items and not an estimation of their weight. Furthermore, our bound presents an additional factor of $1/p^*$ which
Theorem 7.1 For some natural \( k \), let \( \epsilon > 0 \), \( 0 < p^* \leq \frac{1}{2k} \), \( 0 < \delta < 0.014 \), and \( W = \Omega \left( \frac{1}{3(p^*)^2} \cdot \frac{(1+\epsilon)^2}{\delta^4} \right) \) and let \( A \) be any Monte Carlo algorithm. If \( A \) outputs a top-\( k \) \( \epsilon \)-approximation with probability at least \( 1 - \delta \) for any input in \( X(W, n, p^*) \), then its total cell-probe complexity satisfies \( E[CP_A] = \Omega \left( \frac{1}{p^* \epsilon^2} \cdot \log \frac{1}{\delta} \right) \).

Star topology and “smart dust” systems. Theorem 7.1 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., [25]).

Now, for these systems, Theorem 7.1 says that any algorithm that satisfies the conditions of the theorem has individual communication complexity at least \( \Omega \left( \frac{1}{p^* \epsilon^2} \cdot \log \frac{1}{\delta} \right) \).

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 star topology.

Proof of Theorem 7.1: 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 \( \epsilon \), so that the correct output for any \( \epsilon \)-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-\(k\) \(\varepsilon\)-approximation with probability at least \(1 - \delta\) for a random input \(X\) drawn by \(D\). To formalize this, we define the proximity, \(\rho(Q)\) of a set \(Q\) to \(\text{top}(k, X)\); \(\rho(Q)\) is defined as the minimal \(\varepsilon\) for which \(Q\) is a top-\(k\) \(\varepsilon\)-approximation, i.e.,

\[
\rho(Q) \overset{\text{def}}{=} \frac{\max \{p(j) \mid j \in \mathcal{I}\setminus Q\}}{\min \{p(i) \mid i \in Q\}} - 1.
\]

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

\[
\mathcal{A}(D, \varepsilon, \delta) \overset{\text{def}}{=} \left\{ A : \Pr_{X \sim D} \{\rho(A(X)) \leq \varepsilon\} \geq 1 - \delta \right\}.
\]

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: \(E_{X \sim D} \left[ \text{cp}_A^T(X) \right] \geq b\), 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 \(E \left[ \text{CP}_A^T \right] \geq \max \{ E \left[ \text{cp}_{A_R}(X) \mid X \in D \right] \geq \frac{b}{2} \).

Lemma 7.2 that follows gives such a lower bound for deterministic algorithms with \(b = \Omega \left( \frac{1}{p^*} \cdot \frac{(1 + \varepsilon)}{\varepsilon^2} \cdot \log \frac{1}{2\delta} \right)\) giving the result of the theorem.

**Lemma 7.2** Let \(k\) be a natural number. Let \(\varepsilon > 0\), \(0 < p^* \leq \frac{1}{2k}\), \(0 < \delta < 0.014\), \(W = \widetilde{\Omega} \left( \frac{1}{2p^*} \cdot \frac{(1+\varepsilon)^2}{\varepsilon^2} \right)\). 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-\(k\) \(\varepsilon\)-approximation on an input \(X\) chosen randomly from \(D\) with probability at least \(1 - \delta\), then \(E_{X \sim D} \left[ \text{cp}_A^T(X) \right] = \Omega \left( \frac{1}{p^*} \cdot \frac{(1+\varepsilon)}{\varepsilon^2} \cdot \log \frac{1}{2\delta} \right)\).
**Proof:** For the given values $W, k, p^*, \varepsilon > 0$ ($0 < p^* \leq \frac{1}{2k}$), construct the input distribution $D$ as follows. Let $U(X)$ be the uniform distribution above the permutations of $X$ and define $q \overset{\text{def}}{=} \frac{p^*}{\alpha}$ 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:

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

- 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.

\text{i.e. Let } X_0' \sim U(X_0) \text{ and let } X_{v_\ell} \overset{\text{def}}{=} \{X_0'[\frac{\ell-1}{n}W], \ldots, X_0'[\frac{\ell}{n}W]\}.

Note that for any input $X$ in the support of $D$, only the top-$k$ set is a top-$k$ $\varepsilon$-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-$k$ $\varepsilon$-approximation with probability at least $1 - \delta$ for a random input $X \sim D$. Let the sample of $X$, $S_A(X)$ denote the cells actually probed throughout the execution of $A$ on input $X$. Since $A$ is deterministic, $S$ is a deterministic function of $X$ and the output of $A$ is a deterministic function of $S$. We can therefore denote by $A(S)$ the output of $A$ when $S_A(X) = S$. We first argue in that if $A$ does not output the top $k$ items in the sample $\text{top}(k, S)$, 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 7.3 Let \( A \) be any deterministic top-\( k \)-approximation algorithm. For a random input \( X \) drawn by \( D \), if the resulting sample \( S = S_A(X) \) is such that \( A(S) \neq \top(k, S) \), then \( A \) errs with probability at least \( \frac{1}{2} \).

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

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

\[
(i, j) : \ i \in \top(k, S) \setminus Y, \ j \in Y \setminus \top(k, S), \ p_s(j) \leq p_s(i), \ p_X(j) > (1 + \varepsilon)p_X(i) > p_X(i)
\]

Define \( \bar{I} \) to be the instance generated from \( \bar{I} \) by switching the names of items \( i \) and \( j \). Call \( \bar{I} \) the \( ij \)-mirror of \( \bar{I} \). Consider the set \( \mathcal{X} \) of all possible inputs which, for the sampling function of \( A \), result in either \((\bar{w}, \bar{I})\) or its \( ij \)-mirror \((\bar{w}, \bar{I})\). 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 \( p_s(i) > p_s(j) \), and let \( S_j \) denote the samples where \( p_s(i) \geq p_s(j) \) and by \( S_E \) the samples where \( p_s(i) = p_s(j) \).

Figure 7 illustrates the relations of the weights of \( i, j \) during the sampling process. Let \( \pi \) 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: \( \gamma \overset{\text{def}}{=} \Pr \{S_i \mid D_i\} - \Pr \{S_j \mid D_i\} \),
and \( \eta \overset{\text{def}}{=} \Pr \{ S_E \mid D_i \} = \Pr \{ S_E \mid D_j \} \). With this notation, we have the following identities.

\[
\begin{align*}
\Pr \{ S_i \mid D_i \} &= \Pr \{ S_j \mid D_j \} = \frac{1}{2} (1 - \eta + \gamma) \\
\Pr \{ S_j \mid D_i \} &= \Pr \{ S_j \mid D_i \} = \frac{1}{2} (1 - \eta - \gamma) \\
\Pr \{ S_j \} &= \Pr \{ S_j \} = \pi (1 - \eta) \\
\Pr \{ S_E \} &= 2\eta\pi ,
\end{align*}
\]

and therefore, since \( \gamma > 0 \), we obtain

\[
\Pr \{ Y \text{ is correct} \mid \vec{w} \} < \Pr \{ D_j \mid S_i \cup S_E \} < \frac{1}{2} . \tag{7.1}
\]

**Lemma 7.4** 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 \( W = \tilde{\omega} \left( \frac{1}{\delta p^*} \cdot \frac{(1 + \varepsilon)^2}{\varepsilon^4} \right) \) 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 $\delta$, 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 $S$ denote the ordered-set of $X_0$-cells sampled by $A_r$. Then

\[ \Pr \{ \text{there are repetitions in } S \} \leq \sum_{\ell=1}^{S} \frac{\ell - 1}{W} < \frac{S^2}{2W}. \]

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

\[ \Pr \{ A_r \text{ errs} \} \leq 1 - \Pr \{ S \in G \} \leq \delta + \frac{S^2}{2W}(1 - \delta) < \delta + \frac{S^2}{2W}, \]

which implies that for $S \leq \sqrt{2W} \delta$, $\Pr \{ A_r \text{ errs} \} < 2\delta$. Lemma 7.5 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 $S = \Omega \left(\frac{1}{p^*} \cdot \frac{\alpha}{(1 - \alpha)^2} \cdot \ln \frac{1}{4\delta}\right)$.

**Lemma 7.5** For $\delta < \frac{1}{3e^3}$, there exists a value $S_{\text{min}} = \Theta \left(\frac{1}{p^*} \cdot \frac{\alpha}{(1 - \alpha)^2} \cdot \ln \frac{1}{27}\right)$ such that if $A_r$ is run on an input chosen randomly by $D$ and it uses a sample smaller than $S_{\text{min}}$, then $A_r$ errs with probability at least $\delta$. 

47
**Proof**: For a specific execution of $A_r$ we have a specific input $X$, and a specific sample $S$ of cells probed by $A_r$. For this input, let $i \in \text{top}(k, X)$ and some $j \in \mathcal{I} \setminus \text{top}(k, X)$ and define the *advantage within the sample* of $i$ over $j$ to be $d_{ij}(S) \overset{\text{def}}{=} w^S(i) - w^S(j)$. For a permutation $Z = (z_1, \ldots, z_S)$ of $S$, 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 $S$ 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 $w^S(i) < w^S(j)$. Since $i \in \text{top}(k, X)$ and $j \notin \text{top}(k, X)$, it is obvious that this case $\text{top}(k, S) \neq \text{top}(k, S)$ 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 $X'' \overset{\text{def}}{=} M_{ij}(X')$. By definition of $D$, $\Pr \{ X' \} = \Pr \{ X'' \}$. Since for $Z'' \overset{\text{def}}{=} M_{ij}(Z')$, 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 $\Pr \{ A \text{ errs} \mid d_{ij}(S) = 0 \} \geq \frac{1}{2}$.

Finally, we bound the overall error probability of $A_r$ by the probability that $d_{ij} < 0$. Let $\pi$ 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 $S = (\nu_1, \ldots, \nu_S)$, let $S' \overset{\text{def}}{=} M_{ij}(S)$ be the mirror sample. Clearly $d_{ij}(S') = -d_{ij}(S)$. Next we show
the relation between the probability of a sample $S$ and the probability of its $ij$-mirror, $S'$. 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 $\text{perms}(S)$ denote the number of distinct permutations of $S$. Clearly we have $\text{perms}(S) = \text{perms}(S')$. For brevity, let $a \overset{\text{def}}{=} w^S(i)$ and $b \overset{\text{def}}{=} w^S(j)$. Clearly $d_{ij} = a - b$. Furthermore, for a specific input $X$ we have

$$\Pr \{ S \mid X \} = \text{perms}(S) \cdot \prod_{\ell=1}^{S} p_X(v_{\ell})$$

$$= \text{perms}(S) \cdot p_X(i)^a \cdot p_X(j)^b \cdot \prod_{\ell \in \{1..S\}, v_{\ell} \notin \{i,j\}} p_X(v_{\ell})$$

Similarly,

$$\Pr \{ S' \mid X \} = \text{perms}(S') \cdot \prod_{\ell=1}^{S} p_X(v'_{\ell})$$

$$= \text{perms}(S) \cdot p_X(i)^b \cdot p_X(j)^a \cdot \prod_{\ell \in \{1..S\}, v_{\ell} \notin \{i,j\}} p_X(v_{\ell})$$

yielding,

$$\Pr \{ S' \mid X \} = \Pr \{ S \mid X \} \cdot p_X(i)^{b-a} \cdot p_X(j)^{a-b}$$

$$= \Pr \{ S \mid X \} \cdot \left(\frac{1}{\alpha}\right)^{d_{ij}}$$

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

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

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 $d_{ij}(S)$ are:

$$E[d'(\nu_{\ell})] = q(\alpha - 1)$$

$$\text{Var}(d'(\nu_{\ell})) = q\alpha + q - (q\alpha - q)^2$$

$$E[d_{ij}] = S(\alpha - 1)q$$

$$\text{Var}(d_{ij}) = S(q\alpha + q - (q\alpha - q)^2)$$

$$\mu_d = E[d_{ij}] = S(\alpha - 1)q$$

$$\sigma_d^2 = \text{Var}(d_{ij}) = S(q\alpha + q - (q\alpha - q)^2)$$

$$\sigma_d^2 < E[(d_{ij})^2] = S(\alpha + 1)q$$
We also note that by definition $-S \leq d_{ij} \leq S$. By Chebyshev inequality, if we pick a random sample of size $S$, we get:

$$\Pr \{0 < d_{ij} < 7\mu_d\} = \pi - \Pr \{d_{ij} - \mu_d \geq 6\mu_d\} > \frac{1}{2} - \frac{\sigma_d^2}{6\mu_d^2} > \frac{1}{2} - \frac{(\alpha + 1)}{S q (\alpha - 1)^2 6^2}.$$ 

Since the transformation from $S$ to $S'$ 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:

$$\Pr \{-7\mu_d < d_{ij} < 0\} > \Pr \{0 < d_{ij} < 7\mu_d\} \cdot \left(\frac{1}{\alpha}\right)^7\mu_d > \left(\frac{1}{2} - \frac{(\alpha + 1)}{S q (\alpha - 1)^2 6^2}\right) \left(\frac{1}{\alpha}\right)^7\mu_d.$$ 

For $S \geq S_1$, where $S_1 \overset{\text{def}}{=} \frac{(\alpha + 1)}{6 q (\alpha - 1)^2}$, we can say that

$$\Pr \{-7\mu_d < d_{ij} < 0\} > \frac{1}{3} \left(\frac{1}{\alpha}\right)^7\mu_d = \frac{1}{3} \exp (-7 \ln \alpha \mu_d) > \frac{1}{3} \exp (-7(\alpha - 1)\mu_d) = \frac{1}{3} \exp (-7S q (\alpha - 1)^2).$$

And for $S_1 \leq S < S_2$ where $S_2 \overset{\text{def}}{=} \frac{1}{2 q} \cdot \frac{1}{(\alpha - 1)^2} \cdot \ln \frac{1}{3\delta}$, we get

$$\forall X \in D, \quad \Pr \{A_r \text{ errs} \mid X\} = \Pr \{S : \text{A errs for } S \mid X\} = \Pr \{S : \exists (i, j), i \in \text{top}(k, X), j \notin \text{top}(k, X), d_{ij}(S) < 0 \mid X\} > \Pr \{S : -7\mu_d < d_{ij}(S) < 0 \mid X\} \quad \text{for a single pair } (i, j) > \delta.$$ 

The above applies for cases where $S_1 < S_2$ which is satisfied for $\delta < \frac{1}{36^3}$, $\alpha < \frac{11}{7}$. We have shown a specific sample size $S_2$ for which $A_r$ has an error.
probability larger than \( \delta \). 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 \( \delta \). Lemma 7.3 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 \( \alpha \), 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 7.2. Lemma 7.3 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 7.4 showed that 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 \( \delta \). Since this lower bound is concave in \( \delta \), 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 \( \delta \), then the expected number of probed cells is of the same order: \( E_{X \sim D} [\text{cp}(X)] = \Omega \left( \frac{1}{p^*} \cdot \frac{\alpha}{(\alpha-1)^2} \cdot \log \frac{1}{4\delta} \right) \).
Chapter 8

Conclusions

In this work 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.
Bibliography


Appendix A

Additional Proofs

Proof of Lemma 5.1: We say that a pair \((i, j)\) of items is *important* if \(i \in \text{top}(k, X), j \notin \text{top}(k, X), \) and \(p_X(i) > (1 + \varepsilon) \cdot p_X(j)\).

A pair \((i, j)\) of items is said to be *poorly-swapped* in \(S\) if it is important AND \(p_S(i) \leq p_S(j)\). Note that top\((k, S)\) is a top-\(k\) \(\varepsilon\)-approximation if and only if there are no poorly-swapped pairs in \(S\).

For each item \(i \in \text{top}(k, X)\) we define a threshold value \(t_i\) satisfying \(\frac{p_X(i)}{1+\varepsilon} < t_i < p_X(i)\) and for each item \(j \notin \text{top}(k, X)\) we define a threshold value \(t'_j\) by \(t'_j \overset{\text{def}}{=} \min \{t_i \mid (i, j) \text{ is important} \}\). We denote by \(E_i\) the event of receiving a sample \(S\) where \(p_S(i) \leq t_i\). This event is also referred to as “\(i\) violates its threshold”. We denote by \(E'_j\) the event of receiving a sample \(S\) where \(p_S(j) \geq t'_j\). This event is also referred to as “\(j\) violates its threshold”.

From the above properties of the thresholds, if no \(i \in \text{top}(k, X)\) violates its threshold \(t_i\), and no \(j \notin \text{top}(k, X)\) violates its threshold \(t'_j\), then there are no poorly-swapped pairs in \(S\) 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. \(G^1 \overset{\text{def}}{=} \{i \in \text{top}(k, X)\}\)
- \(G^2 \overset{\text{def}}{=} \{j \notin \text{top}(k, X) \mid \frac{p^*}{(1+\varepsilon)e^2} \leq p_X(j) \leq p^*\}\)
- \(G^3 \overset{\text{def}}{=} \{j \notin \text{top}(k, X) \mid p_X(j) < \frac{p^*}{(1+\varepsilon)e^2}\}\)
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 A.1** Let $X$ be a given input and let $i$ be some item from $I$ having empirical probability $p_X(i)$. If $S$ is a random sample of size $S$ obtained by Algorithm A then

$$\forall \varepsilon > 0 : \Pr \{ p_S(i) \geq (1 + \varepsilon) p_X(i) \} \leq \left( \frac{e^\varepsilon}{(1 + \varepsilon)^{1+\varepsilon}} \right)^{SP_X(i)}$$

$$\forall \varepsilon < 0 : \Pr \{ p_S(i) \leq (1 + \varepsilon) p_X(i) \} \leq \left( \frac{e^\varepsilon}{(1 + \varepsilon)^{1+\varepsilon}} \right)^{SP_X(i)}$$

**Proof:** The proof is by Chernoff bound. Since $p_S(i) \overset{\text{def}}{=} \frac{w_S(i)}{S}$, bounding the tail probability of $w_S(i)$ is equivalent. Consider the set $\{ \nu_1, \ldots, \nu_{w(i)} \}$ of all cells holding $i$ in the input and let $x_\ell$ be an indicator random variable having value 1 if the cell $\nu_\ell$ was sampled. Clearly $x_\ell$ are i.i.d random variables and the sampled-weight of $i$ satisfies $w_S(i) = \sum_{\ell=1}^{w(i)} \nu_\ell$. Since the expectation of the sampled-weight is $SP_X(i)$, we get the result by Chernoff Bound. \hfill \rlap{\quad}$

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

$$t_{ij} \overset{\text{def}}{=} p_X(i) \frac{\alpha - 1}{\alpha \ln \alpha} \quad \text{(we later show that } p_X(j) < t_{ij} < p_X(i) \text{)}$$

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

$$\Pr \{ p_S(i) \leq t_{ij} \} \leq \exp \left( -\frac{1}{8} S \cdot p_X(i) \cdot \Theta \left( \left( \frac{\alpha - 1}{\alpha} \right)^2 \right) \right)$$

$$\Pr \{ p_S(j) \geq t_{ij} \} \leq \exp \left( -\frac{1}{8} S \cdot p_X(i) \cdot \Theta \left( \left( \frac{\alpha - 1}{\alpha} \right)^2 \right) \right)$$

Note also that $t_{ij}$ is monotone decreasing with $\alpha$ and so are the ‘violation’ probabilities.
Proof: Let \( f(\alpha) \) \( \overset{\text{def}}{=} \frac{\alpha-1}{\ln \alpha} \), \( t \overset{\text{def}}{=} f(\alpha) \cdot p_X(j) \), \( e_1 \overset{\text{def}}{=} \{ p_S(i) \leq t_{ij} \} \) and \( e_2 \overset{\text{def}}{=} \{ p_S(j) \geq t_{ij} \} \). Applying the tail-probability bounds from Lemma A.1 for \( e_1 \) and \( e_2 \) gives

\[
\Pr \{ p_S(i) \leq t \} < (a_1)^{S_p_X(i)} \quad (A.1)
\]

where

\[
a_1 \overset{\text{def}}{=} \frac{e^{-\delta_1}}{(1 - \delta_1)(1 - \delta_1)} \quad , \quad \delta_1 \overset{\text{def}}{=} \frac{p_S(i) - t}{p_X(i)} \quad (A.2)
\]

and

\[
\Pr \{ p_S(j) \geq t \} < (a_2)^{S_p_X(j)} \quad (A.3)
\]

where

\[
a_2 \overset{\text{def}}{=} \frac{e^{\delta_2}}{(1 + \delta_2)(1 + \delta_2)} \quad , \quad \delta_2 \overset{\text{def}}{=} \frac{t - p_X(j)}{p_X(j)} \quad (A.4)
\]

For \( t = f(\alpha)p_X(j) \), we get from Eq. (A.2) and Eq. (A.4) that \( \delta_1 = 1 - \frac{f}{\alpha} \) and \( \delta_2 = f - 1 \) giving

\[
a_2 = a_1^\alpha = \frac{e^{(f-1)}}{(f)(f)} = \exp \left( -(f \ln f + 1 - f) \right) \quad (A.5)
\]

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. (A.1), Eq. (A.3) and from Eq. (A.5), we get that \( \Pr \{ e_1 \} < a_2^{S_p_X(j)} \) and \( \Pr \{ e_2 \} < a_2^{S_p_X(j)} \) while

\[
a_2^{S_p_X(j)} = \exp \left( -S_p_X(j) \cdot (f \ln f + 1 - f) \right) = \exp \left( -S \frac{p_X(i)}{\alpha} (f \ln f + 1 - f) \right) \quad (A.6)
\]

The function \( h(\alpha) = \frac{1}{\alpha} (f \ln f + 1 - f) \) is monotone increasing for any \( \alpha > 1 \). Let \( \hat{h}(\alpha) \overset{\text{def}}{=} \frac{(\alpha-1)^2}{8\alpha^2} \), then \( h(\alpha) = \Theta(\hat{h}(\alpha)) \). The result of the lemma follows from Eq. (A.6) by replacing \( h(\alpha) \) with \( \Theta(\hat{h}(\alpha)) \).

Continuing with the proof of Lemma 5.1 we now define for each \( i \in \text{top}(k, X) \) the exact choice of its threshold value as \( t_i \overset{\text{def}}{=} p_X(i) \frac{\varepsilon}{(1+\varepsilon) \ln(1+\varepsilon)} \); this is equal to \( t_{ij} \) defined in Lemma A.2 for \( \alpha = 1 + \varepsilon \). Given \( i \in \text{top}(k, S) \), 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 A.2, \( p_X(j) < t_i < p_X(i) \) and \( \Pr \{ E_i \} < \exp \left( -\frac{1}{8} S p_X(i) \cdot \Theta \left( \left( \frac{\varepsilon}{1 + \varepsilon} \right)^2 \right) \right) \).

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 \( \Pr \{ E'_j \} < \exp \left( -\frac{1}{8} S p_X(i) \cdot \Theta \left( \left( \frac{\varepsilon}{1 + \varepsilon} \right)^2 \right) \right) \). 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

\[
\Pr \{ E^1 \cup E^2 \} < 10 \cdot \frac{(1 + \varepsilon)}{p^*} \cdot \exp \left( -\frac{1}{8} S p^* \Theta \left( \left( \frac{\varepsilon}{1 + \varepsilon} \right)^2 \right) \right)
\]

The following lemma bounds the probability of \( E^3 \).

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

\[
\Pr \{ E^3 \} < \frac{2e^2(1 + \varepsilon)}{p^*} \cdot \exp \left( -\frac{1}{8} S p^* \Theta \left( \left( \frac{\varepsilon}{1 + \varepsilon} \right)^2 \right) \right)
\]

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

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

\[
G^3_\ell \overset{\text{def}}{=} \{ j \notin \text{top}(k, X) : t \cdot e^{-3-\ell} \leq p_X(j) < t \cdot e^{-2-\ell} \}
\]

For every item \( j \in G^3_\ell \), let:

\[
a \overset{\text{def}}{=} \frac{t}{p_X(j)}
\]

\[
I \overset{\text{def}}{=} \frac{a}{f(1 + \varepsilon)} = \frac{p^*}{(1 + \varepsilon) \cdot p_X(j)} \quad \Rightarrow \quad e^{2+\ell} < I \leq e^{3+\ell}
\]

Using the tail bounds from Lemma A.1 we get
\[ \Pr \left\{ E_j'' \right\} < \left( \frac{e^{a-1}}{a^a} \right)^{Sp_X(j)} \]
\[ = \exp \left( -(a \ln a - a + 1)Sp_X(j) \right) \]
\[ = \exp \left( -Sp^* \left( \frac{1}{1+\varepsilon} \right) \cdot (If(\ln f + \ln I) - I + 1) \right) \]
\[ = \exp \left( -Sp^* \left( \frac{\left( f \ln f - f + 1 \right) + (f \ln I - \frac{I-1}{I})}{1+\varepsilon} \right) \right) \]
\[ < \exp \left( -Sp^* \left( \frac{(1+\varepsilon)h(1+\varepsilon)}{1+\varepsilon} + (f(\ell + 2) - 1) \right) \right) \]
\[ < \exp \left( -Sp^* h(1+\varepsilon) \right) \cdot \exp \left( \frac{-Sp^*}{1+\varepsilon} \cdot f(\ell + 1) \right) \]

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

Let \( E^3_\ell \) denote the event that one or more items in \( G^3_\ell \) violate the threshold \( t \),
\[ E^3_\ell \overset{\text{def}}{=} \bigcup_{j \in G^3_\ell} E_j'' \]. By the union-bound we get
\[
\Pr \left\{ E^3_\ell \right\} \leq \sum_{\ell=0}^\infty \Pr \left\{ E^3_\ell \right\} \\
\Pr \left\{ E^3_\ell \right\} < \frac{e^{\ell+1}e^2(1+\varepsilon)}{p^*} \cdot \exp \left( \frac{-Sp^*}{1+\varepsilon}(\ell + 1) \right) \cdot \exp \left( -Sp^* h(1+\varepsilon) \right) \\
= \frac{e^2(1+\varepsilon)}{p^*} \cdot \exp \left( -(\ell + 1)(-1 + \frac{Sp^*}{1+\varepsilon}) \right) \cdot \exp \left( -Sp^* h(1+\varepsilon) \right) \\
\]
Using \( S \geq 2(1+\varepsilon)/fp^* \) we get that since \( Sp^*/(1+\varepsilon) > 2 \),
\[
\Pr \left\{ E^3_\ell \right\} < \frac{e^2(1+\varepsilon)}{p^*} \cdot \exp \left( -(\ell + 1) \cdot \frac{Sp^*}{2(1+\varepsilon)} \right) \cdot \exp \left( -Sp^* h(1+\varepsilon) \right) \\
= \frac{e^2(1+\varepsilon)}{p^*} \cdot \exp \left( -Sp^* h(1+\varepsilon) - \frac{Sp^* f}{2(1+\varepsilon)} \right) \cdot \exp \left( -\ell \cdot \frac{Sp^*}{2(1+\varepsilon)} \right) \\
\]
Note that the bounds we got for \( E^3_\ell \) form a descending geometric-series. By summing them up and using the fact that \( Sp^*/(1+\varepsilon) \geq 2 \) again we get
\[
\Pr \left\{ E^3 \right\} < \frac{2e^2(1+\varepsilon)}{p^*} \cdot \exp \left( -Sp^* \left( h(1+\varepsilon) + \frac{f}{2(1+\varepsilon)} \right) \right) \\
= \frac{2e^2(1+\varepsilon)}{p^*} \cdot \exp \left( -Sp^* \left( h(1+\varepsilon) + \frac{1}{2(1+\varepsilon)} \right) \right) \]

\[ 61 \]
To conclude the proof of Lemma 5.1, we combine the result of Lemma A.3 with the result we got for $E^1 \cup E^2$. We get that for sample size satisfying $S \geq 2(1 + \varepsilon) \ln (1 + \varepsilon)/\varepsilon p^*$, the overall error probability satisfies

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

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

Appendix B

Random sampling

While our algorithms use a specific sampling routine, our analysis is correct for any uniform sampling technique. This appendix discusses sampling more generally and generalizes Lemma A.1 accordingly. This extends our results for alternative sampling techniques.

When analyzing the results of sampling, one has to observe whether sampling is done with or without replacement. Consider for example a hat with \( N \) blue balls and \( M \) red balls. Suppose we draw at random 10 balls with the purpose of evaluating \( N \) and \( M \). Sampling with replacement would be to draw a ball, write down its color, put it back in and shuffle and only then draw the next ball. Sampling without replacement means to draw 10 balls without replacing them between draws. Note that when sampling with replacement, the hat-population remains the same throughout the process and so draw-results can be considered as 10 i.i.d. random variables (RV’s). When sampling without replacement, each draw changes the population and affects later draws; draw-results are therefore dependent RV’s with different distributions. Since i.i.d. random variables are much simpler to analyze, it is considerably easier to analyze the results of sampling with replacement. While our work is mainly analytical, our motivation is to find efficient algorithms. We later show that sampling without replacement better serves this purpose. Luckily, some known probability bounds are applicable for sam-
pling without replacement; these bounds are actually slightly stronger than their classic-equivalent. Intuitively, the result of sampling without replacement is more concentrated around the expectations because throughout the sampling process, if the first few draws deviate from expectation, the population is changed so that later draws tend to deviate in the opposite direction. This concept was formalized by Panconesi and Srinivasan in [21] by their definition of self-weakening random variables.

Recall that the sampled-weight of an item, \( w^S(i) \), is the multiplicity of \( i \) within the sample \( S \) and that the sampled-frequency of \( i \) is denoted by \( p_S(i) \).

Following is a generalization of Lemma A.1 for any sampling algorithm.

**Theorem B.1** Let \( X \) be a given input and let \( i \) be some item from \( \mathcal{I} \) having empirical probability \( p_X(i) \). Let \( A \) be a fair sampling algorithm implementing either sampling with replacement or sampling without replacement. If \( S \) is a random sample of size \( S \) obtained by \( A \) then

\[
\forall \varepsilon > 0 : \quad \Pr \{p_S(i) \geq (1 + \varepsilon)p_X(i)\} \leq \left( \frac{e^\varepsilon}{(1 + \varepsilon)^{1+\varepsilon}} \right)^{Sp_X(i)}
\]

\[
\forall \varepsilon < 0 : \quad \Pr \{p_S(i) \leq (1 + \varepsilon)p_X(i)\} \leq \left( \frac{e^\varepsilon}{(1 + \varepsilon)^{1+\varepsilon}} \right)^{Sp_X(i)}
\]

**Proof:** We show the result separately for sampling with replacement and sampling without replacement.

**Lemma B.2** Let \( X \) be a given input and let \( i \) be some item from \( \mathcal{I} \) having empirical probability \( p_X(i) \). Let \( A \) be a sampling algorithm implementing sampling with replacement. If \( S \) is a random sample of size \( S \) obtained by \( A \) then

\[
\forall \varepsilon > 0 : \quad \Pr \{p_S(i) \geq (1 + \varepsilon)p_X(i)\} \leq \left( \frac{e^\varepsilon}{(1 + \varepsilon)^{1+\varepsilon}} \right)^{Sp_X(i)}
\]

\[
\forall \varepsilon < 0 : \quad \Pr \{p_S(i) \leq (1 + \varepsilon)p_X(i)\} \leq \left( \frac{e^\varepsilon}{(1 + \varepsilon)^{1+\varepsilon}} \right)^{Sp_X(i)}
\]
**Proof:** The proof is by Chernoff Bound. Since \( p_S(i) \stackrel{\text{def}}{=} \frac{w_S(i)}{S} \), bounding the tail probability of \( w_S(i) \) is equivalent. Consider the set of \( S \) cells sampled. Since \( A \) performs sampling with replacement we can model it by an algorithm drawing \( S \) cells out of a hat (with replacement). Let \( \nu_\ell \) be an indicator random variable having value 1 if the cell picked at time \( \ell \) holds item \( i \). Note that \( w_S(i) = \sum_{\ell=1}^{S} \nu_\ell \). Note also that the expectation of \( w_S(i) \) equals \( S p_X(i) \). By Chernoff bound, we get the result for the probability that \( w_S(i) \) is within a factor \( 1 + \varepsilon \) of its expectation.

**Lemma B.3** Let \( X \) be a given input and let \( i \) be some item from \( I \) having empirical probability \( p_X(i) \). Let \( A \) be a sampling algorithm implementing sampling without replacement. If \( S \) is a random sample of size \( S \) obtained by \( A \) then

\[
\forall \varepsilon > 0 : \quad \Pr \{ p_S(i) \geq (1 + \varepsilon)p_X(i) \} \leq \left( \frac{e^\varepsilon}{(1 + \varepsilon)^{1+\varepsilon}} \right)^{S p_X(i)}
\]

\[
\forall \varepsilon < 0 : \quad \Pr \{ p_S(i) \leq (1 + \varepsilon)p_X(i) \} \leq \left( \frac{e^\varepsilon}{(1 + \varepsilon)^{1+\varepsilon}} \right)^{S p_X(i)}
\]

**Proof:** The first part of the proof is identical to the proof of lemma B.2 defining a set of indicator random variable \( \{ \nu_\ell | 1 \leq \ell \leq S \} \) having value 1 if the cell picked at time \( \ell \) holds item \( i \). Again we have \( w_S(i) = \sum_{\ell=1}^{S} \nu_\ell \) and \( E \left[ w_S(i) \right] = S p_X(i) \). Since \( \nu_\ell \) are dependent we cannot use Chernoff bound directly. We can however use a result of Panconesi and Srinivasan [21] to bound the tail probabilities; this requires showing that \( \nu_\ell \) are self-weakening random variables. Self-weakening are defined in [21] as variables for which

\[
E \left[ \prod_{k=1}^{K} \nu_{\ell_k} \right] \leq \lambda \prod_{k=1}^{K} E \left[ \nu_{\ell_k} \right] \quad \forall \{ \ell_1 < \ell_2 < \ldots < \ell_K \}.
\]

In our case it is straightforward to see that \( \Pr \{ \nu_{\ell_k} = 1 | \nu_{\ell_1} = \ldots = \nu_{\ell_{k-1}} = 1 \} < \Pr \{ \nu_{\ell_k} = 1 \} \) and it follows that \( \nu_\ell \) are self weakening with parameter \( \lambda = 1 \). The result of the lemma follows by [21].

The result of the theorem follows directly from the two lemmas above.
Appendix C

List of notation

Following are the main symbols used in this work.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>An algorithm</td>
</tr>
<tr>
<td>Algorithm $A$</td>
<td>Our sampling routine.</td>
</tr>
<tr>
<td>Algorithm BoundCount</td>
<td>An algorithm estimating the weight of an item.</td>
</tr>
<tr>
<td>Algorithm $B$</td>
<td>Our basic algorithm which approximates top-$k$ by a single sample.</td>
</tr>
<tr>
<td>Algorithm $R$</td>
<td>Our final algorithm.</td>
</tr>
<tr>
<td>Algorithm $S$</td>
<td>Our first iterative algorithm</td>
</tr>
<tr>
<td>$C_A$</td>
<td>Individual communication complexity of an algorithm $A$.</td>
</tr>
<tr>
<td>$c_A(X)$</td>
<td>Individual communication in bits used by $A$ for the instance $X$.</td>
</tr>
<tr>
<td>$I$</td>
<td>The number of candidate items in the system.</td>
</tr>
<tr>
<td>$i, j$</td>
<td>An item.</td>
</tr>
<tr>
<td>$k$</td>
<td>The number of items found by a top-$k$ query.</td>
</tr>
<tr>
<td>$n$</td>
<td>The number of nodes in a distributed system.</td>
</tr>
<tr>
<td>$p(i), p_X(i)$</td>
<td>The global frequency of item $i$ in $X$.</td>
</tr>
<tr>
<td>$p^<em>, p^</em>(X, k)$</td>
<td>The critical frequency. The frequency of the $k^{th}$ most-frequent item.</td>
</tr>
<tr>
<td>$S$</td>
<td>A sample of the input. $S$ denotes the sample size.</td>
</tr>
<tr>
<td>top($k, X$)</td>
<td>The set of $k$ items having largest global weight in $X$.</td>
</tr>
<tr>
<td>$u, v$</td>
<td>A node.</td>
</tr>
<tr>
<td>$v^*$</td>
<td>The root node.</td>
</tr>
<tr>
<td>$W$</td>
<td>The input size. Equals the total sum of all weights in $X$.</td>
</tr>
<tr>
<td>$w(i)$</td>
<td>The global weight if item $i$.</td>
</tr>
<tr>
<td>$X$</td>
<td>An instance. Consists of all local inputs.</td>
</tr>
<tr>
<td>$\delta$</td>
<td>An allowed error probability.</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>An allowed deviation for an approximation.</td>
</tr>
</tbody>
</table>