Distributed Computation of the Mode

Fabian Kuhn
Institute of Theoretical Computer Science
ETH Zurich
8092 Zurich, Switzerland
kuhn@inf.ethz.ch

Thomas Locher
Computer Engineering and Networks Laboratory (TIK)
ETH Zurich
8092 Zurich, Switzerland
lochert@tik.ee.ethz.ch

Stefan Schmid
Institut für Informatik
Technische Universität
München
85748 Garching, Germany
schmiste@in.tum.de

ABSTRACT

This paper studies the problem of computing the most frequent element (the mode) by means of a distributed algorithm where the elements are located at the nodes of a network. Let k denote the number of *distinct* elements and further let m_i be the number of occurrences of the element e_i in the ordered list of occurrences $m_1 > m_2 \geq ... \geq m_k$. We give a deterministic distributed algorithm with time complexity $O(D+k)$ where D denotes the diameter of the graph, which is essentially tight. As our main contribution, a Monte Carlo algorithm is presented which computes the mode in $O(D + F_2/m_1^2 \cdot \log k)$ time with high probability, where the frequency moment F_ℓ is defined as $F_\ell = \sum_{i=1}^{k} m_i^\ell$. This algorithm is substantially faster than the deterministic algorithm for various relevant frequency distributions. Moreover, we provide a lower bound of $\Omega(D + F_5/(m_1^5 B))$, where B is the maximum message size, that captures the effect of the frequency distribution on the time complexity to compute the mode.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]:
Nonnumerical Algorithms and Problems—*computations on discrete structures*;
G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*;
G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*

General Terms

Algorithms, Theory

Keywords

Aggregation, Distributed Algorithms, Mode, Most Frequent Element

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PODC'08, August 18–21, 2008, Toronto, Ontario, Canada.
Copyright 2008 ACM 978-1-59593-989-0/08/08 ...$5.00.

1. INTRODUCTION

A fundamental requirement of decentralized systems such as, e.g., wireless sensor networks or peer-to-peer networks, is that statistical data about these systems can be acquired in a distributed fashion. A network component may want to analyze relevant data which has been accumulated in the network in order to learn about the state of the system. For example, a participant in a peer-to-peer network might be interested in the most popular files or in the total number of available files. All queries must be performed efficiently as nodes may have limited resources such as a small energy supply in the case of sensor networks, or because the size of the network may forbid any operation that requires a single entity to communicate repeatedly with a large number of other entities in the system.

Fortunately, many aggregation functions can be computed in the network itself. Distributive (*max, min, count, sum*) and algebraic (*plus, minus, average, variance*) aggregation functions can be computed as follows [15]: In a first step, a spanning tree is constructed on which the aggregations are performed. The root sends a message along the spanning tree, asking the leaves to start the aggregation. The inner nodes of the spanning tree wait until all their children have sent their values, and subsequently forward the aggregated value to their respective parent. The time complexity of these operations is $O(D)$ where D denotes the diameter of the spanning tree. Even order statistics and percentiles can be computed efficiently by using a *k-selection algorithm* [10], which computes the k^{th} smallest element in the network.

Although a wide range of queries can be computed by combining these aggregation functions, there are essential queries that cannot be answered using any of these functions, e.g., "How many disjoint elements are there in the network?" or "Which element occurs most often among all elements?". While it has been shown that the number of disjoint elements can be approximated efficiently [6], less is known about the complexity of finding the *mode*, i.e., the element which occurs most often, in a distributed fashion. An algorithm to compute the mode distributively is a useful tool for popularity analyses in large networks.

This paper presents a deterministic algorithm which computes the mode in time $O(D + k)$ for general distributions, where D is the network diameter and k is the total number of distinct elements; this is essentially tight up to logarithmic factors. Our main result however is a distributed *Monte Carlo algorithm* to find the mode in general graphs. This algorithm is especially suited for skewed distributions which naturally arise in various contexts. For example, the frequencies of terms on webpages, files in file-sharing networks etc., are distributed according to a *power law* [2, 12, 17]. We point out that the time complexity of our algorithm is fairly low for such distributions, but our bound on the time complexity is more general and holds for all distributions. Most of our results are expressed in terms of the *frequency moments* $F_\ell = \sum_{i=1}^{k} m_i^\ell$, where m_i denotes the number of occurrences of the element e_i in the ordered list of occurrences, i.e., $m_1 > m_2 \geq \ldots \geq m_k$.[1] The proposed algorithm finds the mode in $O(D+F_2/m_1^2 \cdot \log k)$ time with probability at least $1 - 1/k^c$ for a constant $c \geq 1$. Moreover, we show that it is generally hard to compute the mode for arbitrary distributions by proving a lower bound of $\Omega(D+F_5/(m_1^5 B))$, where B is the maximum message size.

The remainder of this paper is organized as follows. In the subsequent section (Section 2), related work is reviewed. The model used in this work is introduced in Section 3. Our deterministic and randomized algorithms are both discussed in Section 4, together with their respective analyses. In Section 5, the aforementioned lower bound is proven. As an example for skewed distributions, we give the time complexity of the randomized algorithm in the case where the frequencies of the elements follow power laws in Section 6. Finally, the paper concludes in Section 7.

2. RELATED WORK

As mentioned before, while certain aggregation functions such as the *sum*, *minimum* or *maximum* etc. can be computed efficiently both in distributed and non-distributed settings, other functions are more difficult. In a recent paper by Kuhn *et al.* [10] a randomized algorithm for the k-selection problem, where the goal is to find the k^{th} smallest element, is presented. In a graph of diameter D consisting of n nodes, where each node holds a single element, the time complexity is $O(D \log_D n)$ with high probability, which matches the lower bound of $\Omega(D \log_D n)$ also derived in this work. Thus, finding the *median* is asymptotically more difficult than computing distributive and algebraic functions.

Flajolet *et al.* [6, 8] have studied the problem of determining the number of *distinct elements* in a multiset. In their probabilistic LogLog algorithm, all elements are hashed into sufficiently long binary strings in such a way that all bits closely resemble random, independent and uniformly distributed bits. As the number of elements that hash to a value with a prefix of x 0-bits is $k/2^x$ in expectation, where k denotes the number of distinct elements, the basic idea of the algorithm is that the length of the longest prefix consisting of 0-bits can be used as an approximation for $\log_2 k$. By repeating this step m times and by correcting a systematic bias in the asymptotic limit, the resulting estimate is asymptotically unbiased and the standard error is approximately $1.30/\sqrt{m}$. A simple reduction from the set disjointness problem shows that the bit complexity to find the true value k is at least $\Omega(k)$ [14]. This implies that, while finding the exact number of distinct elements is in general hard, an accurate estimate can be computed efficiently.

The problem of finding the *mode*, the most frequent element in a list for random access machines has been studied by Munro *et al.* [5, 13]. In [5], they give an algorithm which needs $n \log n/m + o(n \log n/m) + O(n)$ comparisons in the worst-case, where n is the total number of elements in the list and m is the frequency of the mode. This is asymptotically optimal up to lower order terms. Farzan *et al.* [7] investigate a *cache-oblivious model*, i.e., a random access memory model with a memory hierarchy where the cache sizes are unknown. The paper presents an optimal randomized algorithm and a near-optimal deterministic algorithm to compute the mode which minimize the number of cache misses.

Alon *et al.* [1] have studied the space complexity of approximating the frequency moments $F_\ell = \sum_{i=1}^{k} m_i^\ell$ in a *streaming model* (i.e., only one linear pass through the input is done). Note that the frequency moment $F_\infty = \lim_{\ell \to \infty} (F_\ell)^{1/\ell} = \max_{1 \leq i \leq k} m_i$ is the number of occurrences of the mode. In particular, in this work the lower bound technique used in [11] is adapted to show that for any fixed $k \geq 6$ and $\gamma < 1/2$, given an input sequence I of at most n elements taken from the set $N = \{1, 2, ..., n\}$, any randomized algorithm that outputs a number Z_k such that $\mathbb{P}[|Z_k - F_k| > 0.1 F_k] < \gamma$ uses at least$\Omega(n^{1-5/k})$ memory bits. This implies a lower bound of $\Omega(n)$ on the bit complexity to compute the frequency of the mode. It also follows that computing the mode itself requires $\Omega(n)$ bits, as the frequency of the mode can be computed trivially in $O(D)$ time once the mode is known.

While preparing the camera-ready version, we found a paper by Charikar, Chen, and Farach-Colton studying the space complexity of finding frequent elements in a streaming model [3]. Using a different algorithm but somewhat similar techniques, they give an algorithm with a space complexity of $O\big((k+F_2/m_k^2) \cdot \log n\big)$ that finds k elements with frequency at least $(1 - \varepsilon) m_k$ with high probability, where m_k is the frequency of the k^{th}-most frequent element. Together with some of the ideas of this paper, the techniques of [3] can be adapted for the distributed scenario yielding an alternative distributed algorithm with the same time complexity as the algorithm introduced in this paper. Similarly, the techniques of this paper could also be used to solve the problem of [3] with the same space complexity.

We provide a deterministic algorithm which is essentially asymptotically optimal for arbitrary frequency distributions. Furthermore, we derive general upper and lower bounds on the time complexity taking the frequency distribution of the elements into account.

3. MODEL

We are given a connected graph $G = (V, E)$ of diameter D with node set V and edge set E. The diameter of a graph is defined as the length of the longest shortest path between any two nodes. Each node $v \in V$ stores one or more elements e_i. There are K possible elements, i.e., each element e_i is chosen from an alphabet $\mathcal{A}$, where $|\mathcal{A}| = K$. For simplicity, we will assume that the alphabet is the set $\{1, ..., K\}$, henceforth denoted by $[K]$; this also implies that the elements can be ordered. Each element $e_i \in \mathcal{A}$ can appear multiple times, and we define m_i to be the frequency of the element e_i, where $m_1 > m_2 \geq ... \geq m_k$. We assume that the frequencies m_i are chosen from the set $[M]$. Let k denote the total number of distinct elements, and let $m = \sum_{i=1}^{k} m_i$ be the total number of elements. This paper will often refer to the following definition.

[1]If $m_1 = m_2$, the algorithm simply finds *any* element that occurs at least as often as all others.

Definition 3.1. **(Frequency Moments)** *The ℓ^{th} frequency moment F_ℓ of a multiset containing m_i elements of type $e_i \in [K]$ is defined as $F_\ell = \sum_{i=1}^{k} m_i^\ell$.*

Observe that $F_0 = k$ is the number of distinct elements in the multiset, and $F_1 = m$ is the total number of elements. For the sake of simplicity, when describing the algorithm we will assume that the nodes know the frequency moments F_0 and F_2 as well as the frequency m_1 of the mode. Estimates of these quantities can be obtained efficiently in parallel to the computation of the mode, as we will show in Section 4.3.

The nodes compute the mode by exchanging messages. Two nodes can directly send a message to each other if and only if they are connected by an edge in G. We consider a classic synchronous message passing model as, e.g., described in [15]. Our algorithms are described in terms of synchronous rounds. In every round, each node can receive a message from all adjacent nodes, perform some local computations, and send messages to all adjacent nodes. The time complexity of an algorithm is the number of rounds needed until every node terminates. Note that all our results also hold if we allow asynchronous communication since every synchronous algorithm can be reformulated as an asynchronous algorithm of the same time complexity [15]. We restrict the size of each message to $B \in O(\log K + \log M + \log n)$ bits. The main obstacle for computations in the described model is *edge congestion* that is caused by the bound on the size of the messages. In fact, with arbitrarily large messages, a single *convergecast*—a simple flooding-echo operation—would suffice to accumulate all elements at a single node, which could subsequently solve any problem locally.

Finally, in this paper it is assumed that a *breadth-first search spanning tree* rooted at the node initiating the algorithm has been pre-computed. As computing such a tree only takes $2D$ time and all our time bounds are at least linear in D, this assumption is not critical.

4. ALGORITHMS

First, the deterministic algorithm to compute the mode is presented, together with the proof of the lower bound for arbitrary frequency distributions. In the second subsection, the randomized algorithm $\mathcal{ALG}_{mode}$ whose running time crucially depends on the frequency distribution is described and analyzed. Finally, it is proven that $\mathcal{ALG}_{mode}$ has the same time complexity when a small quasi-random family of hash functions is used, which permits the use of small messages.

4.1 Deterministic Algorithm

There is a straightforward deterministic algorithm to find the mode executed on the pre-computed spanning tree. We assume that there is a total order on all the elements, i.e., $e_1 > \ldots > e_k$. The algorithm starts at the leaves of the tree which send element-frequency pairs $\langle e_i, m_i \rangle$ to their parents in increasing order starting with the smallest element that they possess. Any inner node v stores these pairs received from its children and sums up the frequencies for each distinct element. Node v forwards $\langle e_i, m_i \rangle$, where m_i is the accumulated frequency of e_i in the subtree rooted at v, to its parent as soon as v has received at least one pair $\langle e_j, m_j \rangle$ from each of its children such that $e_j \geq e_i$. Any node v sends $\langle e_i, m_i \rangle$ to its parent at time $t \leq h + i$ where h is the height of the subtree rooted at v. This claim clearly holds for the leaves as each leaf can send the i^{th} smallest element e_i at latest at time i. Inductively, a node v thus receives at least the i^{th} smallest element after $h+i-1$ time, after which it can forward the element including the accumulated frequency to its parent. Observe that there is no congestion, as node v has already sent all smaller element-frequency pairs in earlier rounds. Thus, the algorithm terminates after at most $O(D+k)$ steps. Note that this algorithm does not only compute the mode and its frequency m_1, but also the frequencies of all other elements.

Similarly to the lower bound for the number of distinct elements, a lower bound for the mode problem follows by reduction from the well-known *set disjointness problem.* It has been shown that two entities each holding a set of elements of cardinality $k/2$ must exchange $\Omega(k)$ bits in order to determine whether the two sets are disjoint, even using randomization [9, 11, 16]. This bit complexity implies a time lower bound of $\Omega(k/B)$, as by definition each message can contain at most B bits. Since there are distributions where determining if the sets are disjoint requires $\Omega(k/B)$ time and computing the mode for these distributions solves the disjointness problem,[2] it follows that computing the mode also requires $\Omega(k/B)$ time. As D is also a natural lower bound in the distributed model, the time lower bound for the mode problem is $\Omega(D + k/B)$.

Although this simple deterministic algorithm is optimal up to a factor B, it is worth investigating other algorithms which take the distribution of the elements into account. In particular, the *skewness* of the distribution can reasonably be expected to affect the efficiency of an algorithm to compute the mode. In the following, we present a randomized algorithm whose time complexity is considerably lower for various distributions.

4.2 Randomized Algorithm

In this section the randomized algorithm $\mathcal{ALG}_{mode}$ is introduced, followed by a thorough analysis of its time complexity. In order to compute the mode, $\mathcal{ALG}_{mode}$ makes extensive use of hash functions. Our analysis is organized in two parts. First, we assume that the hash functions are chosen independently and uniformly from the set of all possible hash functions mapping the elements $e_i \in [K]$ to a hash value in the required range $\mathcal{R}$. We require the size of the range $\mathcal{R}$ to be only 2; however, selecting any of these 2^K possible hash functions at random still entails a large communications overhead as choosing one hash function requires the communication of K bits. Therefore, we subsequently show that $\mathcal{ALG}_{mode}$ still works if a random hash function is selected from a much smaller set.

4.2.1 Algorithm

We first summarize the basic mechanism underlying the algorithm $\mathcal{ALG}_{mode}$. Each node in the graph stores a local counter $c(e_i)$, which is initially 0, for each of its t elements $e_1, \ldots, e_t$. The algorithm uses hash functions that map each element randomly to one of two bins with equal probability. All nodes use the same hash function to compute and subsequently forward the number of its elements that mapped to the first and the second bin. Each counter $c(e_i)$ is incremented by the number of elements that have been mapped

[2]If the frequency of the mode is 2, then this element must appear in both sets and thus the sets are not disjoint.

Algorithm 1 *countElementsInBins($\mathfrak{h}$)* at node v with multiset $\langle e_1, \ldots, e_t \rangle$:

```
1: $c_0 = |\{e_i : \mathfrak{h}(e_i) = -1\}|$
2: $c_1 = |\{e_i : \mathfrak{h}(e_i) = +1\}|$
3: if $\Gamma(v) = \emptyset$ then
4:     send $\langle c_0, c_1 \rangle$ to $p(v)$
5: else
6:     for all $v_j \in \Gamma(v)$ in parallel do
7:         $\langle c_0^{(j)}, c_1^{(j)} \rangle$ = countElementsInBins($\mathfrak{h}$)
8:     send $\langle c_0, c_1 \rangle + \sum_{j \in \Gamma(v)} \langle c_0^{(j)}, c_1^{(j)} \rangle$ to $p(v)$
```

to the same bin as element e_i. The idea is to determine the mode by repeating this procedure using different hash functions. Since the mode is likely to end up in the larger bin more often than the other elements, the counter $c(e_1)$ will grow faster than the counters of the other elements. After a first phase, which reduces the set of candidates for the mode, the frequency of each remaining candidate is computed separately in a second phase. The time complexity is bounded by the time required to find a small set of candidates and by the time to check these candidates.

We will now study each step of the algorithm in greater detail. The root node, i.e., the node interested in computing the mode, selects r_1 hash functions $\mathfrak{h}_1, \ldots, \mathfrak{h}_{r_1}$ where $\mathfrak{h}_i : \mathcal{A} \rightarrow \{-1, +1\}$, i.e., each hash function maps every element to one of the two bins. The parameter r_1 will be determined later in the analysis of the algorithm. In the following, we will represent the two bins as a tuple $\langle c_0, c_1 \rangle$, where c_i denotes the number of elements that have been mapped to bin $i \in \{0, 1\}$. All nodes then accumulate the mappings by means of a flooding-echo procedure on the spanning tree using the function *countElementsInBins* parameterized by the hash function: Once the leaves have received information about the hash function, their elements $\langle e_1, \ldots, e_t \rangle$ are hashed and added to the two bins, i.e., c_0 is set to the number of elements that mapped to the first bin and c_1 is set to the number of the remaining elements. This tuple is sent to the parent node, which accumulates the tuples from all its children and adds its own tuple. The resulting tuple is forwarded recursively up to the root. Let $p(v)$ and $\Gamma(v)$ denote the parent and the set of children of node v in the spanning tree, respectively. The sum of two tuples $\langle c_0', c_1' \rangle$ and $\langle c_0'', c_1'' \rangle$ is defined as $\langle c_0, c_1 \rangle$, where $c_i = c_i' + c_i''$ for $i \in \{0, 1\}$. This subroutine is summarized in Algorithm 1.

Once the root has computed the final tuple $\langle c_0, c_1 \rangle$, this tuple is distributed down the spanning tree. Any node that receives this *distribute* message forwards it to its children and updates its local counters according to the following rule: For all elements e_i that mapped to the larger of the two bins, its counter $c(e_i)$ is increased by $|c_0 - c_1|$. These steps can be carried out *in parallel* for all r_1 hash functions, i.e., the root can issue one of the r_1 procedure calls in each communication round. Once the r_1 results have been obtained and the tuples have all been distributed, Phase (1) of the algorithm is completed.

In the second phase, the r_2 elements—the parameter r_2 will also be specified later—with the largest counters are accumulated at the root using the procedure *getPotentialModes*. In this procedure, the nodes always forward the element e_i, including $c(e_i)$, if its counter is the largest among all those whose element has not been sent yet. Moreover,

Algorithm 2 $\mathcal{ALG}_{mode}$

```
1: mode = $\emptyset$, freq = $-\infty$
2: Phase (1):
3: for $i = 1, \ldots, r_1$ in parallel do
4:     $\langle c_0, c_1 \rangle$ = countElementsInBins($\mathfrak{h}_i$)
5:     distribute($\langle \langle c_0, c_1 \rangle, \mathfrak{h}_i \rangle$)
6: Phase (2):
7: $\langle e_1, \ldots, e_{r_2} \rangle$ = getPotentialModes($r_2$)
8: for $i = 1, \ldots, r_2$ in parallel do
9:     $m_i$ = getFrequency($e_i$)
10:    if $m_i >$ freq then
11:        mode = $e_i$, freq = $m_i$
12: return mode
```

an element is only forwarded, if its counter is among the r_2 largest counters ever forwarded to the parent node. Once these elements arrive at the root, the root issues a request to count the individual frequencies for all those elements and the element with the largest frequency is returned as the mode. The entire algorithm is depicted in Algorithm 2.

We will now specify the parameters and analyze the time complexity of $\mathcal{ALG}_{mode}$.

4.2.2 Analysis

In Phase (1), $\mathcal{ALG}_{mode}$ executes r_1 iterations, where a randomly chosen hash function $\mathfrak{h}_i \in \{\mathfrak{h}_1, \mathfrak{h}_2, ..., \mathfrak{h}_{r_1}\}$ assigns all the elements $e_i \in \mathcal{A}$ to one of the two bins in each iteration, i.e., $\mathfrak{h}_1, ..., \mathfrak{h}_{r_1} : \mathcal{A} \rightarrow \{-1, +1\}$. First, we need to prove a bound on the number r_1 of required hash functions to substantially reduce the set of candidates.

Throughout the rest of this section, we will make use of the following helper lemma.

LEMMA 4.1. *For $i = 1, \ldots, k$, let Y_i be independent random variables with*

$$Y_i = \begin{cases} y_i, & \textit{with} \quad \mathbb{P} = 1/2 \\ -y_i, & \textit{with} \quad \mathbb{P} = 1/2 \end{cases}$$

If $Y = \sum_{i=1}^k Y_i$ and $F_2[Y] = \sum_{i=1}^k y_i^2$ is the second frequency moment of a set with frequencies $y_1, \ldots, y_k$, then we have that

$$\mathbb{P}\left[Y \geq \lambda\sqrt{F_2[Y]}\right] \leq e^{-\lambda^2/2}.$$

PROOF. It holds that

$$\begin{aligned}
\mathbb{P}\left[Y \geq \lambda\sqrt{F_2[Y]}\right] &\underset{\gamma>0}{\leq} \frac{\mathbb{E}[e^{\gamma Y}]}{e^{\gamma\lambda\sqrt{F_2[Y]}}} = \frac{\prod_{i=1}^k \mathbb{E}[e^{\gamma Y_i}]}{e^{\gamma\lambda\sqrt{F_2[Y]}}} \\
&\leq \frac{\prod_{i=1}^k \frac{e^{\gamma y_i} + e^{-\gamma y_i}}{2}}{e^{\gamma\lambda\sqrt{F_2[Y]}}} \\
&= \frac{\prod_{i=1}^k \cosh(\gamma y_i)}{e^{\gamma\lambda\sqrt{F_2[Y]}}} \\
&\leq \frac{\prod_{i=1}^k e^{\gamma^2 y_i^2/2}}{e^{\gamma\lambda\sqrt{F_2[Y]}}} = \frac{e^{\gamma^2 \sum_{i=1}^k y_i^2/2}}{e^{\gamma\lambda\sqrt{F_2[Y]}}} \\
&\leq e^{-\lambda^2/2}
\end{aligned}$$

since $(e^x + e^{-x})/2 = \cosh(x) \leq e^{x^2/2}$ and by setting $\gamma = \lambda/\sqrt{F_2[Y]}$. Note that $F_2[Y] = \mathrm{Var}(Y)$. □

The goal of Phase (1) is to reduce the set of elements that could potentially be the mode to a small set of size r_2. The following lemma bounds the number r_1 of hash functions required to ensure that the counter of the mode is larger than the counter of a large fraction of all elements.

Lemma 4.2. *If $r_1 \in O(F_2/m_1^2 \log(k/\varepsilon))$ then $\forall e_i : m_i < m_1/2$ it holds that $c(e_i) < c(e_1)$ with probability at least $1-\varepsilon$.*

Proof. First, we focus only on the events where the mode e_1 and the element e' with the maximum frequency among all elements whose frequency is less than half of the frequency e_1 of the mode are put into different bins. All other elements are added randomly to one of the bins, and this procedure is repeated r_1 times. Alternatively, we can say that there are $(k-2)r_1$ elements $\alpha_1, \ldots, \alpha_{(k-2)r_1}$ that are placed randomly into the two bins. It holds that $\sum_{i=1}^{r_1(k-2)} \alpha_i^2 < r_1 \cdot F_2$. Before the elements $\alpha_1, \ldots, \alpha_{r_1(k-2)}$ are put into the bins, it holds that $c(e_1) > c(e') + r_1 \cdot m_1/2$. In order to ensure that $c(e_1) > c(e')$ after all elements have been placed in one of the bins, the probability that the other elements compensate this imbalance of at least $r_1 \cdot m_1/2$ must be small. Let the Bernoulli variable Z_i indicate into which bin the element α_i is placed. In particular, in case $Z_i = -1$, the element is put into the bin where the mode is, and if $Z_i = 1$, the element is placed into the other bin. By setting $r_1 = 8F_2/m_1^2 \ln(2k/\varepsilon)$ and applying Lemma 4.1 we get that

$$\mathbb{P}\left[\sum_{i=1}^{(k-2)r_1} \alpha_i Z_i \geq r_1 \cdot m_1/2\right] < e^{-\frac{r_1^2(\frac{m_1}{2})^2}{2\sum_{i=1}^{(k-2)r_1} \alpha_i^2}}$$

$$< e^{-\frac{r_1 m_1^2}{8F_2}} = \frac{\varepsilon}{2k}.$$

In order to ensure that the elements e_1 and e' are often placed into different bins, the number of rounds is increased to $r_1 = 32F_2/m_1^2 \ln(2k/\varepsilon)$. Let the random variable $\mathcal{U}$ denote the number of times that the two elements are placed into the same bin. Using a simple Chernoff bound we get that the probability that $32F_2/m_1^2 \ln(2k/\epsilon)$ rounds do not suffice to bound the probability of failure to $\varepsilon/(2k)$, because e_1 and e' are put into different bins less than $8F_2/m_1^2 \ln(2k/\varepsilon)$ times, is itself bounded by $\mathbb{P}[\mathcal{U} > 24F_2/m_1^2 \ln(2k/\varepsilon)] < e^{-F_2/m_1^2 \ln(2k/\varepsilon)} < \varepsilon/(2k)$. Thus, the probability that $c(e_1) > c(e')$ is at least $1 - \varepsilon/k$. Let $\Upsilon = \{e_i : m_i < m_1/2\}$ denote the set of all elements for which we want to prove that their counters are lower than the counter of the mode. The probability that any element e^* in this set has a counter larger than the mode is $\mathbb{P}[\exists e^* \in \Upsilon : c(e^*) > c(e_1)] < \sum_{e \in \Upsilon} \mathbb{P}[c(e) > c(e_1)] < k \cdot (\varepsilon/k) = \varepsilon$, which concludes the proof. $\square$

Note that technically we cannot determine r_1 unless we know F_2, m_1, and k. In the following section, we show that these quantities can all be estimated efficiently. Using this bound on the number of hash functions, we are now in the position to prove the following theorem.

Theorem 4.3. *The time complexity of $\mathcal{ALG}_{mode}$ to compute the mode with probability at least $1-\varepsilon$ on an arbitrary graph G of diameter D is*

$$O\left(D + \frac{F_2}{m_1^2} \log \frac{k}{\varepsilon}\right).$$

Proof. In Phase (1) of the algorithm, r_1 hash functions are applied to all elements and the sum of elements hashed to each bin is accumulated at the root using a simple flooding-echo procedure. It is important to see that all these hash functions can be handled *in parallel* as opposed to computing the resulting bin sizes for each hash function sequentially, i.e., the number of communication rounds required is bounded by $O(D + r_1)$ and not $O(D \cdot r_1)$. Each result is distributed back down the tree in order to allow each node to updates its counters for all its stored elements, which requires $O(D)$ time. Hence, the time complexity of the first phase is bounded by $O(D + r_1)$.

In Phase (2), the r_2 elements with the largest counters are accumulated at the root, and the element with the highest number of occurrences out of this set is returned as the mode. The procedure *getPotentialModes* performs this operation in $O(D + r_2)$ time, as the i^{th} largest value arrives at the root after at most $D + i$ rounds of communication. Naturally, the frequency of r_2 elements can also be determined in $O(D + r_2)$ time, and thus the entire phase requires $O(D + r_2)$ time.

The parameter r_2 has to be large enough to ensure that the mode is in fact in this set of elements with high probability. Let $\bar{\Upsilon} = \{e_i : m_i \geq m_1/2\}$ be the complement of Υ. According to Lemma 4.2, if $r_1 = 32F_2/m_1^2 \ln(2k/\varepsilon)$, then the mode is in $\bar{\Upsilon}$ with probability at least $1 - \varepsilon$. We have that $|\bar{\Upsilon}|(m_1/2)^2 \leq \sum_{e_i \in \bar{\Upsilon}} m_i^2 \leq F_2$, implying that $|\bar{\Upsilon}| \leq 4F_2/m_1^2$. Thus, by setting $r_2 = 4F_2/m_1^2$, both phases complete after $O(D + F_2/m_1^2 \log(k/\varepsilon))$ rounds and the mode is found with probability $1 - \varepsilon$. $\square$

4.3 Estimators for F_2, m_1, and k

In order to estimate F_2, the algorithm described by Alon et al. [1] can be used, which employs a set of *four-wise independent* hash functions mapping elements to either -1 or 1. A set $\mathcal{H}$ of hash functions is called four-wise independent if for any four elements $e_1, \ldots, e_4$, the values $\mathfrak{h}(e_i)$ of a randomly chosen hash function $\mathfrak{h}$ from the set are statistically independent. Hence, for any elements $e_1, \ldots, e_4$ and any choice $c_1 \ldots, c_4 \in \{-1, 1\}$ there are $|\mathcal{H}|/16$ hash functions $\mathfrak{h} \in \mathcal{H}$ for which it holds that $\mathfrak{h}(e_i) = c_i$ for all $i = \{1 \ldots, 4\}$. It is shown that, by using $s := 32\frac{\lg(1/\varepsilon')}{\lambda^2}$ of these hash functions $\mathfrak{h}_1, \ldots, \mathfrak{h}_s$, where each function $\mathfrak{h}_j$ is used to compute $X_j := \sum_{i=1}^{k} \mathfrak{h}_j(e_i) \cdot m_i$, these values X_j can be used to compute an estimate $\hat{F}_2$ which deviates from F_2 by at most λF_2 with probability $1 - \varepsilon'$. The transition from the streaming model to our model is straightforward: The values X_j can be computed by usig two counters c_{-1} and c_{+1} that sum up the values that map to -1 and $+1$ (just like in our algorithm), as it holds that $X_j = \sum_{i=1}^{k} \mathfrak{h}_j(e_i) \cdot m_i = c_{+1} - c_{-1}$. Since λ is a constant, we can aggregate all counters and thus all values X_j for all hash functions $j \in \{1, \ldots, s\}$ at the node that wishes to approximate F_2 in $O(D + \log(1/\varepsilon'))$ time and compute the estimate $\hat{F}_2$ locally.

As mentioned in Section 2, there is a probabilistic algorithm to estimate $k = F_0$ efficiently [6, 8]. The algorithm uses hash functions that map elements to bit strings. The key idea is that the length of the longest prefix consisting of 0-bits can be used as an approximation for $\log_2 k$. Several runs of this procedure using randomly chosen hash functions are used in order to bound the variance. In the distributed version of this algorithm, the lengths of the longest prefix of

0-bits for each hash function are accumulated and then used to compute an estimate $\hat{k}$, which also takes $O(D)$ time.[3]

Thus, estimators $\hat{F}_2$ and $\hat{k}$ for both F_2 and k can be computed in parallel to the computation of the mode. Since the variances are bounded, we can in fact compute estimates for which it holds that $\hat{F}_2 \geq F_2$ and $\hat{k} \geq k$ with a certain probability $1-\varepsilon''$ by modifying the original estimates. Given the estimators $\hat{F}_2$ and $\hat{k}$, an estimator $\hat{m}_1$ for m_1 can be obtained as follows. We know that after $r_1 = 32F_2/m_1^2 \log(2k/\varepsilon)$ time, the first phase of the algorithm may terminate. After each distribution of $\langle\langle c_0, c_1\rangle, \mathfrak{h}_i\rangle$ we determine the frequency of any element e_i whose counter is currently the largest in $O(D)$ time and use this frequency as the new estimator for $\hat{m}_1$, if this frequency is greater than any frequency encountered before. Aggregation rounds are performed iteratively as long as $32\hat{F}_2/\hat{m}_1^2 \log 2\hat{k}/\varepsilon < T$, where T denotes the number of aggregation rounds executed so far. Once this inequality does no longer hold, we can conclude that the algorithm must have executed enough aggregation rounds, as $\hat{m}_1 \leq m_1$. The algorithm does not run much longer than needed, since $\hat{m}_1 \geq m_1/2$ after $32\hat{F}_2/\hat{m}_1^2 \log(2\hat{k}/\varepsilon) \geq r_1$ rounds.

Note that, since all algorithms exhibit a certain error probability, the number of rounds r_1 of the first phase must be increased slightly by a small constant factor to ensure that the mode is still computed correctly with probability at least $1-\varepsilon$. We dispense with the analysis of the exact factor.

4.4 Selecting Hash Functions

$\mathcal{ALG}_{mode}$ makes extensive use of hash functions. In Section 4.2, we assumed that the hash functions can be chosen uniformly at random from the set of all 2^K possible hash functions. However, in order to select one of these hash functions at random, K bits have to be communicated, which is not allowed in our model where we restrict message sizes to at most $O(\log K + \log M + \log n)$ bits. In the following, we show that our algorithm still works if a random hash function is chosen from a much smaller set.

We first need a few definitions. Let $S \subseteq [K] \times [M]$ be a subset of all possible element-frequency pairs and let $\mathfrak{h} : [K] \to \{-1, +1\}$ be a hash function. We define the *imbalance* $\lambda_S(\mathfrak{h})$ of $\mathfrak{h}$ with respect to S as

$$\lambda_S(\mathfrak{h}) = \frac{\sum_{(e_i,m_i)\in S} \mathfrak{h}(e_i)\cdot m_i}{\sqrt{F_2[S]}}, \qquad (1)$$

where $F_2[S]$ is the second frequency moment of the set S. We call a hash function $\mathfrak{h}$ *λ-good* with respect to a set $S \in [K] \times [M]$ if $|\lambda_S(\mathfrak{h})| \leq \lambda$.

Let $\mathcal{H}$ be a family of hash functions $\mathfrak{h} : [K] \to \{-1, 1\}$. Further, let $\mathcal{H}_{e,e'} \subseteq \mathcal{H}$ be the set of hash functions $\mathfrak{h}$ for which $\mathfrak{h}(e) \neq \mathfrak{h}(e')$. We call $\mathcal{H}$ a *quasi-random* hash family with parameters $\delta, \varepsilon \in (0,1)$ if the following conditions hold. Let $\ell_1 = \lceil\sqrt{2\ln(5(2+\delta)K^2/\varepsilon)}/\delta\rceil$.

(I) **2-independence:** For all $e, e' \in [K]$, $|\mathcal{H}_{e,e'}| \geq (1-\delta)\cdot|\mathcal{H}|/2$.

(II) **Exponentially small imbalance:** For all $e, e' \in [K]$, $S \subseteq ([K]\setminus\{e,e'\}) \times [M]$, and all integers $\ell \leq \ell_1$, the number of hash functions $\mathfrak{h} \in \mathcal{H}_{e,e'}$ that are not $\delta\ell$-good with respect to S is at most $(2+\delta)\cdot e^{-(\delta\ell)^2/2}$.

(III) **Independence of imbalance sign:** For all $e, e' \in [K]$, $S \subseteq ([K]\setminus\{e,e'\}) \times [M]$, and all integers $\ell \leq \ell_1$, let $\Lambda_{e,e',S}(\ell)$ be the set of hash functions $\mathfrak{h} \in \mathcal{H}_{e,e'}$ that are not $(\delta\ell-1)$-good but that are $\delta\ell$-good with respect to S. We have

$$\begin{aligned} &\Big| \big|\{\mathfrak{h} \in \Lambda_{e,e',S}(\ell) \,|\, \lambda_S(\mathfrak{h})\cdot\mathfrak{h}(e) < 0\}\big| - \\ &\quad \big|\{\mathfrak{h} \in \Lambda_{e,e',S}(\ell) \,|\, \lambda_S(\mathfrak{h})\cdot\mathfrak{h}(e) > 0\}\big| \Big| \\ \leq &\quad \delta\cdot\left(e^{-\delta^2(\ell-1)^2/2} - e^{-\delta^2\ell^2/2}\right)\cdot|\mathcal{H}_{e,e'}|. \end{aligned}$$

Note that the family of all 2^K hash functions clearly is a quasi-random family for all δ and ε (cf. Lemma 4.1). However, we will show that there are exponentially smaller quasi-random hash families and that our algorithm achieves the same bounds if the hash functions $\mathfrak{h}_1, \ldots, \mathfrak{h}_{r_1}$ are chosen uniformly from any quasi-random family $\mathcal{H}$ with appropriate values of δ and ε. We proceed as follows: We first show that choosing $O(F_2/m_1^2 \cdot \log(k/\varepsilon))$ hash functions from a quasi-random family results in a time complexity of $O(D + F_2/m_1^2 \cdot \log(k/\varepsilon))$ rounds if the algorithm is allowed to err with probability at most ε. Subsequently, we prove that quasi-random hash families of size $O(\text{poly}(K, M))$ exist, allowing to run $\mathcal{ALG}_{mode}$ with messages of size $O(\log M + \log K + \log n)$.

Assume that we are given a set $S \subseteq [K] \times [M]$ of elements $e_1, \ldots, e_k \in [K]$ with frequencies $m_1 \geq \ldots \geq m_k \in [M]$ and a quasi-random hash family $\mathcal{H}$. The hash functions $\mathfrak{h}_1, \ldots, \mathfrak{h}_{r_1}$ in $\mathcal{ALG}_{mode}$ are chosen independently and uniformly at random from $\mathcal{H}$. Let e_i be an element that occurs less than half as often as the mode, i.e., $m_i < m_1/2$. As above, let $\mathcal{H}_{e_1,e_i} \subseteq \mathcal{H}$ be the set of hash functions $\mathfrak{h}$ for which $\mathfrak{h}(e_1) \neq \mathfrak{h}(e_i)$. As in Section 4.2 for random hash functions, we again need to choose the number of rounds r_1 such that $c(e_1) > c(e_i)$ with sufficiently large probability. Let $S_{e_i} = S \setminus \{(e_1, m_1), (e_i, m_i)\}$ and let $\mathcal{H}_{e_i} = \mathcal{H}_{e_1,e_i} \cap \{\mathfrak{h}_1, \ldots, \mathfrak{h}_{r_1}\}$ be the set of chosen hash functions $\mathfrak{h}$ for which $\mathfrak{h}(e_1) \neq \mathfrak{h}(e_i)$. The difference Δ_{e_i} between $c(e_1)$ and $c(e_i)$ after r_1 rounds can be computed as

$$\Delta_{e_i} = |\mathcal{H}_{e_i}|\cdot(m_1 - m_i) + N_{e_i}, \qquad (2)$$

where

$$N_{e_i} = \sum_{\mathfrak{h}\in\mathcal{H}_{e_i}} \mathfrak{h}(e_1)\cdot \sum_{(e,m)\in S_{e_i}} \mathfrak{h}(e)\cdot m = \sum_{\mathfrak{h}\in\mathcal{H}_{e_i}} \mathfrak{h}(e_1)\cdot\lambda_{S_{e_i}}(\mathfrak{h})\cdot\sqrt{F_2[S_{e_i}]}.$$

Thus, we have to show that $N_{e_i} > -|\mathcal{H}_{e_i}|(m_1 - m_i) > -|\mathcal{H}_{e_i}|\cdot m_1/2$ with sufficiently high probability. We need to start with some additional definitions. Recall that $\ell_1 = \lceil\sqrt{2\ln(5(2+\delta)K^2/\varepsilon)}/\delta\rceil$. We partition the hash functions $\mathfrak{h} \in \mathcal{H}_{e_1,e_i}$ according to the value of $\mathfrak{h}(e_1)\lambda_{S_i}(\mathfrak{h})$. For integers $1 \leq \ell \leq \ell_1$, let $\Lambda^+(\ell)$ be the set of hash functions $\mathfrak{h} \in \mathcal{H}_{e_1,e_i}$ for which $\mathfrak{h}(e_1)\lambda_{S_i}(\mathfrak{h}) \in (\delta(\ell-1), \delta\ell]$ and let $\Lambda^-(\ell)$ be the set hash functions $\mathfrak{h} \in \mathcal{H}_{e_1,e_i}$ for which $\mathfrak{h}(e_1)\lambda_{S_i}(\mathfrak{h}) \in [-\delta\ell, -\delta(\ell-1))$. Further let Λ_∞ be the set of hash functions $\mathfrak{h} \in \mathcal{H}_{e_1,e_i}$ for which $|\lambda_{S_i}(\mathfrak{h})| > \delta\ell_1$. In order to lower bound the value of N_{e_i}, we need the following lemma.

[3]It has been pointed out that the somewhat ideal properties of the hash functions assumed in the original work are not required and that by slightly modifying the algorithm it suffices to use a set of linear hash functions [1].

LEMMA 4.4. *For every constant $c_2 > 1$, there is a constant $c_1 > 0$, such that*

$$\mathbb{P}\left[\sum_{\mathfrak{h}\in\mathcal{H}_{e_i}\setminus\Lambda_\infty} \lambda^2_{S_{e_i}}(\mathfrak{h}) \;>\; c_1\cdot\left|\mathcal{H}_{e_i}\right|\right] \;<\; c_2^{-|\mathcal{H}_{e_i}|}.$$

PROOF. For simplicity, assume that $1/\delta \in \mathbb{Z}$ such that every integer $\lambda = \delta\ell$ for some integer ℓ. Then by using Condition (II) of the quasi-randomness definition, we obtain that for every $\lambda \in \mathbb{N}$, the number of hash functions $\mathfrak{h} \in \mathcal{H}_{e_1,e_i}$ that is not λ-good is at most $(2+\delta)\cdot|\mathcal{H}_{e_1,e_i}|\cdot e^{-\lambda^2/2}$. The probability that we want to bound is maximized if for all $\lambda\in\mathbb{N}$ we have that

$$\begin{aligned}\beta_\lambda &= \frac{\left|\{\mathfrak{h}\in\mathcal{H}_{e_1,e_i}|\mathfrak{h}\text{ is not }\lambda\text{-good}\}\right|}{\left|\mathcal{H}_{e_1,e_i}\right|}\\ &= \begin{cases}(2+\delta)\cdot e^{-\lambda^2/2} & \text{if } \lambda\geq\lceil\sqrt{2\ln(2+\delta)}\rceil\\ 1 & \text{otherwise.}\end{cases}\end{aligned}$$

Let $\mathcal{H}' = \mathcal{H}_{e_1,e_i}\setminus\Lambda_\infty$ and $t = |\mathcal{H}'|$, and let

$$p := \mathbb{P}\left[\sum_{\mathfrak{h}\in\mathcal{H}'}\lambda^2_{S_{e_i}}(\mathfrak{h}) > c_1'\cdot t\right]$$

We have

$$\begin{aligned}p &\leq \mathbb{P}\left[\sum_{\mathfrak{h}\in\mathcal{H}'}\left\lceil|\lambda_{S_{e_i}}(\mathfrak{h})|\right\rceil^2 > c_1'\cdot t\right]\\ &\underset{\gamma>0}{<} \frac{\mathbb{E}\left[e^{\gamma\sum_{\mathfrak{h}\in\mathcal{H}'}\lceil|\lambda_{S_{e_i}}(\mathfrak{h})|\rceil^2}\right]}{e^{\gamma c_1' t}}\\ &= \frac{\prod_{\mathfrak{h}\in\mathcal{H}'}\mathbb{E}\left[e^{\gamma\lceil|\lambda_{S_{e_i}}(\mathfrak{h})|\rceil^2}\right]}{e^{\gamma c_1' t}}\\ &\leq \left(\frac{\sum_{\lambda=1}^\infty(\beta_\lambda-\beta_{\lambda-1})\cdot e^{\gamma\lambda^2}}{e^{\gamma c_1'}}\right)^t\\ &\leq \left(\frac{(2+\delta)\cdot\sum_{\lambda=1}^\infty\left(e^{-(\lambda-1)^2/2}-e^{-\lambda^2/2}\right)\cdot e^{\gamma\lambda^2}}{e^{\gamma c_1'}}\right)^t\\ &= \left(\frac{(2+\delta)\cdot\sum_{\lambda=1}^\infty e^{\lambda^2(\gamma-1/2)}\cdot\left(e^{\lambda-1/2}-1\right)}{e^{\gamma c_1'}}\right)^t.\end{aligned}$$

There are constants c_1' and γ such that the above expression is at most c_2^{-t} for a constant $c_2 > 1$. The claim now follows because for a hash function $\mathfrak{h}$ that is chosen uniformly at random from $\mathcal{H}_{e_1,e_i}$, $\mathbb{P}[\mathfrak{h}\in\Lambda_\infty] \leq \varepsilon/(5K^2)$ and therefore $t = \Omega(|\mathcal{H}_{e_i}|)$ with probability $e^{-\Theta(|\mathcal{H}_{e_1,e_i}|)}$. □

We can now bound the value of N_{e_i} with high probability.

LEMMA 4.5. *If $\delta < 1/(8c\cdot K\cdot\ln^{3/2}(K/\varepsilon))$ and $|\mathcal{H}_{e_i}| \geq c\cdot F_2[S_{e_i}]/m_1^2\cdot\ln(k/\varepsilon)$ for a sufficiently large constant $c\geq 1$, we get*

$$\mathbb{P}\left[N_{e_i} \leq -\frac{m_1}{2}\cdot|\mathcal{H}_{e_i}|\right] \;<\; \frac{\varepsilon}{k}.$$

PROOF. Note that $K \geq k > F_2[S_{e_i}]/m_1^2$. In order to simplify the analysis, we define $\tilde{\lambda}_{S_{e_i}}(\mathfrak{h}) = \delta(\ell-1)$ for every $\mathfrak{h}\in\Lambda^+(\ell)$ and $\tilde{\lambda}_{S_{e_i}}(\mathfrak{h}) = -\delta(\ell-1)$ for every $\mathfrak{h}\in\Lambda^-(\ell)$. Instead of N_{e_i}, we now consider the random variable

$$\tilde{N}_{e_i} \;=\; \sum_{\mathfrak{h}\in\mathcal{H}_{e_i}}\mathfrak{h}(e_1)\cdot\tilde{\lambda}_{S_{e_i}}(\mathfrak{h})\cdot\sqrt{F_2[S_{e_i}]} \qquad (3)$$

and get

$$N_{e_i} > \tilde{N}_{e_i} - \delta\cdot|\mathcal{H}_{e_i}|\cdot\sqrt{F_2[S_{e_i}]},$$

since we change no $\lambda_{S_{e_i}}(\mathfrak{h})$-value by more than δ. We set $d(\ell) = \left||\Lambda^-(\ell)| - |\Lambda^+(\ell)\right|$ to the difference between the sizes of two symmetric sets of the described partition of $\mathcal{H}_{e_1,e_i}$. Since $\mathcal{H}$ is a quasi-random family, we have $d(\ell) \leq \delta(e^{-\delta^2(\ell-1)^2/2} - e^{-\delta^2\ell^2/2})|\mathcal{H}_{e_1,e_i}|$ (independence of imbalance sign). For each $1\leq\ell\leq\ell_1$, remove $d(\ell)$ hash functions from the larger of the two sets $\Lambda^+(\ell)$ and $\Lambda^-(\ell)$ and add them to a set Λ_ℓ. We then have $|\Lambda^+(\ell)| = |\Lambda^-(\ell)|$ for $1\leq\ell\leq\ell_1$. Let $\mathcal{H}_1 = \bigcup_{\ell=1}^{\ell_1}\Lambda^+(\ell)\cup\Lambda^-(\ell)$ be the hash functions in some $\Lambda^+(\ell)$ or $\Lambda^-(\ell)$ and let $\mathcal{H}_2 = \mathcal{H}_{e_1,e_i}\setminus(\mathcal{H}_1\cup\Lambda_\infty)$ be the set of hash functions from the sets Λ_ℓ. We define

$$X_{e_i} = \sum_{\mathfrak{h}\in\mathcal{H}_1\cap\mathcal{H}_{e_i}}\mathfrak{h}(e_1)\cdot\tilde{\lambda}_{S_{e_i}}(\mathfrak{h})$$

$$Y_{e_i} = \sum_{\mathfrak{h}\in\mathcal{H}_2\cap\mathcal{H}_{e_i}}\mathfrak{h}(e_1)\cdot\tilde{\lambda}_{S_{e_i}}(\mathfrak{h})$$

$$Z_{e_i} = \sum_{\mathfrak{h}\in\Lambda_\infty\cap\mathcal{H}_{e_i}}\mathfrak{h}(e_1)\cdot\lambda_{S_{e_i}}(\mathfrak{h})\;,$$

and then have $\tilde{N}_{e_i} = (X_{e_i}+Y_{e_i}+Z_{e_i})\cdot\sqrt{F_2[S_{e_i}]}$. Because $|\Lambda^+(\ell)| = |\Lambda^-(\ell)|$ for $1\leq\ell\leq\ell_1$, for every hash function $\mathfrak{h}\in\mathcal{H}_1$, there is a corresponding hash function $\mathfrak{h}'\in\mathcal{H}_1$ such that $\tilde{\lambda}_{S_{e_i}}(\mathfrak{h}) = -\tilde{\lambda}_{S_{e_i}}(\mathfrak{h}')$. We can therefore use Lemma 4.1 to bound X_{e_i} and get that

$$\mathbb{P}\left[X_{e_i} \leq -\alpha\cdot\sqrt{\sum_{\mathfrak{h}\in\mathcal{H}_1}\tilde{\lambda}^2_{S_{e_i}}(\mathfrak{h})}\right] \;\leq\; e^{-\alpha^2/2}.$$

Together with Lemma 4.4 and because $\lambda_{S_{e_i}}(\mathfrak{h})/\tilde{\lambda}_{S_{e_i}}(\mathfrak{h}) \geq 1$ for every $\mathfrak{h}\in\mathcal{H}_1$, we obtain that for every constant $c_2 > 1$, there is a constant $c_1 > 0$ such that

$$\mathbb{P}\left[X_{e_i} < -\alpha\cdot\sqrt{c_1\cdot|\mathcal{H}_{e_i}|}\right] \;<\; e^{-\alpha^2/2} + c_2^{-|\mathcal{H}_{e_i}|}.$$

Choosing $\alpha = \sqrt{2\ln(5k/\varepsilon)}$ and an appropriate constant c_2 (note that $|\mathcal{H}_{e_i}| \geq c\ln(k/\varepsilon)$), we have

$$\mathbb{P}\left[X_{e_i} < -\sqrt{2\cdot c_1\cdot\ln\left(\frac{5k}{\varepsilon}\right)\cdot|\mathcal{H}_{e_i}|}\right] \;<\; \frac{2\varepsilon}{5k}. \qquad (4)$$

In order to bound the value of Y_{e_i}, remember that $|\Lambda_\ell| = d(\ell) \leq \delta(e^{-\delta^2(\ell-1)^2/2} - e^{-\delta^2\ell^2/2})|\mathcal{H}_{e_1,e_i}|$. Hence, the probability that we choose at least one hash function $\mathfrak{h}$ from $\mathcal{H}_2$ with $|\lambda_{S_{e_i}}(\mathfrak{h})| > \lambda$ is at most $|\mathcal{H}_{e_i}|\cdot\delta\cdot e^{-\lambda^2/2}$. Hence, there is a constant $c_3 > 0$ such that

$$\mathbb{P}\left[\max_{\mathfrak{h}\in\mathcal{H}_2\cap\mathcal{H}_{e_i}}|\lambda_{S_{e_i}}(\mathfrak{h})| \;>\; c_3\cdot\sqrt{\ln\left(\frac{k+|\mathcal{H}_{e_i}|}{\varepsilon}\right)}\right] \;\leq\; \frac{\varepsilon}{5k}. \qquad (5)$$

Let us now consider the size of the set $\mathcal{H}_2$. By Condition (III) of the quasi-randomness definition, we have $|\mathcal{H}_2| \leq$

$\delta \cdot e \cdot |\mathcal{H}_{e_1,e_i}|$. The probability for choosing a hash function $\mathfrak{h} \in \mathcal{H}_2$ is therefore at most δe. Using Chernoff and $\mathbb{E}\big[|\mathcal{H}_{e_i} \cap \mathcal{H}_2|\big] \le e/\sqrt{\ln(K/\varepsilon)}$, we conclude that there is a constant $c_4 > 0$ such that

$$\mathbb{P}\left[|\mathcal{H}_{e_i} \cap \mathcal{H}_2| \; > \; c_4 \cdot \sqrt{\ln\left(\frac{|\mathcal{H}_{e_1}|}{\varepsilon}\right)}\right] \; \le \; \frac{\varepsilon}{5k}. \qquad (6)$$

Combining Inequalities (5) and (6), we obtain

$$\mathbb{P}\left[Y_{e_i} < -c_3 c_4 \cdot \ln\left(\frac{k}{\varepsilon}\right)\right] \le \frac{2\varepsilon}{5k}. \qquad (7)$$

Finally, by the definition of ℓ_1, we have that

$$\begin{aligned} \mathbb{P}\big[Z_{e_i} < 0\big] &\le \mathbb{P}\big[|\Lambda_\infty| > 0\big] \\ &\le 2(1+\delta) \cdot |\mathcal{H}_{e_i}| \cdot e^{-\delta^2 \ell_1^2/2} \le \frac{\varepsilon}{5k}. \end{aligned} \qquad (8)$$

Let

$$\nu := \left(\delta \cdot |\mathcal{H}_{e_i}| + \sqrt{c_1' \cdot \ln\left(\frac{k}{\varepsilon}\right) \cdot |\mathcal{H}_{e_i}|} + c_3 c_4 \cdot \ln\left(\frac{k}{\varepsilon}\right)\right)$$

for some constant c_1'. Combining Inequalities (3), (4), (7), and (8), we get that

$$\mathbb{P}\left[N_{e_i} < -\nu \cdot \sqrt{F_2[S_{e_i}]}\right] \; < \; \frac{\varepsilon}{k} \qquad (9)$$

for a constant $c_1' > 0$. Using $1 \le F_2[S_{e_i}]/m_1^2 \le k \le K$, we obtain

$$\begin{aligned} \delta \cdot |\mathcal{H}_{e_i}| \cdot \sqrt{F_2[S_{e_i}]} &< \frac{\sqrt{F_2[S_{e_i}]} \cdot |\mathcal{H}_{e_i}|}{8cK \ln^{3/2}(K/\varepsilon)} \\ &\le \frac{m_1^2 \cdot |\mathcal{H}_{e_i}|}{8c\sqrt{F_2[S_{e_i}]} \ln^{3/2}(K/\varepsilon)} \\ &< \frac{m_1}{8} \cdot |\mathcal{H}_{e_i}|. \end{aligned}$$

For the second term in the sum of Inequality (9), we have

$$\begin{aligned} \sqrt{c_1' \cdot \ln\left(\frac{k}{\varepsilon}\right) \cdot |\mathcal{H}_{e_i}| \cdot F_2[S_{e_i}]} &\le \frac{\sqrt{c \cdot c_1'} \cdot F_2[S_{e_i}] \ln(k/\varepsilon)}{m_1} \\ &\le \frac{m_1}{8} \cdot |\mathcal{H}_{e_i}| \end{aligned}$$

if c is chosen sufficiently large. Finally, the third term of Inequality (9) can be bounded by

$$c_3 c_4 \cdot \ln\left(\frac{k}{\varepsilon}\right) \cdot \sqrt{F_2[S_{e_i}]} \; \le \; \frac{m_1}{8} \cdot |\mathcal{H}_{e_i}| \qquad (10)$$

with $\sqrt{F_2[S_{e_i}]} \ge m_1$ and c large enough. Combining Inequalities (9) – (10) completes the proof. □

Next, an upper bound on the time complexity of $\mathcal{ALG}_{mode}$ is derived if the hash functions are chosen from a quasi-random family. The following theorem shows that we obtain the same asymptotic running time as with hash functions that are chosen uniformly from all possible hash functions.

Theorem 4.6. *If the hash functions are chosen from a quasi-random family $\mathcal{H}$ with parameters $\delta < 1/(8c \cdot K \cdot \ln^{3/2}(\log(K/\varepsilon))$ and $\varepsilon \in (0,1)$, $\mathcal{ALG}_{mode}$ needs $O(D + F_2/m_1^2 \log(k/\varepsilon))$ rounds to compute the mode with probability at least $1-\varepsilon$.*

Proof. We need to show that when using $r_1 = O(D + F_2/m_1^2 \log(k/\varepsilon))$ hash functions, the counters $c(e_i)$ of all elements e_i with multiplicity $m_i < m_1/2$ are smaller than the counter $c(e_1)$ of the mode with probability at least $1-\varepsilon$. This follows almost immediately from Equation (2) and Lemma 4.5. It only remains to show that $|\mathcal{H}_{e_i}| = \Omega(r_1)$ with high probability. This follows immediately from Condition (I) (2-independence) in the quasi-randomness definition by using a Chernoff bound. □

Since we only allow messages of size $O(\log K + \log M + \log n)$, $\mathcal{ALG}_{mode}$ is efficient only when applied to a quasi-random family $\mathcal{H}$ of size polynomial in K, M, and n. In the following, using the *probabilistic method* we show that indeed, small quasi-random hash families exist. We prove that if we choose sufficiently many random hash functions, they form a quasi-random family with positive (in fact high) probability.

Theorem 4.7. *For all parameters $\delta \in (0,1)$ and $\varepsilon \in (0,1)$, there is a quasi-random hash family $\mathcal{H}$ of size $|\mathcal{H}| = O((K + M + \log(1/\delta) + \log\log(1/\varepsilon))/\delta^6)$.*

Proof. Let $\mathcal{H} = \{\mathfrak{h}_1, \ldots, \mathfrak{h}_q\}$ be q hash functions that are chosen independently and uniformly at random from the set of all 2^K possible hash functions $\mathfrak{h} : [K] \to \{0,1\}$. We need that $\mathcal{H}$ is a quasi-random family with parameters δ and ε with positive probability. Let us now examine the three conditions of the quasi-randomness definition.

2-independence:
For two elements e and e' from $[K]$, we have $\mathbb{P}\big[|\mathcal{H}_{e,e'}| < (1-\delta) \cdot |\mathcal{H}/2\big] < e^{-\delta^2 q/4}$ by Chernoff's inequality. By using a union bound, we therefore get that

$$\mathbb{P}\big[\mathcal{H} \text{ does not satisfy Condition (I)}\big] \; < \; K^2 \cdot e^{-\delta^2 q/4}. \quad (11)$$

Exponentially small imbalance:
We assume that $\mathcal{H}$ satisfies Condition (1). Let $e, e' \in [K]$, $S \subseteq \big([K] \setminus \{e,e'\}\big) \times [M]$, and $\ell \le \ell_1$. Further, let X be the fraction of hash functions $\mathfrak{h} \in \mathcal{H}_{e,e'}$ that are not $\delta\ell$-good with respect to S. By Lemma 4.1 and by Chernoff's inequality, $\mathbb{P}\big[X > (2+\delta)e^{-\delta^2\ell^2/2}\big] < e^{-\delta^2 2e^{-\delta^2\ell^2/2}|\mathcal{H}_{e,e'}|/8}$. Note that being in $\mathcal{H}_{e,e'}$ and being $\delta\ell$-good with respect to S are independent. With a union bound, we get

$$\mathbb{P}\big[\mathcal{H} \text{ does not satisfy Condition (II)}\big] <$$
$$K^2 \cdot 2^K \cdot 2^M \cdot \ell_1 \cdot e^{-\delta^2 2e^{-\delta^2\ell_1^2/2}(1-\delta)q/16}.$$

Independence of imbalance sign:
We assume that $\mathcal{H}$ satisfies Conditions (I) and (II). Let $e, e' \in [K]$, $S \subseteq \big([K]\setminus\{e,e'\}\big) \times [M]$, and $\ell \le \ell_1$. Let Λ be the set of hash functions $\mathfrak{h} \in \mathcal{H}$ for which $|\lambda_S(\mathfrak{h})| \in (\delta(\ell-1), \delta\ell]$. Further let $\Lambda^+ \subseteq \Lambda$ be the hash functions $\mathfrak{h}$ for which $\mathfrak{h}(e) \cdot \lambda_S(\mathfrak{h}) > 0$ and let $\Lambda^- = \Lambda \setminus \Lambda^+$. We need to bound the value of $Y = \big||\Lambda^+| - |\Lambda^-|\big|$. Because the value of $\mathfrak{h}(e)$ is independent of $\lambda_S(\mathfrak{h})$, for $\gamma \in (0,1)$, we get $\mathbb{P}\big[Y > \gamma \cdot |\Lambda|\big] < 2e^{-\gamma^2|\Lambda|/4}$ by using Chernoff's inequality. We define $\Phi = e^{-\delta^2(\ell-1)^2/2} - e^{-\delta^2\ell^2}/2$ and assume that $|\Lambda| = \beta \cdot \Phi \cdot q$. We then have

$$\begin{aligned} \mathbb{P}\big[Y > \delta \cdot \Phi \cdot q\big] &= \mathbb{P}\left[Y > \frac{\delta}{\beta} \cdot |\Lambda|\right] \\ &< 2e^{-\delta^2|\Lambda|/(4\beta^2)} \\ &= 2e^{-\delta^2\Phi \cdot q/(4\beta)}. \end{aligned}$$

Based on the assumption that $\mathcal{H}$ satisfies Condition (II), we have

$$\frac{\beta}{\Phi} \leq \frac{(2+\delta)e^{-\delta^2(\ell-1)^2/2}}{\left(e^{-\delta^2(\ell-1)^2/2} - e^{-\delta^2\ell^2/2}\right)^2} = \frac{2+\delta}{\left(1-e^{-\delta^2(\ell-1/2)}\right)^2 \cdot e^{-\delta^2(\ell-1)^2/2}}.$$

The bound on β/Φ is largest if $\ell = 1$. We then get $\beta/\Phi \leq (2+\delta)/(\delta^2/2 - \delta^4/4)^2$. Hence, we obtain

$$\mathbb{P}\big[\mathcal{H} \text{ does not satisfy Condition (III)}\big] < K^2 \cdot 2^K \cdot 2^M \cdot \ell_1 \cdot 2e^{-\frac{\delta^2}{4}\cdot\left(\frac{\delta^2}{2}-\frac{\delta^4}{4}\right)^2 \cdot q}.$$

If we choose

$$q \geq \frac{C}{\delta^6} \cdot \left(K + M + \log\left(\frac{\sqrt{\ln(K/\varepsilon)}}{\delta}\right)\right)$$

for a sufficiently large constant C, the right-hand sides of the inequalities become less than 1/3 and it is therefore possible to satisfy Conditions (I)-(III). This concludes the proof. $\square$

Theorems 4.6 and 4.7 conclude our analysis of $\mathcal{ALG}_{mode}$: We have shown that $\mathcal{ALG}_{mode}$ needs no more than $O(D + F_2/m_1^2 \cdot \log(k/\varepsilon))$ rounds for a given error probability ε, and that there indeed exist sufficiently small sets of hash functions yielding the desired message size.

Remark: A natural generalization of our models presented so far is to allow *weighted* elements, i.e., each element $e_i \in \mathcal{A}$ has a corresponding weight w_i. The task of finding the mode would then translate into finding the element which maximizes the product of the frequency and weight, i.e., the mode is defines as the element e_i maximizing $w_i m_i$. It is easy to see that our algorithms can be generalized to the case of weighted elements as well.

5. LOWER BOUND

In this section, we show that for every $m_1, F_5 \in \mathbb{N}$ with $F_5 \geq m_1^5$, there is a graph G with diameter D and a frequency distribution with maximum frequency m_1 and a 5th frequency moment F_5 such that finding the mode requires $\Omega(D + F_5/(B \cdot m_1^5))$ rounds. Here B denotes the number of bits that can be transmitted in a single message.

To prove our lower bound, we use a reduction from the *set disjointness problem*, a well-known problem from communication complexity theory [11]. Assume that two nodes u_1 and u_2 have sets of elements S_1 and S_2 with $|S_1| = |S_2| = \ell$ such that $|S_1 \cap S_2| \in \{0,1\}$. In [16], Razborov showed that in order to distinguish between the case where S_1 and S_2 are disjoint and the case where they intersect in exactly one element, u_1 and u_2 have to exchange $\Omega(\ell)$ bits even if we allow them to err with probability $\varepsilon \leq \varepsilon_0$ for a constant ε_0. In order to derive a lower bound on the space complexity for approximating the frequency moments F_ℓ of a distribution in a streaming model, Alon et al. extended Razborov's lower bound to a scenario with more than two nodes [1]. Assume that there are d nodes $u_1, \ldots, u_d$ with sets $S_1, \ldots, S_d$ with $|S_1| = \ldots = |S_d| = \ell$ such that either the sets are pairwise disjoint or there is an element e such that $S_i \cap S_j = \{e\}$ for all $1 \leq i < j \leq d$. The following theorem is proven in [1].

Theorem 5.1. *[1] For every $\ell \geq d^4$, the total number of bits that the nodes $u_1, \ldots, u_d$ have to communicate in order to distinguish between the case where the sets are pairwise disjoint and the case where they intersect in exactly one element is $\Omega(\ell/d^3)$. This also holds for randomized algorithms with error probability $\varepsilon < 1/2$.*

We prove the lower bound for computing the mode in two steps. We first look at a special type of frequency distributions where the frequencies $m_2, \ldots, m_k$ are equal up to a factor of 2 and then generalize the result to arbitrary frequency distributions. In the following, we will assume that all the nodes know about the frequency distribution are bounds m_1^+ and m_i^- for $i \neq 2$ such that it is guaranteed that $m_1^+/2 \leq m_1 \leq m_1^+$ and that $m_i^- \leq m_i \leq 2m_i^- < m_1^+/2$. We require that an algorithm works for all distributions satisfying the given bounds and for all assignments of elements to nodes. We first look at the case where the bound m_i^- is the same for all i.

Theorem 5.2. *Let $m_1, F_5 \in \mathbb{N}$ such that $F_5 \geq m_1^5$. There is a frequency distribution $m_1 > m_2 \geq m_3 \geq \ldots \geq m_k$ with $F_5 = \sum_{i=1}^k m_i^5$ such that for every $D > 1$, there is a graph G with diameter D on which finding the mode needs*

$$\Omega\left(D + \frac{F_5}{m_1^5 \cdot B}\right)$$

rounds where B is the number of bits that can be sent in a single message.

Proof. We set $m_2 = m_3 = \ldots = m_k = 1$, which implies that $F_5 = \sum_{i=1}^k m_i^5 = m_1^5 + k - 1$. Let k' be the maximal integer $k' \geq k$ such that m_1 divides $k' - 1$. We assume that the algorithm knows that the elements $e_{k'+1}, \ldots, e_k$ need not be considered. It then remains to solve the problem for the frequency distribution $m_1, \ldots, m_{k'}$. Note that finding the mode for this distribution can be at most as hard as finding the mode for the frequency distribution $m_1, \ldots, m_k$ because additional information cannot make the problem harder. We can further assume that $F_5 \geq 2m_1^5$ since otherwise the statement of the theorem becomes trivial. This implies that $k - 1 \geq m_1^5$ and thus $k' > m_1^5 - m_1 + 1$.

Let G be a star graph with $m_1 + 1$ nodes, i.e., G consists of an inner node of degree m_1 and m_1 leaves $u_1, \ldots, u_{m_1}$. There is a total of $m_1 + k' - 1$ elements (element e_1 occurs m_1 times, all other $k' - 1$ elements occur only once). We distribute these elements among the m_1 leaf nodes such that every leaf node receives e_1 exactly once and $(k'-1)/m_1$ of the other elements. Let S_i be the set of elements of leaf node u_i. For $i \neq j$, we have $S_i \cap S_j = \{e_1\}$. we can apply Theorem 5.1 to show that the total number of bits the nodes $u_1, \ldots, u_d$ have to communicate in order to find e_1 is $\Omega(k'/m_1/m_1^3) = \Omega(k'/m_1^4)$. This is due to the fact that any algorithm $\mathcal{ALG}$ which computes e_1 can be used to solve the problem of Theorem 5.1 as follows. If $\mathcal{ALG}$ terminates without returning a value e_1, we know that the sets $S_1, \ldots, S_d$ are pairwise disjoint. If $\mathcal{ALG}$ returns a value e_1, we can test whether e_1 indeed is in all sets S_i be exchanging a logarithmic number of additional bits. Note that we have $1 + (k'-1)/m_1 \geq m_1^4$ (corresponding to the condition $\ell \geq d^4$ in Theorem 5.1) because we assumed that $F_5 \geq m_1^5$. Because $k \geq k' > k(1 - 1/m_1^4)$, we also obtain $\Omega(k'/m_1^4) = \Omega(k/m_1^4)$. Hence, since one of the leaf nodes has to send at least $\Omega(k/m_1^5)$ bits, the number of rounds is at least $\Omega(k/(m_1^5 B))$.

From G, we obtain a graph with diameter D on which the lower bound becomes $\Omega(D + k'/(m_1^5 B))$ by replacing each edge in G by a path of length $D/2$. We get a lower bound of $\Omega(D + F_5/(m_1^5 B))$ because

$$\frac{F_5}{m_1^5} = \frac{\sum_{i=1}^{k} m_i^5}{m_1^5} = \frac{m_1^5 + (k-1)}{m_1^5} < \frac{2k}{m_1^5}.$$

The inequality follows because we assume that $F_5 = m_1^5 + k - 1 \geq 2m_1^5$. □

6. POWER LAW DISTRIBUTIONS

An interesting and widely studied distribution is the *power-law distribution* $p(x) \propto x^{-\alpha}$ for some constant $\alpha > 0$ [2, 12, 17], that is, by normalization, $m_i = 1/i^\alpha$. Let $m = \sum_{i=1}^{k} m_i$. It holds that $n \in \Theta(k^{1-\alpha})$ for $\alpha < 1$, $m \in \Theta(\log k)$ for $\alpha = 1$ and $m \in \Theta(1)$ for $\alpha > 1$. For $\mathcal{ALG}_{mode}$, we hence obtain the following upper bounds on the running time T:

$$T \in \begin{cases} O\big(D + k^{1-2\alpha} \cdot (\log k + \log(1/\varepsilon))\big), & \text{if } \alpha < 1/2 \\ O\big(D + \log k \cdot (\log k + \log(1/\varepsilon))\big) & \text{if } \alpha = 1/2 \\ O\big(D + \log k + \log(1/\varepsilon)\big), & \text{if } \alpha > 1/2. \end{cases}$$

We observe an interesting threshold phenomenon. If $\alpha < 1/2$, our randomized algorithm needs polynomial time whereas for $\alpha \geq 1/2$, the mode can be determined in polylogarithmic time. Our lower bound on the time T needed to find the mode becomes $\Omega\big(D + k^{1-5\alpha}/(B \log k)\big)$ if $\alpha < 1/5$. For $\alpha \geq 1/5$, we do not obtain a non-trivial lower bound. Hence, indeed, there seems to exist a value $\alpha_0 \geq 1/5$ such that the time complexity of every algorithm is polynomial in k if $\alpha < \alpha_0$.

7. CONCLUSIONS

This paper has shown that the mode can be computed deterministically in time $O(D+k)$ in arbitrary graphs and that there are distributions for which this is tight up to a logarithmic factor. In an effort to exploit properties of the actual frequency distribution, we presented a randomized Monte Carlo type algorithm which finds the mode with high probability in time $O(D + F_2/m_1^2 \cdot \log k)$. We did not prove that this is tight; however, the lower bound of $\Omega(D + F_5/(m_1^5 B))$ rounds shows that the general dependence of the upper bound on the skewness of the distribution is correct. We believe that at least up to polylogarithmic factors the upper bound is the correct bound for all distributions. An improvement of the lower bound would most likely also solve an open problem from [1]: It is shown that in a streaming model, F_0, F_1, and F_2 can be approximated in polylogarithmic space, whereas polynomial space is needed to approximate F_ℓ for $\ell \geq 6$. It is conjectured in [1] that approximating F_3, F_4, and F_5 also requires polynomial time.

We formulated all our results for a message passing model where communication is constricted by congestion on edges—note that we do not bound congestion at nodes—and by the adjacency relationships of the underlying network graph. However, our general approach directly applies to all distributed models (e.g. gossiping [4]) where aggregation can be studied. The time complexity of our algorithm is then equal to the time complexity of computing $O(F_2/m_1^2 \cdot \log k)$ independent basic aggregation functions such as computing the sum or the maximum of all elements.

8. REFERENCES

[1] N. Alon, Y. Matias, and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. In *Proc. 28th Annual ACM Symposium on Theory of Computing (STOC)*, 2002.

[2] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. 18th Annual IEEE Conference on Computer Communications (INFOCOM)*, 1999.

[3] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

[4] J. G. D. Kempe, A. Dobra. Gossip-Basid Computation of Aggregate Information. In *Proc. 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.

[5] D. Dobkin and J. I. Munro. Determining the Mode. *Theoretical Computer Science*, 12:255–263, 1980.

[6] M. Durand and P. Flajolet. LogLog Counting of Large Cardinalities. In *Proc. 11th Annual European Symposium on Algorithms (ESA)*, 2003.

[7] A. Farzan, P. Ferragina, G. Franceschini, and J. J. Munro. Cache-Oblivious Comparison-Based Algorithms on Multisets. In *Proc. 13th Annual European Sympoisum on Algorithms (ESA)*, 2005.

[8] P. Flajolet and G. N. Martin. Probabilistic Counting Algorithms for Data Base Applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[9] B. Kalyanasundaram and G. Schnitger. The Probabilistic Communication Complexity of Set Intersection. *SIAM Journal on Discrete Mathematics (SIDMA)*, 5(4):545–557, 1992.

[10] F. Kuhn, T. Locher, and R. Wattenhofer. Tight Bounds for Distributed Selection. In *Proc. 19th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2007.

[11] E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[12] M. Mitzenmacher. A Brief History of Generative Models for Power Law and Lognormal Distributions. *Internet Mathematics*, 1:226–251, 2003.

[13] J. I. Munro and P. M. Spira. Sorting and Searching in Multisets. *SIAM Journal of Computing (SICOMP)*, 5(1):1–8, 1976.

[14] B. Patt-Shamir. A Note on Efficient Aggregate Queries in Sensor Networks. *Theoretical Computer Science.*, 370(1-3):254–264, 2007.

[15] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.

[16] A. A. Razborov. On the Distributional Complexity of Disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.

[17] H. Simon. On a Class of Skew Distribution Functions. *Biometrika*, 42:425–440, 1955.