Optimal Gossip Algorithms for Exact and Approximate Quantile Computations

Bernhard Haeupler*
CMU
haeupler@cs.cmu.edu

Jeet Mohapatra
MIT
jeetmo@mit.edu

Hsin-Hao Su
UNC Charlotte
hsinhaosu@uncc.edu

ABSTRACT

This paper gives drastically faster gossip algorithms to compute exact and approximate quantiles.

Gossip algorithms, which allow each node to contact a uniformly random other node in each round, have been intensely studied and been adopted in many applications due to their fast convergence and their robustness to failures. Kempe et al. [24] gave gossip algorithms to compute important aggregate statistics if every node is given a value. In particular, they gave a beautiful $O(\log n + \log \frac{1}{\epsilon})$ round algorithm to ϵ-approximate the sum of all values and an $O(\log^2 n)$ round algorithm to compute the exact ϕ-quantile, i.e., the $\lceil \phi n \rceil$ smallest value.

We give an quadratically faster and in fact optimal gossip algorithm for the exact ϕ-quantile problem which runs in $O(\log n)$ rounds. We furthermore show that one can achieve an exponential speedup if one allows for an ϵ-approximation. In particular, we give an $O(\log \log n + \log \frac{1}{\epsilon})$ round gossip algorithm which computes a value of rank between ϕn and $(\phi + \epsilon)n$ at every node. Our algorithms are extremely simple and very robust - they can be operated with the same running times even if every transmission fails with a, potentially different, constant probability. We also give a matching $\Omega(\log \log n + \log \frac{1}{\epsilon})$ lower bound which shows that our algorithm is optimal for all values of ϵ.

ACM Reference Format:
Bernhard Haeupler, Jeet Mohapatra, and Hsin-Hao Su. 2018. Optimal Gossip Algorithms for Exact and Approximate Quantile Computations. In *PODC '18: ACM Symposium on Principles of Distributed Computing, July 23–27, 2018, Egham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3212734.3212770

1 INTRODUCTION

Today, due to the vast amount of data and advances in connectivity between computers, distributed data processing has become increasingly important. In distributed systems, data is stored across different nodes. When one requires some aggregate properties of the data, such as, sums, ranks, quantiles, or other statistics, nodes must communicate in order to compute these properties. Aggregating data in an efficient and reliable way is a central topic in distributed systems such as P2P networks and sensor networks [27, 39, 40].

We consider uniform gossip protocols, which are often very practical due to their fast convergence, their simplicity, and their stability under stress and disruptions. In uniform gossiping protocols, computation proceeds in synchronized rounds. In each round, each node chooses to **pull** or **push**. In a push a node chooses a message, which is delivered to a uniformly random other node. In a pull each node receives a message from a random node. The message size is typically restricted to $O(\log n)$ bits. The (time) complexity of an algorithm is measured by the number rounds executed. One typically wants algorithms that succeed with high probability, i.e., with probability at least $1 - 1/\text{poly}(n)$.

In this paper, we study the quantile computation problem. In the **exact ϕ-quantile problem** each node v is given a distinct[1] $O(\log n)$ bit value x_v and wants to compute the $\lceil \phi n \rceil$ smallest value overall. In the **ϵ-approximate ϕ-quantile problem** each node wants to compute a value whose rank is between $(\phi + \epsilon)n$ and $(\phi - \epsilon)n$.

Previously, Kempe et al. [24] gave a beautiful and simple $O(\log n + \log \frac{1}{\epsilon})$ round gossip algorithm to approximate the sum of all values up to a $(1 \pm \epsilon)$ factor. They also showed how to use this algorithm to solve the exact ϕ-quantile problem in $O(\log^2 n)$ rounds with high probability.

The main result of this paper is a quadratically faster gossip algorithm for the ϕ-quantile problem. The $O(\log n)$ round complexity of our algorithm means that the exact ϕ-quantile can be computed as fast as broadcasting a single message.

Theorem 1.1. *For any $\phi \in [0, 1]$ there is a uniform gossip algorithm which solves the exact ϕ-quantile problem in $O(\log n)$ rounds with high probability using $O(\log n)$ bit messages.*

Clearly the running time of this algorithm is optimal as $\Omega(\log n)$ rounds are known to be necessary to even just broadcast to each node the ϕ-quantile value, after it has been identified.

Equally interestingly we show that one achieve even faster algorithms if one considers the approximate ϕ-quantile problem. While a $O(\frac{1}{\epsilon^2} \log n)$ round algorithm for an ϵ-approximation follows from simple sampling and a $O(\log n)$ round algorithm computing the median up to a $\pm O(\sqrt{\frac{\log n}{n}})$ (but not general quantiles) was given by Doerr et al. [7], no approximation algorithm for the quantile problem with a sub-logarithmic round complexity was known prior

*Supported in part by NSF grants CCF-1527110, CCF-1618280 and NSF CAREER award CCF-1750808.

PODC '18, July 23–27, 2018, Egham, United Kingdom

ACM ISBN 978-1-4503-5795-1/18/07…$15.00
https://doi.org/10.1145/3212734.3212770

[1] The assumption of distinct values is without loss of generality and made for simplicity as one can always break ties consistently, e.g., according to the ID of the node where the value started.

to this work. We give an $O(\log\log n + \log(1/\epsilon))$ round algorithm for ϵ-approximating any ϕ-quantile which, for arbitrarily good constant approximations, is exponentially faster than our optimal exact algorithm:

Theorem 1.2. *For any constant or non-constant $\epsilon(n) > 0$ and any $\phi \in [0,1]$, there exists a uniform gossip algorithm that solves the ϵ-approximate ϕ quantile problem in $O(\log\log n + \log\frac{1}{\epsilon(n)})$ rounds with high probability using $O(\log n)$ bit messages.*

We also give a $\Omega(\log\frac{1}{\epsilon})$ and a $\Omega(\log\log n)$ lower bound for the ϵ-approximate ϕ-quantile problem, which shows that our algorithm is optimal for essentially any value of ϵ.

Theorem 1.3. *For any $\frac{10\log n}{n} < \epsilon < 1/8$ and $\phi \in [0,1]$, any gossip algorithm that uses less than $\frac{1}{2}\log\log n$ or less than $\log_4\frac{8}{\epsilon}$ round fails to solve the ϵ-approximate ϕ-quantile problem with probability at least $1/3$. This remains true even for unlimited message sizes.*

We furthermore show that our algorithms can be made **robust to random failures**, i.e., the same round complexities apply even if nodes fail with some constant probability. We remark that a small caveat of excluding an $\exp(-t)$ fraction of nodes after a running time of t rounds is necessary and indeed optimal given that this is the expected fraction of nodes which will not have participated in any successful push or pull after t rounds.

Theorem 1.4. *Suppose in every round every node fails with a, potentially different, probability bounded by some constant $\mu < 1$. For any $\phi \in [0,1]$ there is a gossip algorithm that solves the ϕ quantile problem in $O(\log n)$ rounds. For any t and any $\epsilon(n) > 0$, there furthermore exists a gossip algorithm that solves the ϵ-approximate ϕ-quantile problem in $O(\log\log n + \log\frac{1}{\epsilon(n)} + t)$ rounds for all but $\frac{n}{2^t}$ nodes, with high probability.*

Despite being extremely fast and robust our algorithms remain very simple. In fact they do little more than repeatedly sampling two or three nodes, requesting their current value and selecting the largest, smallest, or median value. Given that in many cases an ϵ-approximation is more than sufficient we expect this algorithm to easily find applications in areas like sensor networks and distributed database systems. For instance, suppose that a sensor network consisting of thousands of devices is spread across an object to monitor and control the temperature. Say the top and bottom 10%-quantiles need special attention. By computing the 90%- and 10%-quantile, each node can determine whether it lies in the range. It is unlikely that such a computation needs to be exact. In fact, for any $0 < \epsilon < 1$, running $O(1/\epsilon)$ approximate quantile computations suffices for each node to determine its own quantile/rank up to an additive ϵ. The fact that this can be done in $O(\log\log n)$ rounds further demonstrates the power of approximations as there is not even a $o(n)$ algorithm known[2] which allows each node to compute its quantile/rank exactly.

Corollary 1.5. *For any constant or non-constant $\epsilon > 0$ and given that each node has a value there is a gossip algorithm that allows every node to approximate the quantile of its value up to an additive ϵ in $\frac{1}{\epsilon}\cdot O(\log\log n + \log\frac{1}{\epsilon})$ rounds with high probability using $O(\log n)$ bit messages.*

[2]The trivial algorithm which broadcasts the maximum value n times requires $O(n\log n)$ rounds. Using network coding gossip [18] one can improve this to $O(n)$ rounds. It is likely that computing the exact rank at each node cannot be done faster.

Technical Summary. While computing exact quantiles seems to be a very different problem from the approximate problem at first we achieve our exact algorithm by first designing an extremely efficient solution to the ϵ-approximate quantile problem running in $O(\log\log n + \log\frac{1}{\epsilon})$ rounds. This algorithm is in some sense based on sampling and inherently is not able to work for the exact quantile problem or too small ϵ itself, as it might randomly discard the target value in its first iteration. However, we show that the algorithm does work for ϵ larger than some polynomial in n. This allows us to bootstrap this algorithm and repeatedly remove a polynomial fraction of values within $O(\log n)$ rounds until, after a constant number of such iterations, only the target value persists. Overall this leads to an $O(\log\log n + \log\frac{1}{\epsilon})$ algorithm for the ϵ-approximate ϕ-quantile problem which works for ϵ values that can be any function of n. This actually generalizes the $O(\log n)$ algorithm for the exact ϕ-quantile problem, too, given that the $\frac{1}{2n}$-approximate ϕ-quantile problem is accurate enough to compute exact ranks and quantiles.

Next we outline an, indeed very simple, intuition of why a time complexity of $O(\log\log n + \log\frac{1}{\epsilon})$ is conceivable for the quantile computation problem: Suppose we sample $\Theta(\log n/\epsilon^2)$ many values uniformly and independently at random. With high probability the ϕ-quantile of the sampled values is an ϵ-approximation to the ϕ-quantile in the original data. Since each node can sample t node values (with replacement) in t rounds, one immediately get an $O(\log n/\epsilon^2)$ round algorithm that uses $O(\log n)$ bit messages.

Using larger message sizes, it is possible to reduce the number of rounds. Consider the following *doubling algorithm*: Each node v maintains a set S_v such that initially $S_v = \{x_v\}$. In each round, let $t(v)$ be the node contacted by v. Node v updates S_v by setting $S_v \leftarrow S_v \cup S_{t(v)}$. Since the set size essentially doubles each round, after $\log O(\log n/\epsilon^2) = O(\log\log n + \log\frac{1}{\epsilon})$ rounds, we have sampled $\Omega(\log n/\epsilon^2)$ values uniformly, albeit not quite independently, at random. A careful analysis shows that indeed a $O(\log\log n + \log\frac{1}{\epsilon})$ running time can be achieved using messages of size $\Theta(\log^2 n/\epsilon^2)$ bits.

Our first approach for reducing the message sizes tried to utilize the quantile approximation sketches from the streaming algorithm community (see related work). We managed to reduce the message complexity to $O(\frac{1}{\epsilon}\cdot\log n\cdot(\log\log n + \log(1/\epsilon)))$ by adopting the compactor ideas from these streaming sketches. A write-up of this might be of independent interest and can be found in the full version. Unfortunately even if one could losslesly port the state-of-the-art compactors scheme from [21] into this setting, the $\Omega(\frac{1}{\epsilon}\log\log(1/\delta))$ for getting ϵ-approximate quantile sketches with probability $1-\delta$ suggests that one cannot achieve a $o(\log n\log\log n)$ message size this way, even if ϵ is a constant. In contrast to most other distributed models of computation gossip algorithms furthermore do not allow to easily shift extra factors in the message size over to the round complexity. In fact, we did not manage to devise any $o(\log n)$ round gossip algorithm based on sketching which adheres to the standard $O(\log n)$ bound on message sizes.

Instead of sticking to the centralized mentality where each node tries to gather information to compute the answer, consider the following 3-TOURNAMENT mechanism. In each iteration, each node uniformly samples three values (in three rounds) and assign its value to the middle one[3]. Intuitively nodes close to the median should have a higher probability of surviving and being replicated. Indeed, we show that the number of nodes that have values that are ϵ close to the median grows exponentially for the first $O(\log \frac{1}{\epsilon})$ iterations. After that, the number of nodes with values more than ϵn away from the median decreases double exponentially. For sufficiently large ϵ this simple 3-TOURNAMENT algorithm gives an approximation of the the median in $O(\log \frac{1}{\epsilon} + \log\log n)$ iterations. In general, if we want to approximate the ϕ-quantile, we shift the $[\phi-\epsilon, \phi+\epsilon]$ quantiles to the quantiles around the median by the following 2-TOURNAMENT mechanism. If $\phi < 1/2$, each node samples two values and assign its value to the higher one. The case for $\phi > 1/2$ is symmetric. Intuitively this process makes it more likely for nodes with higher/smaller values to survive. Indeed, we show that with a bit of extra care in the last iterations, $O(\log \frac{1}{\epsilon})$ invocations suffice to shift the values around the ϕ-quantile to almost exactly the median, at which point one can apply the median approximation algorithm.

Finally, our lower bound comes from the fact that if one chooses $\Theta(\epsilon n)$ nodes and either gives them a very large or a very small value, then knowing which of the two cases occurred is crucial for computing any ϵ-approximate quantiles. However, initially only these $\Theta(\epsilon n)$ nodes have such information. We show that it takes $\Omega(\log\log n + \log \frac{1}{\epsilon})$ rounds to spread the information from these nodes to every node, regardless of the message size.

Related Work. The randomized gossip-based algorithms dates back to Demers et al. [6]. The initial studies are on the spreading of a single message [14, 22, 32], where Karp et al. [22] showed that $O(\log n)$ round and $O(n \log\log n)$ total messages is sufficient to spread a single message w.h.p. Kempe et al. [24] studied gossip-based algorithms for the quantile computation problem as well as other aggregation problems such as computing the sum and the average. Kempe et al. developed $O(\log n)$ rounds algorithm to compute the sum and average w.h.p. Later, efforts have been made to reduce the total messages to $O(n \log\log n)$ for computing the sum and the average [4, 23]. Using the ability to sample and count, Kempe et al. implemented the classic randomized selection algorithm [12, 19] in $O(\log^2 n)$ rounds. Doerr et al. [7] considered gossip algorithms for the problem of achieving a stabilizing consensus algorithm under adversarial node failures. They analyze the median rule, i.e., sample three values and keep the middle value, in this setting and show that $O(\log n)$ rounds suffice to converge to an $\pm O(\sqrt{\frac{\log n}{n}})$-approximate median even if $O(\sqrt{n})$ adversarial node failures occur. Similar gossip dynamics were also studied in [10] which considers randomly corrupted (binary) messages.

The exact quantile computation is also known as the selection problem, where the goal is to select the k'th smallest element. The problem has been studied extensively in both centralized and distributed settings. Blum et al. [3] gave a deterministic linear time algorithm for the problem in the centralized setting. In the distributed setting, Kuhn et al. [25] gave an optimal algorithm for the selection problem that runs in $O(D \log_D n)$ rounds in the CONGEST model, where D is the diameter of the graph. Many works have been focused on the communication complexity aspect (i.e. the total message size sent by each node) of the problem [16, 30, 31, 34–38]. Most of them are for complete graphs or stars. Others studied specific class of graphs such as two nodes connected by an edge [5, 33], rings, meshes, and complete binary trees [13].

The quantile computation problem has also been studied extensively in the streaming algorithm literature [1, 2, 11, 15, 17, 20, 21, 26, 28, 29], where the goal is to approximate ϕ-quantile using a small space complexity when the data comes in a single stream.

[3]Such gossip dynamics have also been analyzed in [7] for the setting of adversarial node failures and in [10] for the setting of random message corruptions.

2 THE TOURNAMENT ALGORITHMS

In this section, we present our algorithm for the ϵ-approximate quantile computation problem for sufficiently large ϵ. For convenience, we use $a \pm b$ to denote the interval $[a-b, a+b]$.

Theorem 2.1. *For any constant or non-constant $\epsilon(n) = \Omega(1/n^{0.096})$ and any $\phi \in [0,1]$, there exists a uniform gossip algorithm that solves the ϵ-approximate ϕ quantile problem in $O(\log\log n + \log \frac{1}{\epsilon(n)})$ rounds with high probability using $O(\log n)$ bit messages.*

The algorithm is divided into two phases. In the first phase, each node adjusts its value so that the quantiles around the ϕ-quantile will become the median quantiles approximately. In the second phase, we show how to compute the approximate median.

2.1 Phase I: Shifting the Target Quantiles to Approximate Medians

Algorithm 1 2-TOURNAMENT(v)

$h_0 \leftarrow (1-(\phi+\epsilon))$
$i \leftarrow 0$, $T = 1/2 - \epsilon$.
while $h_i > T$ **do**
  $h_{i+1} \leftarrow h_i^2$
  $\delta \leftarrow \min\left(1, \frac{h_i - T}{h_i - h_{i+1}}\right)$
  With probability δ **do**
    Select two nodes $t_1(v)$ and $t_2(v)$ randomly
    $x_v \leftarrow \min(x_{t_1(v)}, x_{t_2(v)})$
  Otherwise **do**
    Select a node $t_1(v)$ randomly
    $x_v \leftarrow x_{t_1(v)}$
  $i \leftarrow i+1$
end while

Let L_i, M_i, and H_i denote the nodes whose quantiles lie in $[0, \phi-\epsilon)$, $[\phi-\epsilon, \phi+\epsilon]$, and $(\phi+\epsilon, 1]$ respectively at the end of iteration i, and L_0, M_0, and H_0 be the nodes with those quantiles in the beginning. We run the 2-TOURNAMENT algorithm (Algorithm 1) until the iteration t such that $h_t \le T$, where $T = 1/2 - \epsilon$. The goal is to show that by the end of iteration t, the size of $|L_t|$ and $|H_t|$ are $(1/2 - \Omega(\epsilon)) \cdot n$ so that an approximate median lies in M_t, which consists of our target quantiles.

Let $h_0 = 1 - (\phi + \epsilon)$ and $l_0 = \phi - \epsilon$. We first consider the case where $h_0 \geq l_0$ and the other case is symmetric. Initially, we have $\frac{|H_0|}{n} \in h_0 \pm 1/n$. Let $h_{i+1} = h_i^2$ for $i \geq 1$. We will show that $t = O(\log(1/\epsilon))$ and $\frac{|H_i|}{n}$ concentrates around h_i for iteration $1 \leq i \leq t-1$ and in the end we have $\frac{|H_t|}{n} \in T \pm \frac{\epsilon}{2}$.

The algorithm ends when h_i decreases below T. The lemma below bounds the number of iterations needed for this to happen. It can be shown that since h_i squares in each iteration, the quantity $(1 - h_i)$ roughly grows by a constant factor in each iteration. Since initially $(1-h_0) \geq \epsilon$, $O(\log(1/\epsilon))$ iterations suffice for h_i to decrease below T. The missing proofs of the lemmas in this section can be found in the full version.

Lemma 2.2. *Let t denote the number of iterations in Algorithm 1, $t \leq \log_{7/4}(4/\epsilon) + 2$.*

Proof. The algorithm ends when $h_t \leq 1/2 - \epsilon$. The highest possible value for h_0 is $1 - \epsilon$. We show that $h_i \leq 1 - \left(\frac{7}{4}\right)^i \cdot \epsilon$ provided that $1 - \left(\frac{7}{4}\right)^{i-1} \cdot \epsilon \geq 3/4$.

Suppose by induction that $h_{i-1} \leq 1 - \left(\frac{7}{4}\right)^{i-1} \cdot \epsilon$. We have

$$
\begin{aligned}
h_i &= h_{i-1}^2 \\
&\leq \left(1 - \left(\frac{7}{4}\right)^{i-1} \cdot \epsilon\right)^2 \\
&= 1 - \left(\frac{7}{4}\right)^{i-1} \cdot \epsilon \cdot \left(2 - \left(\frac{7}{4}\right)^{i-1} \cdot \epsilon\right) \\
&\leq 1 - \left(\frac{7}{4}\right)^{i-1} \cdot \epsilon \cdot \left(2 - \frac{1}{4}\right) \leq 1 - \left(\frac{7}{4}\right)^{i} \cdot \epsilon \quad 1 - \left(\frac{7}{4}\right)^{i-1} \cdot \epsilon \geq 3/4
\end{aligned}
$$

Therefore, after $i_0 = \log_{7/4}(4/\epsilon)$ iterations we have, $h_{i_0} \leq 1 - (\frac{7}{4})^i \cdot \epsilon \leq 1 - \frac{1}{4} = \frac{3}{4}$. Since $h_{i_0+2} \leq (\frac{3}{4})^4 < \frac{1}{2} - \epsilon$ for $\epsilon < 1/8$, it must be the case $t \leq i_0 + 2 = O(\log(1/\epsilon))$. □

Note that Line 7 and Line 8 are executed with probability 1 during the first $t-1$ iterations. We show the following.

Lemma 2.3. *For iteration $1 \leq i < t-1$, $\mathbb{E}[\frac{|H_{i+1}|}{n} \mid H_i] = \frac{|H_i|^2}{n^2}$.*

Proof.

$$
\begin{aligned}
\mathbb{E}\left[\frac{|H_{i+1}|}{n} \mid H_i\right] &= \frac{1}{n} \sum_{v \in V} \Pr(x_{t_1(v)} \in H_i \wedge x_{t_2(v)} \in H_i) \\
&= \frac{1}{n} \sum_{v \in V} \left(\frac{|H_i|}{n}\right)^2 = \left(\frac{|H_i|^2}{n^2}\right)
\end{aligned}
$$

□

Therefore, for $1 \leq i \leq t-1$, if $\frac{|H_i|}{n} \sim h_i$ ($\sim$ means they are close), then $\frac{|H_{i+1}|}{n} \sim h_{i+1}$ if $\frac{|H_{i+1}|}{n}$ concentrates around its expectation, since $h_{i+1} = h_i^2$.

In the last iteration, we truncate the probability of doing the tournament by doing it with only probability δ for each node, so that ideally we hope to have $\mathbb{E}[\frac{|H_t|}{n}|H_{t-1}] = T$. However, since δ is calculated with respect to h_{t-1} instead of the actual $\frac{|H_{t-1}|}{n}$, we need the following lemma to bound the deviated expectation. In the next lemma, we show that if $\frac{|H_{t-1}|}{n} \sim h_{t-1}$, then indeed we have $\mathbb{E}[\frac{|H_t|}{n}|H_{t-1}] \sim T$.

Lemma 2.4. *Suppose that $(1-\epsilon'')h_{t-1} \leq \frac{|H_{t-1}|}{n} \leq (1+\epsilon'')h_{t-1}$ for some $0 < \epsilon'' < 1$. We have $T - 3\epsilon'' \leq \mathbb{E}\left[\frac{|H_t|}{n} \mid H_{t-1}\right] \leq T + 3\epsilon''$.*

In the end, we hope that $\frac{|H_t|}{n}$ deviates from T by at most $\epsilon/2$. To achieve this, we show that in each iteration $1 \leq i \leq t-1$, $\frac{|H_i|}{n}$ deviates from its expectation, $\frac{|H_{i-1}|^2}{n^2}$, by at most a $(1 \pm \epsilon')$ factor, where $\epsilon' = \epsilon/2^{t+4}$ is an error control parameter that is much less than ϵ. Note that $\frac{|H_{i-1}|^2}{n^2}$ is already deviated from $h_{i-1}^2 = h_i$ to some degree. The next lemma bounds the cumulative deviation of $\frac{|H_i|}{n}$ from h_i. Note that ϵ' has to be large enough in order guarantee that $\frac{|H_i|}{n}$ lies in the $(1 \pm \epsilon') \cdot \frac{|H_{i-1}|^2}{n^2}$ range. This also implies ϵ has to be large enough.

Lemma 2.5. *Let $\epsilon = \Omega(1/n^{1/4.47})$ and $\epsilon' = \frac{\epsilon}{2^{t+4}}$. W.h.p. for iteration $0 \leq i < t$, we have $\frac{|H_i|}{n} \in (1 \pm \epsilon')^{2^{i+1}-1} \cdot h_i$.*

Combining Lemma 2.4 for the deviation on the last round and Lemma 2.5 for the deviation on first $t-1$ rounds, the following lemma summarizes the final deviation.

Lemma 2.6. *Let $\epsilon = \Omega(1/n^{1/4.47})$. At the end of the algorithm, w.h.p. $T - \frac{\epsilon}{2} \leq \frac{|H_t|}{n} \leq T + \frac{\epsilon}{2}$.*

Proof. Let $\epsilon' = \frac{\epsilon}{2^{t+4}}$. By Lemma 2.5, we have $\frac{|H_{t-1}|}{n} \in (1 \pm \epsilon')^{2^t-1} \cdot h_{t-1}$ w.h.p.

Note that $(1+\epsilon')^{2^t-1} \leq 1 + 2(2^t-1)\epsilon' \leq 1 + 2^{t+1}\epsilon'$, since $(1+a)^b \leq 1 + 2ab$ provided $0 < ab \leq 1/2$ and we have $(2^t-1)\epsilon' \leq \epsilon/16 \leq 1/2$. Similarly, $(1-\epsilon')^{2^t-1} \geq 1 - 2^{t+1}\epsilon'$. Therefore, we can let $\epsilon'' = 2^{t+1}\epsilon'$ and apply Lemma 2.4 to conclude that

$$
T - 3 \cdot 2^{t+1}\epsilon' \leq T - 3\epsilon'' \leq \mathbb{E}\left[\frac{|H_t|}{n} \mid H_{t-1}\right] \leq T - 3\epsilon'' \leq T + 3 \cdot 2^{t+1}\epsilon'
$$

Since $\frac{|H_{t-1}|}{n} = \Omega(1)$ (by Lemma 2.5) and $\epsilon' = \epsilon/2^{t+4} = \Omega(\sqrt{\log n/n})$. we can apply Chernoff Bound to show w.h.p. $|H_t| \in (1 \pm \epsilon)\mathbb{E}[|H_t| \mid H_{t-1}]$. Thus,

$$
\begin{aligned}
(1-\epsilon')(T - 3 \cdot 2^{t+1}\epsilon') \leq \frac{|H_t|}{n} &\leq (1+\epsilon')(T + 3 \cdot 2^{t+1}\epsilon') \\
T - 4 \cdot 2^{t+1}\epsilon' \leq \frac{|H_t|}{n} &\leq T + 4 \cdot 2^{t+1}\epsilon' \\
T - \frac{\epsilon}{2} \leq \frac{|H_t|}{n} &\leq T + \frac{\epsilon}{2} \qquad \epsilon' = \epsilon/(2^{t+4})
\end{aligned}
$$

□

Initially, $\frac{|M_0|}{n} \geq 2\epsilon$. In a similar vein, we show that that $\frac{|M_t|}{n}$ does not decreases much from 2ϵ.

Lemma 2.7. *Let $\epsilon = \Omega(1/n^{1/4.47})$. At the end of the algorithm, w.h.p. $\frac{|M_t|}{n} \geq \frac{7\epsilon}{4}$.*

The following lemma shows that at the end of Algorithm 1, the problem has been reduced to finding the approximate median.

Lemma 2.8. *Let $\epsilon = \Omega(1/n^{1/4.47})$. At the end of iteration t of Algorithm 1, w.h.p. any ϕ'-quantile where $\phi' \in [\frac{1}{2} - \frac{\epsilon}{4}, \frac{1}{2} + \frac{\epsilon}{4}]$ must be in M_t.*

Proof. Since $\frac{1}{2}-\frac{3\epsilon}{2} \le \frac{|H_t|}{n} \le \frac{1}{2}-\frac{\epsilon}{2}$ and $\frac{|M_t|}{n} \ge \frac{7\epsilon}{4}$ by Lemma 2.5 and Lemma 2.7, we have $\frac{|M_t|}{n}+\frac{|H_t|}{n} \ge \frac{1}{2}+\frac{\epsilon}{4}$. Combined with the fact that $\frac{|H_t|}{n} \le \frac{1}{2}-\frac{\epsilon}{2}$, we conclude that any ϕ'-quantile where $\phi' \in [\frac{1}{2}-\frac{\epsilon}{4}, \frac{1}{2}+\frac{\epsilon}{4}]$ must be in M_t. □

2.2 Phase II: Approximating the Median

Let L_i, M_i, and H_i denote the nodes whose quantiles lie in $[0, \frac{1}{2}-\epsilon)$, $[\frac{1}{2}-\epsilon, \frac{1}{2}+\epsilon]$, and $(\frac{1}{2}-\epsilon, 1]$ respectively at the end of iteration i, and L_0, M_0, and H_0 be the nodes with those quantiles in the beginning. Note that L_i and H_i are the nodes whose values are not our targets. We will show the quantities of $\frac{|L_i|}{n}$ and $\frac{|H_i|}{n}$ decrease in each iteration as our 3-TOURNAMENT algorithm (Algorithm 2) makes progress.

Initially, $l_0 = h_0 = \frac{1}{2}-\epsilon$. Let $h_{i+1} = 3h_i^2 - 2h_i^3$ and $l_{i+1} = 3l_i^2 - 2l_i^3$ for $i \ge 0$, we will show that $\frac{|L_i|}{n}$ and $\frac{|H_i|}{n}$ concentrate around l_i and h_i. Note that h_i and l_i roughly square in each iteration. Once they decrease below a constant after the first $O(\log(1/\epsilon))$ iterations, they decrease double exponentially in each iteration. The tournaments end when l_i and h_i decrease below $T = 1/n^{1/3}$.

Algorithm 2 3-TOURNAMENT(v)

1: $h_0, l_0 \leftarrow \frac{1}{2}-\epsilon$
2: $i \leftarrow 0, T = 1/n^{1/3}$.
3: **while** $l_i > T$ **do**
4: $\quad h_{i+1} \leftarrow 3h_i^2 - 2h_i^3$, $l_{i+1} \leftarrow 3l_i^2 - 2l_i^3$
5: $\quad$ Select three nodes $t_1(v), t_2(v), t_3(v)$ randomly.
6: $\quad x_v \leftarrow median(x_{t_1(v)}, x_{t_2(v)}, x_{t_3(v)})$.
7: **end while**
8: Sample $K = O(1)$ nodes uniformly at random and output the median value of these nodes.

Lemma 2.9. *Let t denote the number of iterations in Algorithm 2. We have $t \le \log_{11/8}(\frac{1}{4\epsilon}) + \log_2\log_4 n = O(\log(1/\epsilon) + \log\log n)$.*

Proof. Suppose that $(\frac{11}{8})^{i-1}\epsilon \le 1/4$, we will show that $l_i \le 1/2 - (\frac{11}{8})^i\epsilon$. First, $l_0 = 1/2-\epsilon$. Suppose by induction that $l_{i-1} \le 1/2-(\frac{11}{8})^{i-1}\epsilon$. We have:

$$\begin{aligned} l_i &= 3l_{i-1}^2 - 2l_{i-1}^3 \\ &= 3\left(\frac{1}{2}-\left(\frac{11}{8}\right)^{i-1}\cdot\epsilon\right)^2 - 2\left(\frac{1}{2}-\left(\frac{11}{8}\right)^{i-1}\cdot\epsilon\right)^3 \\ &= \frac{1}{2}-\frac{3}{2}\left(\frac{11}{8}\right)^{i-1}\cdot\epsilon + 2\left(\left(\frac{11}{8}\right)^{i-1}\cdot\epsilon\right)^3 \\ &\le \frac{1}{2}-\frac{3}{2}\left(\frac{11}{8}\right)^{i-1}\cdot\epsilon + \frac{1}{8}\left(\left(\frac{11}{8}\cdot\epsilon\right)^{i-1}\right) \\ &= \frac{1}{2}-\left(\frac{11}{8}\right)^{i}\cdot\epsilon \end{aligned}$$

Therefore, after $i_0 = \log_{11/8}(\frac{1}{4\epsilon})$ iterations, we have $l_i \le \frac{1}{2} - \left(\frac{11}{8}\right)^{i_0}\epsilon \le \frac{1}{2}-\frac{1}{4} = \frac{1}{4}$.

Suppose that $i_1 = \log_2\log_4 n$, we have

$$\begin{aligned} l_{i_0+i_1} &\le 3l_{i_0+i_1-1}^2 \le 3^{i_1}l_{i_0}^{2^{i_1}} \le 3^{i_1}\cdot\left(\frac{1}{4}\right)^{2^{i_1}} \\ &\le (\log_4 n)^{\log_2 3}\cdot\left(\frac{1}{4}\right)^{\log_4 n} = \frac{(\log_4 n)^{\log_2 3}}{n} \le n^{2/3} \end{aligned}$$

Therefore, $t \le i_0 + i_1 = \log_{11/8}(\frac{1}{4\epsilon}) + \log_2\log_4 n = O(\log(1/\epsilon) + \log\log n)$. □

Lemma 2.10. *For each iteration $0 \le i < t$, $\mathbb{E}[\frac{|L_{i+1}|}{n} \mid L_i] = 3(\frac{|L_i|}{n})^2 - 2(\frac{|L_i|}{n})^3$ and $\mathbb{E}[\frac{|H_{i+1}|}{n} \mid H_i] = 3(\frac{|H_i|}{n})^2 - 2(\frac{|H_i|}{n})^3$.*

Proof. We will show the proof for $\mathbb{E}[\frac{|L_{i+1}|}{n}|L_i]$, since $\mathbb{E}[\frac{|H_{i+1}|}{n} \mid H_i]$ is the same. Note that a node v is in L_{i+1} if and only if at least 2 of $t_1(v)$, $t_2(v)$, $t_3(v)$ are in L_i. Therefore,

$$\begin{aligned} \mathbb{E}\left[\frac{|L_{i+1}|}{n} \mid L_i\right] &= \frac{1}{n}\sum_{v\in V}\left(\left(\frac{|L_i|}{n}\right)^3 + 3\left(\frac{|L_i|}{n}\right)^2\left(1-\frac{|L_i|}{n}\right)\right) \\ &= 3\left(\frac{|L_i|}{n}\right)^2 - 2\left(\frac{|L_i|}{n}\right)^3 \end{aligned}$$

□

The following lemma shows the probabilities that $\frac{|L_i|}{n}$ and $\frac{|H_i|}{n}$ deviate from their expectation by a $(1+\epsilon')$ factor are polynomially small in n, provided ϵ' is sufficiently large. We cap the quantity at T for the purpose of applying concentration inequalities.

Lemma 2.11. *Let $\epsilon' = \Omega(\frac{(\log n)^{1/2}}{n^{1/3}})$, w.h.p. for each iteration $0 \le i <= t$, $\frac{|H_i|}{n} \le (1+\epsilon')\cdot\max(T, \mathbb{E}[\frac{|H_i|}{n} \mid H_{i-1}])$ and $\frac{|L_i|}{n} \le (1+\epsilon')\cdot\max(T, \mathbb{E}[\frac{|L_i|}{n} \mid L_{i-1}])$.*

Proof. In each iteration, since each node set its value independently, by Chernoff Bound (see Lemma A.2), we have

$$\begin{aligned} &\Pr\left(\frac{|L_i|}{n} \le (1+\epsilon')\cdot\max(T, \frac{\mathbb{E}[|L_i| \mid L_{i-1}]}{n})\right) \\ &\le \exp(-\Omega(\epsilon'^2\max(n\cdot T, \mathbb{E}[|L_i| \mid L_{i-1}]))) \\ &= \exp(-\Omega(\epsilon'^2 n^{2/3})) = 1/\mathrm{poly}(n) \end{aligned}$$

The proof for $|H_i|$ is the same. □

Then we bound the cumulative deviation of $\frac{|H_i|}{n}$ from h_i and the cumulative deviation of $\frac{|L_i|}{n}$ from l_i.

Lemma 2.12. *Let $\epsilon' = \Omega(\frac{\log n^{1/2}}{n^{1/3}})$. W.h.p. for each iteration $0 \le i <= t$, $\frac{|L_i|}{n} \le \max((1+\epsilon')^{\frac{3^i-1}{2}}\cdot l_i, (1+\epsilon')T)$ and $\frac{|H_i|}{n} \le \max((1+\epsilon')^{\frac{3^i-1}{2}}\cdot h_{i+1}, (1+\epsilon')T)$.*

Proof. We only show the proof for $|L_i|$ and we will prove by induction. Initially, $\frac{|L_0|}{n} \le l_0$. Suppose that $\frac{|L_i|}{n} \le \max((1+\epsilon')^{\frac{3^i-1}{2}}\cdot l_{i+1}, (1+\epsilon')T)$ is true.

By Lemma 2.11, we have w.h.p., $\frac{|L_{i+1}|}{n} \le (1+\epsilon')\cdot\max(\frac{\mathbb{E}[|L_{i+1}||L_i]}{n}, T)$. If $\frac{\mathbb{E}[|L_{i+1}||L_i]}{n} \le T$, then we are done, since $\frac{|L_{i+1}|}{n} \le (1+\epsilon')\cdot\frac{\mathbb{E}[|L_{i+1}||L_i]}{n} = (1+\epsilon')\cdot T$. Also, if $\frac{|L_i|}{n} \ge (1+\epsilon')T$, then $\frac{\mathbb{E}[|L_{i+1}||L_i]}{n} \le 3\cdot\frac{|L_i|^2}{n^2} \le \frac{|L_i|}{n}\cdot O(\frac{1}{n^{2/3}}) \le T$.

Otherwise, if $\frac{|L_i|}{n} > (1+\epsilon')T$ then it must be the case that $\frac{|L_i|}{n} \le (1+\epsilon')^{\frac{3^i-1}{2}} \cdot l_{i+1}$ by induction hypothesis. Therefore,

$$\begin{aligned}
\frac{|L_{i+1}|}{n} &\le (1+\epsilon') \cdot \frac{\mathbb{E}[|L_i| \mid L_{i-1}]}{n} \\
&\le (1+\epsilon') \cdot 3\left(\frac{|L_i|}{n}\right)^2 - 2\left(\frac{|L_i|}{n}\right)^3 \\
&\le (1+\epsilon') \cdot \left(3\left((1+\epsilon')^{\frac{3^i-1}{2}} \cdot l_i\right)^2 - 2\left((1+\epsilon')^{\frac{3^i-1}{2}} \cdot l_i\right)^3\right) \\
&\le (1+\epsilon') \cdot \left(\left((1+\epsilon')^{\frac{3^{i+1}-3}{2}}\right) \cdot \left(3 \cdot l_i^2 - 2 \cdot l_i^3\right)\right) \\
&= \left((1+\epsilon')^{\frac{3^{i+1}-1}{2}}\right) \cdot l_{i+1}
\end{aligned}$$

□

Now, by setting ϵ' roughly equal to $1/3^t \sim \epsilon^{3.45}/(\log_4 n)^{1.59}$, we can bound the final deviations of $\frac{|H_i|}{n}$ and $\frac{|L_i|}{n}$ from T by a factor of 2 w.h.p.

Lemma 2.13. *Let $\epsilon = \Omega(\frac{\log^{0.61} n}{n^{0.096}})$ and $\epsilon' = \epsilon^{3.45}/(\log_4 n)^{1.59}$, then $\frac{|L_t|}{n} \le 2T$ and $\frac{|H_t|}{n} \le 2T$ w.h.p.*

Proof. First, $\epsilon' = \Omega(\frac{\epsilon^{3.45}}{\log^{1.59} n}) = \Omega(\frac{\log^{1/2} n}{n^{1/3}})$. By Lemma 2.12, w.h.p. we have either $\frac{|L_t|}{n} \le (1+\epsilon') \cdot T$ or $\frac{|L_t|}{n} \le (1+\epsilon')^{\frac{3^i-1}{2}} \cdot l_t$. If it is the former, then we are done since $\frac{|L_t|}{n} \le (1+\epsilon') \cdot T \le 2T$. Otherwise, we have

$$\begin{aligned}
\frac{|L_t|}{n} &\le \left((1+\epsilon')^{\frac{3^t-1}{2}}\right) \cdot T \\
&\le \left(1 + 2 \cdot \epsilon' \cdot \frac{3^t-1}{2}\right) \cdot T \\
&\qquad\qquad (1+x)^n \le 1 + 2nx \text{ for } nx \le \tfrac{1}{2} \\
&\le \left(1 + \epsilon' \cdot 3^{\log_{11/8}(\frac{1}{4\epsilon}) + \log_2 \log_4 n}\right) \cdot T \\
&\le \left(1 + \epsilon' \cdot \left(\frac{1}{4\epsilon}\right)^{\log_{11/8} 3} \cdot \log_4^{\log_2 3} n\right) \cdot T \\
&\le 2 \cdot T \qquad\qquad \epsilon' = \epsilon^{3.45}/(\log_4 n)^{1.59}
\end{aligned}$$

□

Finally, we show that when $\frac{|L_t|}{n}$ and $\frac{|H_t|}{n}$ are $O(1/n^{2/3})$, if we sample a constant number of values randomly and output the median of them, then w.h.p. the median is in M_t.

Lemma 2.14. *W.h.p. every node outputs a quantile in $[\frac{1}{2}-\epsilon, \frac{1}{2}+\epsilon]$.*

Proof. By Corollary 2.13, w.h.p. at the end of iteration t, $\frac{|L_t|}{n} \le 2/n^{2/3}$ and $\frac{|H_t|}{n} \le 2/n^{2/3}$. The algorithm outputs a quantile in $[\frac{1}{2}-\epsilon, \frac{1}{2}+\epsilon]$ if there are less than $K/2$ nodes in L_t are sampled and less than $K/2$ nodes in H_t are sampled.

The probability that at least $K/2$ nodes in $|L_t|$ are sampled is at most

$$\binom{K}{K/2} \cdot \left(\frac{2}{n^{2/3}}\right)^{K/2} \le \left(\frac{eK}{\frac{K}{2}}\right)^{K/2} \cdot \left(\frac{2}{n^{2/3}}\right)^{K/2} \le \left(\frac{4e}{n^{2/3}}\right)^{K/2}$$

Similarly, the probability that at least $K/2$ nodes in $|H_t|$ are sampled is also at most $\left(\frac{4e}{n^{2/3}}\right)^{K/2}$. By an union bound, the probability that less than $K/2$ nodes in L_t are sampled and less than $K/2$ nodes in H_t are sample is at least $1 - 2 \cdot \left(\frac{4e}{n^{2/3}}\right)^{K/2} = 1 - 1/\text{poly}(n)$. □

Theorem 2.1 follows from Lemma 2.8 and Lemma 2.14.

3 EXACT QUANTILE COMPUTATION

To fill the gap of approximating the ϕ-quantile with ϵ error for $\epsilon = O(1/n^{0.096})$, we show that the exact quantile computation can be done in $O(\log n)$ rounds using $O(\log n)$ message size. Since we are to compute the exact quantile ϕ, we can assume that $k_0 \stackrel{\text{def}}{=} \phi \cdot n$ is an integer. The problem is to compute a value whose rank is k_0. Again, w.l.o.g. we assume that every node has a distinct value initially.

Algorithm 3 Exact Quantile Computation

1: $k_0 \leftarrow \phi \cdot n$
2: **for** $i = 1, 2, \ldots, 25$ **do**
3: Each node v computes an $\frac{\epsilon}{2}$-approximate of the $(\frac{k_{i-1}}{n} - \frac{\epsilon}{2})$-quantile and an $\frac{\epsilon}{2}$-approximate of the $(\frac{k_{i-1}}{n} + \frac{\epsilon}{2})$-quantile with $\epsilon = n^{-0.05}/2$.
4: Each node learns the max and the min of these approximates of all nodes.
5: Compute the rank of min among the original x_v's and denote it by R.
6: Each node v set $x_v \leftarrow \infty$ if $x_v \notin [\min, \max]$. Call these nodes *valueless*, otherwise *valued*.
7: Let m_i be the smallest power of 2 that is larger than $(n^{0.99}/2)/(\#\text{ valued nodes})$. Each valued node makes m_i copies of its value and distribute them to valueless nodes so that there are at least $n^{0.99}/2$ valued nodes. (For convenience, we let the duplicated values to have smaller ranks than the original one.)
8: Set $k_i \leftarrow m_i \cdot (k_{i-1} - R + 1)$.
9: **end for**
10: Every node outputs an $(\epsilon/3)$-approximate $(\frac{k_{25}}{n} - \frac{\epsilon}{2})$-quantile.

The following is a detailed implementation of each step.

Step 3: Since $\epsilon = n^{-0.05}/2$, we can $\frac{\epsilon}{2}$-approximate the quantiles in $O(\log n)$ rounds by Theorem 2.1.

Step 4: The maximum (the minimum) can be computed by having each node forwarding the maximum (the minimum) value it has ever received. Since it takes $O(\log n)$ rounds to spread a message by [14, 32], this step can be done in $O(\log n)$ rounds.

Step 5: The rank of the minimum can be computed by performing a counting. The nodes whose x_v values are less than or equal to the minimum will be assigned 1. Otherwise they are assigned 0. By Kempe et al. [24], the sum can be aggregated in $O(\log n)$ rounds w.h.p.

Step 7: Consider the following process for distributing the values. Initially, every valued node v generates a token with weight equal to m_i, the targeted number of copies. We denote the token by a value-weight pair (x_v, m_i). Recall that m_i is a power of 2 and it can

be computed by counting the number of valued nodes in $O(\log n)$ rounds. The goal is to split and distribute the tokens so that every node has at most one token of weight 1 in the end.

The process consists of $O(\log n)$ phases and each phase uses $O(1)$ rounds. Suppose that node v holds tokens $(x_1, w_1), (x_2, w_2), \ldots (x_m, w_m)$ at the beginning of a phase. For each token (x_i, w_i), if $w_i \neq 1$, v splits (x_i, w_i) into two tokens $(x_i, w_i/2)$ and push each one to a node randomly. If $w_i = 1$, then the token remains at v. Note that when two tokens of the same value are on the same node, they do not merge.

First note $\lg m_i = O(\log n)$ phases are needed to split to tokens into tokens of weight 1. Now, we show that it takes constant number of rounds to implement one phase.

Let $N(v, i)$ denote the number of tokens at v at the end i'th phase. Let $N = \max_{v,i} N(v, i)$. It takes N rounds to implement one phase. The value of $N(v, i)$ can be bounded by following calculation. Since at any phase, there cannot be more than $n^{0.99}$ tokens and each token appears at a node chosen uniformly at random, we have

$$\Pr(N(v,i) \geq 100K) \leq \binom{n^{0.99}}{100K} \cdot \frac{1}{n^{100K}} \leq \left(\frac{n^{0.99}}{n}\right)^{100K} = (1/n)^K.$$

By taking an union bound over each v and i, we conclude that $N < 100K$ w.h.p. Therefore, w.h.p. each phase can be implemented in $200K = O(1)$ rounds. At the end of $O(\log n)$ phase, every node has at most $100K$ tokens of weight 1 w.h.p.

Next, in the subsequent phases, if a node holds more than one token, then it will push every token except one to a random node (one random node per token). We say a token succeeded at the end of the phase if it was pushed to a node without any other tokens. Consider a token, the probability a token did not succeed in a phase is at most $n^{0.99}/n = 1/n^{0.01}$, since there are at most $n^{0.99}$ nodes with tokens at the end of the phase.

Therefore, after $100K$ phases, the probability that a token did not succeed is at most $1/n^{0.01 \cdot 100K} = 1/n^K$. By a union bound over the tokens, we conclude that w.h.p. all tokens succeeded. Moreover, again, each phase can be implemented using $100K$ rounds. Therefore, we can split and distribute the tokens in $O(\log n)$ rounds. Each original token is duplicated with exactly m_i copies. Then, those nodes holding a token set its value to the value of the token. Nodes without a token will remain valueless.

Correctness: Let ans be the answer (the value whose initial rank is k_0). Let $M_i = \prod_{j=1}^{i} m_i$ denote the number of copies of each value from the beginning. We show by induction that the values whose ranks lying in $(k_i - M_i, k_i]$ are ans after iteration i. Initially, $M_0 = 1$, the statement trivially holds.

Suppose that at the end of iteration $i-1$, the values whose ranks lying in $(k_{i-1} - M_i, k_{i-1}]$ are ans. After Step 6 in iteration i, the rank of ans is exactly $k_{i-1} - R + 1$. Since every value is duplicated to have m_i copies after Step 7, the rank of ans becomes $k_i = m_i \cdot (k_{i-1} - R + 1)$ (Recall that we let the original value to have a larger rank than the duplicated values). Since the values whose ranks lying in $(k_{i-1} - M_{i-1}, k_{i-1}]$ were ans at the end of iteration $i-1$ and ans $\in [\min, \max]$, it must be the case that every value in this range is duplicated with m_i copies. Therefore, all values whose ranks lying in $(k_i - m_i \cdot M_{i-1}, k_i] = (k_i - M_i, k_i]$ are ans after iteration i.

Next, we show that after 25 iterations, $M_{25} \geq \epsilon n$. Suppose that $M_{i-1} \leq \epsilon n$. The number of valueless node after Step 6 is at most $2\epsilon n + 2M_{i-1} \leq 4\epsilon n$. The $2\epsilon n$ comes from the fact that at most $2\epsilon n$ values can lie in the range (min, max). The $2M_{i-1}$ term is because at most M_{i-1} values are equal to min and max. Therefore, $m_i \geq (n^{0.99}/2)/(4\epsilon n) = (n^{0.04}/4)$ and $M_i \geq (n^{0.04}/4) \cdot M_{i-1}$.

Therefore, $M_{25} \geq \min(\epsilon n, (n^{0.04}/4)^{25}) = \epsilon n$. Now the the values of the items whose ranks are in the range of $(k_{25} - \epsilon n, k_{25}]$ must be ans. We then compute a quantile in $(\frac{k_{25}}{n} - \epsilon, \frac{k_{25}}{n}]$ using our $\epsilon/3$-approximate quantile computation algorithm to approximate the $(\frac{k_{25}}{n} - \frac{\epsilon}{2})$-quantile in $O(\log n)$ rounds.

4 A LOWER BOUND

We show that it takes at least $\Omega(\log(1/\epsilon) + \log\log n)$ rounds to approximate the ϕ-quantile up to ϵ error w.h.p.

Theorem 1.4. *Suppose in every round every node fails with a, potentially different, probability bounded by some constant $\mu < 1$. For any $\phi \in [0, 1]$ there is a gossip algorithm that solves the ϕ quantile problem in $O(\log n)$ rounds. For any t and any $\epsilon(n) > 0$, there furthermore exists a gossip algorithm that solves the ϵ-approximate ϕ-quantile problem in $O(\log\log n + \log \frac{1}{\epsilon(n)} + t)$ rounds for all but $\frac{n}{2^t}$ nodes, with high probability.*

Proof. Consider the following two scenarios. The first is when each node is associated with a distinct value from $\{1, 2, \ldots, n\}$. The second is when each node is associated with a distinct value from $\{1 + \lfloor 2\epsilon n \rfloor, \ldots, n + \lfloor 2\epsilon n \rfloor\}$.

A node is able to distinguish between these cases only if it recieves an value from $S \stackrel{\text{def}}{=} \{1, 2, \ldots, \lfloor 2\epsilon n \rfloor\} \cup \{n+1, n+2, \ldots, n + \lfloor 2\epsilon n \rfloor\}$. Otherwise, it can only output a value whose quantile is in $[1/2 - \epsilon, 1/2 + \epsilon]$ with probability 1/2, since the difference of the ϕ-quantile in both scenarios is at least $\lfloor 2\epsilon n \rfloor \geq \epsilon n$.

Call a node *good* if it ever received a value from S, and *bad* otherwise. Note that a bad node cannot outputs a correct answer with probability more than 1/2. Initially there are at most $2 \cdot \lfloor 2\epsilon n \rfloor \leq 4\epsilon n$ good nodes. We will show that with probability $1 - 1/n$ there exists at least one bad node at the end of round t.

Let $X_0 = 2 \cdot \lfloor 2\epsilon n \rfloor$ and let X_i denote the number of good nodes at the end of round i. Given a bad node v, it can become good if it pulls from a good node or some good node pushes to it. Let Y_v denote the event that v pulls from a good node, we have $\Pr(Y_v \mid X_i) = X_i/n$. Also, the pushes from the good nodes can only generate another at most X_i good nodes. Therefore,

$$\mathbb{E}[X_{i+1} \mid X_i] \leq 2X_i + \mathbb{E}\left[\sum_{v \in B_i} Y_v\right] \leq 3X_i$$
since $\mathbb{E}[\sum_{v \in B_i} Y_v \mid X_i] \leq X_i$

By Chernoff Bound, we have $\Pr(\sum_{v \in B_i} Y_v > 2X_i \mid X_i) \leq e^{-X_i/2} \leq e^{-5\log n} \leq 1/n^5$, since $X_i \geq X_0 \geq 10 \log n$. Therefore, with probability at least $1 - 1/n^5$, $X_{i+1} \leq 4X_i$. By taking a union bound over such events for the first $t' = \log_4(8/\epsilon)$ rounds, we conclude with probability at least $1 - 1/n^4$, $X_{t'} \leq (4\epsilon n)4^{t'} \leq n/2$.

Let t_0 be the last round such that $X_{t_0} \leq n/2$. Define $Z_i = |B_i|/n$. A node v remains in B_i if it did not pull from a good node **and** it

was not pushed from any good nodes. Denote the event by W_v, we have $\Pr(W_v \mid B_i) \geq Z_i \cdot (1-\frac{1}{n})^{n-1} \geq Z_i \cdot e^{-1}$. Therefore,

$$\mathbb{E}[|B_{i+1}| \mid B_i] = \mathbb{E}\left[\sum_{v \in B_i} W_v \mid B_i\right] \geq \left(\sum_{v \in B_i} Z_i \cdot e^{-1}\right) = |B_i| \cdot Z_i \cdot e^{-1}$$

Note that the events $\{W_v\}_{v \in B_i}$ are negatively dependent [8] (the number of empty bins in the balls into bins problem). Suppose that $Z_i \cdot |B_i| \geq 60e \log n$. By Chernoff Bound,

$$\Pr\left(|B_{i+1}| \leq |B_i| \cdot Z_i/(2e) \mid B_i\right) \leq e^{-Z_i \cdot |B_i|/(12e)} \leq 1/n^5$$

Therefore, $\Pr(Z_{i+1} \leq Z_i^2/(2e) \mid Z_i) \leq 1/n^5$. Suppose that $t_1 = O(n)$. We can take an union bound over the subsequent t_1 rounds to show that with probability at least $1 - 1/n^4$, $Z_{i+1} \geq Z_i^2/(2e)$ holds for these rounds, as long as $Z_{t_0+t_1}^2 \geq (60 \log n)/n$.

Let $z_{t_0} = 1/2$ and $z_{i+1} = z_i^2/(2e)$. If $z_{t_0+t_1}^2 \geq (60 \log n)/n$, with probability at least $1 - 1/n^4$, we have $Z_{t_0+t_1} \geq z_{t_0+t_1}$. Let $t_1 = \frac{1}{2} \lg \lg n$, by definition of z_i, we have

$$\begin{aligned}
z_{t_0+t_1} &= \left(\frac{1}{2e}\right)^{t_1} \cdot z_{t_0}^{2^{t_1}} \\
&\geq \left(\frac{1}{2e}\right)^{t_1+2^{t_1}} && z_{t_0} = 1/2 \geq 1/(2e) \\
&= \left(\frac{1}{2e}\right)^{(\lg \lg n)/2+\sqrt{\log n}} \\
&\geq \left(\frac{1}{2e}\right)^{2\sqrt{\log n}} && \text{for sufficiently large } n
\end{aligned}$$

Since $(2e)^{-2\sqrt{\log n}} = \Omega((\log n)/n)$, $z_{t_0+t_1} \geq 2^{-2\sqrt{\log n}} \geq (60 \log n)/n$ for sufficiently large n. Therefore, with probability at least $1 - 1/n^4$, $Z_{t_0+t_1} > 0$.

Since with probability $1 - 1/n^4$, $t_0 \geq \log_4 8/\epsilon$. By taking an union bound over these two events, we conclude with probability at most $2/n^4$, all nodes are good by the end of round $t_0 + t_1$.

Therefore, the probability that every node computes a correct output at the end of round $t_0 + t_1$ is at most $1/2 + 2/n^4$. □

5 ROBUSTNESS

In this section, we show that our algorithm is robust against failures. We consider the following model of failures: Let $0 < \mu < 1$ be a constant. Every node v in every round i is associated with a probability $0 \leq p_{v,i} \leq \mu$. Note that $p_{v,i}$ is pre-determined before the execution of the algorithm. During the execution of round i, each node v fails with probability $p_{v,i}$ to perform its operation (which may be either push or pull). We show how to modify the algorithm and the analysis for the tournament algorithms and the exact quantile computation algorithms in the following.

5.1 The Tournament Algorithms

First consider the 2-TOURNAMENT algorithm. Initially, every node is *good*. Consider node v, instead of only pulling from 2 neighbors each iteration, now pull from $\Theta(\frac{1}{1-\mu} \cdot \log(\frac{1}{1-\mu}))$ neighbors. We say a pull is *good* if the node performing the pull operation did not fail **and** it pulls from a node who is good at the end of iteration $i-1$. A node remains good at the end of iteration i if there are at least two good pulls. Then, v uses the first two good pulls to execute the tournament procedure. Also, note that in the last iteration, we let a good node to have probability of δ to do the two tournament using the first two good pulls and probability $1-\delta$ to set the value equal to the first good pull. We show that the good nodes always consists of a constant fraction of nodes.

Lemma 5.1. *For $0 \leq i \leq t-1$, w.h.p. at the end of each iteration i of* 2-TOURNAMENT *, there are at least $n/2$ good nodes if every node pulls from $k = \Theta(\frac{1}{1-\mu} \cdot \log(\frac{1}{1-\mu}))$ other nodes.*

Proof. We will prove by induction. Initially, every node is good. Suppose that there are at least $n/2$ good nodes at the end of iteration $i-1$. During iteration i, a node with more than $k-2$ bad pulls will become bad at the end of iteration i.

Let $k = \frac{4}{1-\mu} \log \frac{4}{1-\mu} + 1$. The probability that there are at least $k-1$ bad pulls is at most:

$$\begin{aligned}
\binom{k}{k-1} \cdot \left(1 - \frac{(1-\mu)}{2}\right)^{k-1} &\leq k \cdot e^{-\frac{(1-\mu)}{2} \cdot (k-1)} \\
&\leq \left(\frac{4}{1-\mu} \log \frac{4}{1-\mu} + 1\right) \cdot \left(\frac{1-\mu}{4}\right)^2 \\
&\leq 1/e + 1/16 \leq 0.44 \\
&\qquad x \log(1/x) \text{ maximized at } 1/e
\end{aligned}$$

Therefore, the expected number of bad nodes is at most $0.44n$ for iteration $1 \leq i \leq t-1$. Since each node becomes bad independently of other nodes, by Chernoff Bound, w.h.p. there are at most $0.5n$ bad nodes at the end of iteration i. □

At the end of iteration $t-1$, there are at least $n/2$ good nodes w.h.p. The expected number of bad nodes at the end of iteration t is at most $\sum_{v \in V} \delta \cdot 0.44 + (1-\delta)/2 \leq 0.5n$. By Chernoff Bound again, we can conclude that w.h.p. there are at least $n/3$ good nodes.

We can also modify the process in the same way for the 3-TOURNAMENT algorithm. That is, in each iteration each node pulls from $\Theta(\frac{1}{1-\mu} \log \frac{1}{1-\mu})$ other nodes. If there are less than 3 good pulls, then the node becomes bad. Otherwise, it uses the first 3 good pulls to do the tournament procedure. We can show similarly that there are at most a constant fraction bad nodes in each iteration.

Consider a node v in the modified processes. Suppose that v is good in iteration i, then v must have at least two (or three) good pulls. Note that the probability that v pulls from a particular good node, conditioned v is good at the end of iteration, is uniform among all nodes that are good at the end of iteration $i-1$.

Let V_i denote the set of good nodes and $n_i = |V_i|$. Given any subset of good nodes $S \subseteq V_{i-1}$, the probability of choosing a node in $|S|$ is therefore $|S|/n_i$. Therefore, we can replace $\frac{|L_i|}{n}$, $\frac{|H_i|}{n}$, and $\frac{|M_i|}{n}$ in our proofs in the previous section with $\frac{|L_i|}{n_i}$, $\frac{|H_i|}{n_i}$, and $\frac{|M_i|}{n_i}$ and observe that all the statements hold. Note that $n_i \geq n/3 = \Omega(n)$ for $0 \leq i \leq t$ w.h.p. Thus, all the concentration inequalities also hold.

In the last step of Algorithm 2, all nodes pulls from $\Theta(\frac{K}{1-\mu} \log \frac{K}{1-\mu})$ nodes. If there are K good pulls, then each node outputs the median directly. Otherwise, it becomes bad and outputs nothing. We can similarly show that there are at least constant

fraction of good nodes. Therefore, at least a constant fraction of nodes output a correct answer and all the others output nothing. Note that we can use additional $O(t)$ rounds of pulling from valueless nodes to have all but $\frac{n}{2^t}$ nodes learn a correct answer.

5.2 Exact Quantile Computation

Consider Algorithm 3. We know Step 3 (Section 5.1), Step 4 [9], and Step 5 [24] tolerate such failures with a constant factor delay on the running time.

The only step that remains to discuss is Step 7. We run the same algorithm described. Initially each valued node v generates a token (x_v, m_i). In the first $O(\log n)$ phases, each node tries to split every token whose weight is larger than 1 into two tokens with halved weight, and push one of them to a random node. If the push succeed, the token has been pushed is considered a *new token* and the one with halved weight remaining at the same node is considered as the *old one*. If the push fails, then we will merge them back to the original one.

We will show that at the end of $O(\log n)$ phase, each node holds at most $O(1)$ tokens and their weights are 1. In each subsequent phases, if a node has more than one tokens, then it will push every token except one to a random neighbor. We argue that after $O(\log n)$ phases, every node contains exactly one token.

First, we show that the number of tokens at each node is bounded by a constant w.h.p. so that each phase can be implemented in $O(1)$ rounds. Recall that $N(v, i)$ is the number of tokens at v at the end of i'th phase. Suppose the total number of phases is $C \log n$ for some constant $C > 0$. For a token (except the one that is initially at v) to be at node i, it must be the case that it was pushed to i during one of the phase. Note that at any phase, there are at most $n^{0.99}$ tokens. By taking an union bound over all possible set of tokens of size $200K$ and all possible combinations of phases on when each of the token was pushed to i, we have

$$\Pr(N(v,i) \geq 200K+1) \leq \binom{n^{0.99}}{200K} \cdot (C\log n)^{200K} \cdot \frac{1}{n^{200K}} \leq \left(\frac{Cn^{0.99}\log n}{n}\right)^{200K} \leq 1/n^K.$$

Next, we show that for a token (x_v, m_i), it takes $O(\log n)$ rounds to split into tokens of weight 1 w.h.p. Let $T_v(i)$ denote the set of all v's tokens with weight at least 2 at the end of Phase i. Let $\Phi(i) = \sum_{(x_v,w)\in T_v(i)} w^2$ be a potential function. Consider a token $(x_v, w) \in T_v(i)$, since with probability at most μ it fails to split, the expected contribution of the token to the $\Phi(i+1)$ is at most:

$$\mu w^2 + (1-\mu)2\cdot(w/2)^2 = \left(1 - \frac{1-\mu}{2}\right)\cdot w^2$$

Therefore, $\mathbb{E}[\Phi(i+1) \mid \Phi(i)] \leq (1-(1-\mu)/2)\cdot\Phi(i)$. Since $\Phi(0) \leq n$, after $t = \frac{2K}{1-\mu}\log n$ rounds, we have

$$\mathbb{E}[\Phi(t)] \leq \left(1 - \frac{1-\mu}{2}\right)^{\frac{2K}{1-\mu}\log n} \cdot n \leq e^{-K\log n}\cdot n \leq 1/n^{K-1}$$

Since $\Phi(t)$ must be an integer, we conclude that with probability at least $1 - 1/n^{K-1}$, $\Phi(t) = 0$ by Markov's inequality. Therefore, after t rounds w.h.p. every token of value v has weight 1. We can further take an union over all tokens to show that it holds for all tokens w.h.p.

Therefore, w.h.p. the weight of all tokens are 1 at the end of $O(\log n)$ round. In the subsequent phases, the probability a token fail to be pushed to a node without any other tokens is at most $\mu + n^{0.99}/n = O(1)$. Therefore, the probability a token fails in the all of the next $O(\log n)$ phases is at most $1/\text{poly}(n)$. Thus, w.h.p. after $O(\log n)$ phases, every node has at most one token. Since each phase can be implemented using $O(1)$ rounds, we conclude that Step 7 can be done in $O(\log n)$ rounds in our failure model.

ACKNOWLEDGEMENT

We thank Lewis Tseng and the anonymous reviewers for helpful suggestions and comments. We also thank Frederik Mallmann-Trenn for pointing out [7] to us.

REFERENCES

[1] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. 2013. Mergeable Summaries. *ACM Trans. Database Syst.* 38, 4, Article 26 (2013), 28 pages.
[2] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. 1997. A One-Pass Algorithm for Accurately Estimating Quantiles for Disk-Resident Data. In *Proc. 23rd Int'l Conference on Very Large Data Bases (VLDB)*. 346–355.
[3] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. 1973. Time Bounds for Selection. *J. Comput. Syst. Sci.* 7, 4 (1973), 448–461.
[4] Jen-Yeu Chen and Gopal Pandurangan. 2012. Almost-Optimal Gossip-Based Aggregate Computation. *SIAM J. Comput.* 41, 3 (2012), 455–483.
[5] Francis Chin and H. F. Ting. 1987. An improved algorithm for finding the median distributively. *Algorithmica* 2, 1 (1987), 235–249.
[6] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. 1987. Epidemic Algorithms for Replicated Database Maintenance. In *Proc. 6th ACM Symposium on Principles of Distributed Computing (PODC)*. 1–12.
[7] Benjamin Doerr, Leslie Ann Goldberg, Lorenz Minder, Thomas Sauerwald, and Christian Scheideler. 2011. Stabilizing Consensus with the Power of Two Choices. In *Proc. 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 149–158.
[8] Devdatt Dubhashi and Desh Ranjan. 1998. Balls and bins: A study in negative dependence. *Random Structures & Algorithms* 13, 2 (1998), 99–124.
[9] R. Elsässer and T. Sauerwald. 2009. On the Runtime and Robustness of Randomized Broadcasting. *Theor. Comput. Sci.* 410, 36 (2009), 3414–3427.
[10] Ofer Feinerman, Bernhard Haeupler, and Amos Korman. 2017. Breathe before speaking: efficient information dissemination despite noisy, limited and anonymous communication. *Distributed Computing* 30, 5 (2017), 339–355.
[11] David Felber and Rafail Ostrovsky. 2015. A Randomized Online Quantile Summary in $O((1/\epsilon)\log(1/\epsilon))$ Words. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*. 775–785.
[12] Robert W. Floyd and Ronald L. Rivest. 1975. Expected Time Bounds for Selection. *Commun. ACM* 18, 3 (1975), 165–172.
[13] Greg N. Frederickson. 1983. Tradeoffs for Selection in Distributed Networks (Preliminary Version). In *Proc. 2nd ACM Symposium on Principles of Distributed Computing (PODC)*. 154–160.
[14] A.M. Frieze and G.R. Grimmett. 1985. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics* 10, 1 (1985), 57 – 77.
[15] Michael Greenwald and Sanjeev Khanna. 2001. Space-efficient Online Computation of Quantile Summaries. *SIGMOD Rec.* 30, 2 (2001), 58–66.
[16] Michael B. Greenwald and Sanjeev Khanna. 2004. Power-conserving Computation of Order-statistics over Sensor Networks. In *Proc. of the 23rd ACM Symposium on Principles of Database Systems (PODS)*. 275–285.
[17] Sudipto Guha and Andrew McGregor. 2009. Stream Order and Order Statistics: Quantile Estimation in Random-Order Streams. *SIAM J. Comput.* 38, 5 (2009), 2044–2059.
[18] Bernhard Haeupler. 2016. Analyzing Network Coding (Gossip) Made Easy. *J. ACM* (2016), 26:1–26:22. https://doi.org/10.1145/2629696
[19] C. A. R. Hoare. 1961. Algorithm 63 (PARTITION) and Algorithm 65 (FIND). *Commun. ACM* 4, 7 (1961), 321–322.
[20] Regant Y. S. Hung and Hingfung F. Ting. 2010. An $\Omega((1/\epsilon)\log(1/\epsilon))$ Space Lower Bound for Finding ϵ-approximate Quantiles in a Data Stream. In *Proc. 4th International Conference on Frontiers in Algorithmics (FAW)*. 89–100.
[21] Z. Karnin, K. Lang, and E. Liberty. 2016. Optimal Quantile Approximation in Streams. In *Proc. 57th IEEE Annual Symposium on Foundations of Computer Science*

(FOCS). 71–78.
[22] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. 2000. Randomized rumor spreading. In *Proc. 41st IEEE Symposium on Foundations of Computer Science (FOCS)*. 565–574.
[23] Srinivas Kashyap, Supratim Deb, K. V. M. Naidu, Rajeev Rastogi, and Anand Srinivasan. 2006. Efficient Gossip-based Aggregate Computation. In *Proc. of the 25th ACM Symposium on Principles of Database Systems (PODS)*. 308–317.
[24] David Kempe, Alin Dobra, and Johannes Gehrke. 2003. Gossip-Based Computation of Aggregate Information. In *Proc. 44th IEEE Symposium on Foundations of Computer Science (FOCS 2003)*. 482–491.
[25] Fabian Kuhn, Thomas Locher, and Rogert Wattenhofer. 2007. Tight Bounds for Distributed Selection. In *Proc. of 19th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. 145–153.
[26] Qiang Ma, S. Muthukrishnan, and Mark Sandler. 2013. Frugal Streaming for Estimating Quantiles. In *Space-Efficient Data Structures, Streams, and Algorithms: Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday*. 77–96.
[27] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. 2002. TAG: A Tiny AGgregation Service for Ad-hoc Sensor Networks. *SIGOPS Oper. Syst. Rev.* 36, SI (Dec. 2002), 131–146.
[28] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. 1999. Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Datasets. *SIGMOD Rec.* 28, 2 (1999), 251–262.
[29] J.I. Munro and M.S. Paterson. 1980. Selection and sorting with limited storage. *Theoretical Computer Science* 12, 3 (1980), 315 – 323.
[30] A. Negro, N. Santoro, and J. Urrutia. 1997. Efficient distributed selection with bounded messages. *IEEE Transactions on Parallel and Distributed Systems* 8, 4 (1997), 397–401.
[31] Boaz Patt-Shamir. 2007. A Note on Efficient Aggregate Queries in Sensor Networks. *Theor. Comput. Sci.* 370, 1-3 (Feb. 2007), 254–264.
[32] Boris Pittel. 1987. On Spreading a Rumor. *SIAM J. Appl. Math.* 47, 1 (1987), 213–223.
[33] Michael Rodeh. 1982. Finding the median distributively. *J. Comput. System Sci.* 24, 2 (1982), 162–166.
[34] Doron Rotem, Nicola Santoro, and Jeffrey B. Sidney. 1986. Shout echo selection in distributed files. *Networks* 16, 1 (1986), 77–86. https://doi.org/10.1002/net.3230160108
[35] Nicola Santoro, Michael Scheutzow, and Jeffrey B. Sidney. 1988. On the expected complexity of distributed selection. *J. Parallel and Distrib. Comput.* 5, 2 (1988), 194 – 203.
[36] Nicola Santoro, Jeffrey B. Sidney, and Stuart J. Sidney. 1992. A distributed selection algorithm and its expected communication complexity. *Theoretical Computer Science* 100, 1 (1992), 185 – 204. https://doi.org/10.1016/0304-3975(92)90368-P
[37] N. Santoro and E. Suen. 1989. Reduction techniques for selection in distributed files. *IEEE Trans. Comput.* 38, 6 (1989), 891–896.
[38] Liuba Shrira, Nissim Francez, and Michael Rodeh. 1983. Distributed K-selection: From a Sequential to a Distributed Algorithm. In *Proceedings of 2nd ACM Symposium on Principles of Distributed Computing (PODC)*. 143–153.
[39] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. 2004. Medians and Beyond: New Aggregation Techniques for Sensor Networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*. ACM, New York, NY, USA, 239–249. https://doi.org/10.1145/1031495.1031524
[40] Yong Yao and Johannes Gehrke. 2002. The Cougar Approach to In-network Query Processing in Sensor Networks. *SIGMOD Rec.* 31, 3 (2002), 9–18.

A TOOLS

Lemma A.1. (Chernoff Bound) *Let $X_1, \ldots, X_n$ be indicator variables such that $\Pr(X_i = 1) = p$. Let $X = \sum_{i=1}^{n} X_i$. Then, for $\delta > 0$:*

$$\Pr(X \geq (1+\delta)\mathbb{E}[X]) < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mathbb{E}[X]}$$

$$\Pr(X \leq (1-\delta)\mathbb{E}[X]) < \left[\frac{e^{\delta}}{(1-\delta)^{(1-\delta)}}\right]^{\mathbb{E}[X]}$$

The two bounds above imply that for $0 < \delta < 1$, we have:

$$\Pr(X \geq (1+\delta)\mathbb{E}[X]) < e^{-\delta^2\mathbb{E}[X]/3}$$

$$\Pr(X \leq (1-\delta)\mathbb{E}[X]) < e^{-\delta^2\mathbb{E}[X]/2}.$$

Lemma A.2. *Suppose that for any $\delta > 0$,*

$$\Pr(X > (1+\delta)np) \leq \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{np}$$

then for any $M \geq np$ and $0 < \delta < 1$,

$$\Pr(X > np + \delta M) \leq \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{M} \leq e^{-\delta^2 M/3}$$

Proof. Without loss of generality, assume $M = tnp$ for some $t \geq 1$, we have

$$\begin{aligned}
&\Pr(X > np + \delta M) \\
&\leq \left[\frac{e^{t\delta}}{(1+t\delta)^{(1+t\delta)}}\right]^{np} \\
&= \left[\frac{e^{\delta}}{(1+t\delta)^{(1+t\delta)/t}}\right]^{M} \\
&\leq \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{M} \qquad (*) \\
&\leq e^{-\delta^2 M/3} \qquad \frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} \leq e^{-\delta^2/3} \text{ for } 0 < \delta < 1
\end{aligned}$$

Inequality (*) follows if $(1+t\delta)^{(1+t\delta)/t} \geq (1+\delta)^{(1+\delta)}$, or equivalently, $((1+t\delta)/t)\ln(1+t\delta) \geq (1+\delta)\ln(1+\delta)$. Letting $f(t) = ((1+t\delta)/t)\ln(1+t\delta) - (1+\delta)\ln(1+\delta)$, we have $f'(t) = \frac{1}{t^2}(\delta t - \ln(1+\delta t)) \geq 0$ for $t > 0$. Since $f(1) = 0$ and $f'(t) \geq 0$ for $t > 0$, we must have $f(t) \geq 0$ for $t \geq 1$. □