The Complexity of Data Aggregation in Directed Networks

Fabian Kuhn[1] and Rotem Oshman[2⋆]

[1] University of Lugano, Switzerland
[2] Massachusetts Institute of Technology, USA

Abstract. We study problems of data aggregation, such as approximate counting and computing the minimum input value, in synchronous directed networks with bounded message bandwidth $B = \Omega(\log n)$. In *undirected* networks of diameter D, many such problems can easily be solved in $O(D)$ rounds, using $O(\log n)$-size messages. We show that for directed networks this is not the case: when the bandwidth B is small, several classical data aggregation problems have a time complexity that depends polynomially on the size of the network, even when the diameter of the network is constant. We show that computing an ϵ-approximation to the size n of the network requires $\Omega(\min\left\{n, 1/\epsilon^2\right\}/B)$ rounds, even in networks of diameter 2. We also show that computing a sensitive function (e.g., minimum and maximum) requires $\Omega(\sqrt{n/B})$ rounds in networks of diameter 2, provided that the diameter is *not known in advance* to be $o(\sqrt{n/B})$. Our lower bounds are established by reduction from several well-known problems in communication complexity. On the positive side, we give a nearly optimal $\tilde{O}(D + \sqrt{n/B})$-round algorithm for computing simple sensitive functions using messages of size $B = \Omega(\log N)$, where N is a loose upper bound on the size of the network and D is the diameter.

1 Introduction

Consider a wireless network comprising two base stations, transmitting at high power, and an unknown number of client devices which communicate only with the base stations. The base stations are received at all devices, and each client device is received by at least one base station. However, due to power constraints, the clients are not necessarily received at both stations. The bandwidth of each base station is limited, allowing it to send only a certain number B of bits per timeslot. How many timeslots are required for the base stations to determine the approximate number of clients? We study this problem and other data aggregation problems in *directed networks*, where communication is not necessarily bidirectional.

Data aggregation tasks are central to many distributed systems; for example, a peer-to-peer network might require information about the number of clients that have a local copy of a file, and a sensor network might need to verify that an anomalous reading was detected by a certain percentage of sensors before reporting it. With the increasing

⋆ Rotem Oshman was supported by the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370.

availability of dynamic, large-scale distributed systems, efficient data aggregation has become a particularly interesting challenge.

Classically, data aggregation has been studied in networks with bidirectional communication links. In this setting the method of choice is to first construct a spanning tree of the network graph, and then perform distributed data aggregation "up the tree". In a synchronous undirected network, if computation is initiated by some node, a global broadcast starting at the initiating node induces a breadth-first search spanning tree of the network. Basic aggregation functions, such as the minimum, maximum, sum, or average of values distributed across the nodes of the system, can then efficiently be computed by a simple convergecast on the tree. Even when the message bandwidth is quite restricted (e.g., if only a constant number of data items can be sent in a single message), this method allows any of the functions above to be computed in $O(D)$ rounds in networks of diameter D. Network properties such as the size of the network and the diameter D itself can also be determined in $O(D)$ time using small messages. In fact, in [1] Awerbuch observes that computing certain aggregation functions and computing a spanning tree are intimately related problems, whose time and message complexities are within constant factors of each other. This makes the spanning-tree/convergecast approach a canonical solution of sorts.

The situation changes significantly when communication is not necessarily bidirectional. Constructing a rooted directed spanning tree becomes much more challenging, as it is much harder for the sender of a message to obtain feedback from the recipients, or even to determine who are the recipients. In this paper we show that in contrast to undirected networks, in directed networks with restricted bandwidth it is not always desirable to aggregate data by first computing a rooted spanning tree; for some functions, such as minimum and maximum, it is faster to compute the aggregate by other means. Moreover, we show that the time complexity of computing an aggregate with restricted bandwidth is not governed by the diameter of the network alone; for small-diameter networks, the time complexity of computing certain aggregates is dominated by a factor polynomial in n, the size of the network. We are particularly interested in the effect of *initial knowledge*, i.e., whether or not the problem becomes easier if parameters such as the size or diameter of the network are known in advance.

The paper is organized as follows. In Section 2 we discuss related work. In Section 3 we introduce the model and problems studied in the paper, and review several results in communication complexity that form the basis for our lower bounds. In Section 4 we consider the problems of exact and approximate counting, when the diameter of the network is known to be 2; we show that computing an ϵ-approximate count with constant probability requires $\Omega(\min\left\{n, 1/\epsilon^2\right\}/B)$ rounds where B is the message bandwidth. Our lower bound implies that computing a rooted spanning tree in networks of diameter 2 requires $\Omega(n/B)$ rounds.

In Section 5 we turn our attention to computing sensitive functions in networks of unknown diameter. Informally, a function is *globally sensitive* if its value depends on all the inputs, and ϵ-*sensitive* if its value depends on an ϵ-fraction of inputs. In undirected networks, or even in directed networks of known diameter D, some globally-sensitive functions can be computed in $O(D)$ time with only single-bit messages. We show that for directed networks of *unknown* diameter the picture is quite different: $\Omega(\sqrt{n/B})$ rounds are required, even when the diameter of the network is 2 (but this

fact is not known in advance). This lower bounds holds for randomized computation of any globally-sensitive function and for deterministic computation of any ϵ-sensitive function where $\epsilon \in (0, 1/2)$. The lower bound holds even when the size n of the network is known in advance and the UID space is $1, \ldots, n$.

Finally, in Section 5.2 we give a randomized algorithm for the problem of determining when a node has been causally influenced by all nodes in the graph. This condition is necessary to compute a globally-sensitive function, and sufficient to compute simple functions such as minimum or maximum. The algorithm requires $D + \tilde{O}(\sqrt{n/B})$ rounds w.h.p., nearly matching our lower bound. For lack of space, some of the proofs are omitted here, and appear in the full version of this paper.

2 Background and Related Work

Distributed data aggregation and spanning tree computation. Early work on these problems was concerned with their *message complexity*, that is, the total number of messages sent by all processes, as well as their time complexity. Awerbuch observed in [1] that in undirected networks, the message and time complexity of leader election, computing a distributive sensitive function (e.g., minimum or maximum) and counting are all within a constant factor of the complexity of finding a spanning tree in the network. It is also shown in, e.g., [1, 3] that the time complexity of these problems in undirected networks is $\Theta(n)$ and the message complexity is $\Theta(m + n \log n)$ in networks of size n with m edges. However, the $\Omega(n)$ lower bound is obtained in networks of diameter $\Omega(n)$, and the message complexity lower bound does not yield a non-trivial bound in our model. In a synchronous undirected network of diameter D edges, it is possible to construct a breadth-first search spanning tree in $O(D)$ rounds, even if the diameter and size of the network are not known in advance. Using such a tree, functions such as minimum, maximum, sum, or average can all be computed in time $O(D)$. Based on a pre-computed spanning tree, researchers have also considered the computation of more complicated functions such as the median or the mode [7, 8, 12, 13, 15–17].

Communication complexity. A two-player communication game involves two players, Alice and Bob, which are given private inputs x, y and must compute some joint function of their inputs, $f(x, y)$. In order to compute f the players communicate over several rounds, and are charged for the total number of bits exchanged. The *deterministic communication complexity* of f is the worst-case number of bits exchanged in any deterministic protocol for computing f. The *randomized communication complexity* is defined similarly; in the current paper we are interested in randomized algorithms that err with constant probability.

Communication complexity lower bounds have often been used to obtain lower bounds in distributed computing. The classical reduction technique (see, e.g., [10]) partitions the network into two parts, with each player simulating the nodes on one side of the cut. The input to each player is reflected in the structure of its part of the network or in the input to the network nodes it simulates, and the output or behavior of the distributed algorithm is used , and the communication-complexity lower bound then shows that a certain amount of information must cross the cut. For example, this technique is used in [13] to obtain a lower bound on the complexity of computing the number of distinct elements in the input.

The reductions we give here are quite different in nature. Instead of partitioning the network, the players simulate non-disjoint sets of nodes. Care must be taken to ensure that information about one player's private input does not "leak" to the other player through nodes that both players simulate; this aspect of our reductions strongly relies on the fact that the network is directed.

3 Preliminaries

Network model. We model a synchronous directed network as a strongly connected directed graph $G = (V, E)$, where $E \subseteq V^2$. We use $N^d(v) = \{u \in V \mid \text{dist}(u, v) \leq d\}$ to denote the *d-in-neighborhood* of v, that is, the set of nodes whose distance to v is at most d. Nodes communicate by local broadcast: in each round, every node u sends a single message of size at most B, where $B = \Omega(\log n)$, and this message is delivered to all nodes v such that $(u, v) \in E$. (Each node does not know which nodes receive its message, i.e., it does not know its set of out-neighbors.) We assume that nodes and communication links are reliable and do not fail during an execution.

In the sequel we often refer to algorithms whose correctness is only guaranteed in networks that satisfy some fixed bound on the size or diameter of the network. In this case we say that the bound is *known a priori* (or *known in advance*). Our lower bounds assume that each node has a unique identifier (UID) drawn from some UID space $1, \ldots, N$, where N is an upper bound on the size of the network that is known in advance. For convenience, we assume the existence of two distinguished UIDs $a, b \notin [N]$; our reductions "embed" the two players in the graph as nodes a and b respectively. Some of our lower bounds allow for the case where $N = n$, i.e., the exact size of the network is known to all nodes and the UID space is $1, \ldots, n$. In contrast, the algorithm in Section 5.2 requires only a loose upper bound $N \geq n$ and does not use UIDs at all.

Problem statements. We are interested in the following distributed problems.

- *ϵ-approximate counting*: nodes are initially provided with some loose upper bound N on the size n of the network, and each node v must eventually output an approximate count $\tilde{n}_v$ satisfying $|\tilde{n}_v - n| \leq \epsilon \cdot n$.
- Computing *globally-sensitive* functions of the input: a function is said to be *globally sensitive* if there exists an input assignment $\overline{x}$ such that changing any single coordinate of $\overline{x}$ yields a different function value. For example, the all-one input assignment witnesses the global sensitivity of computing a minimum.
- Computing *ϵ-sensitive* functions of the input: a function is *ϵ-sensitive* if there is an input assignment $\overline{x}$ such that changing any $\lceil \epsilon n \rceil$ coordinates of $\overline{x}$ yields a different function value. For example, the function that returns 1 iff at least 25% of the inputs are 1 is $(1/4)$-sensitive, as witnessed by the all-zero input assignment.

Communication complexity lower bounds. Our results rely on several celebrated lower bounds in communication complexity. Perhaps the best known lower bound concerns the Set Disjointness problem, DISJ_n, in which the players are given sets $X, Y \subseteq [n]$ (respectively) and must determine whether $X \cap Y = \emptyset$.

Theorem 1 ([5, 14]). *The randomized communication complexity of* DISJ_n *is* $\Omega(n)$.

We are also interested in a relaxed variant called Gap Set Disjointness, GAP-DISJ$_{n,g}$: here the players are given sets $X, Y \subseteq [n]$, with the *promise* that either $X \cap Y = \emptyset$ or $|X \cap Y| \geq g$. The players must determine which of these cases holds. When the gap g is large with respect to n, GAP-DISJ$_{n,g}$ is quite easy for randomized algorithms (one can use random sampling to find an element of the intersection if it is large). However, for deterministic protocols the problem remains hard even with a linear gap. (This fact appears to be folklore in the communication complexity community; we include a proof in the full version of this paper.)

Theorem 2. *For any constant $\epsilon \in (0, 1/2)$, the deterministic communication complexity of* GAP-DISJ$_{n,(1/2-\epsilon)n}$ *is $\Omega(n)$.*

The final problem is GAP-HAMMING-DISTANCE, denoted GHD$_{n,g}$, where the players receive vectors $x, y \in \{0,1\}^n$ and must determine whether the Hamming distance $\Delta(x, y)$ satisfies $\Delta(x, y) > n/2 + g$ or whether $\Delta(x, y) \leq n/2 - g$. (If neither holds, any answer is allowed.) Characterizing the randomized communication complexity of GHD remained an open problem for a long time after its introduction in [4] (for the case $g = \sqrt{n}$, which is in some sense the most interesting setting), until in [2], Chakrabarti and Regev proved the following lower bound.

Theorem 3 ([2]). *For any $g \leq n$, the randomized communication complexity of* GHD$_{n,g}$ *is $\Omega(\min\{n, n^2/g^2\})$.*

The reductions in this paper are *public-coin protocols*: they assume that Alice and Bob have access to a shared random string (of unbounded length). The lower bounds above are stated for *private-coin protocols*, where each player has its own private randomness. However, any public-coin protocol can be transformed into a private-coin protocol at the cost of $O(\log n)$ additional bits [10], so the distinction is mostly immaterial for our purposes.

4 Approximate and Exact Counting

We begin by describing a lower bound for ϵ-approximate counting or exact counting. In this setting we assume that nodes know some loose upper bound $N \geq n$ on the size of the network, and must determine the exact or approximate size. Since exact counting is a special case of approximate counting, we describe the lower bound for approximate counting, and later discuss exact counting.

The lower bound is obtained by reduction from GHD$_{N,\epsilon N}$. Suppose we are given an ϵ-approximate counting algorithm $\mathcal{A}$. Given an instance (x, y) of GHD$_{N,\epsilon N}$, we construct a network $G_{x,y}$, in which Alice and Bob jointly simulate the execution of $\mathcal{A}$. When $\mathcal{A}$ terminates, Alice and Bob use the output of $\mathcal{A}$ to determine the correct answer to GHD on the instance (x, y). Since Alice knows only her input x and Bob knows only y, neither player knows the complete topology of the network $G_{x,y}$, which depends on both x and y. The players therefore cooperate to simulate the execution of $\mathcal{A}$ in $G_{x,y}$.

Let $X, Y \subseteq [N]$ be the sets whose characteristic vectors are x and y, respectively. The network $G_{x,y}$ is given by $G_{x,y} = (V_{x,y}, E_{x,y})$, where $V_{x,y} = X \cup Y \cup \{a, b\}$ (for $a, b \notin [N]$), and $E_{x,y} = (\{a\} \times V_{x,y}) \cup (\{b\} \times V_{x,y}) \cup (X \times \{a\}) \cup (Y \times \{b\})$ (see Fig. 1).

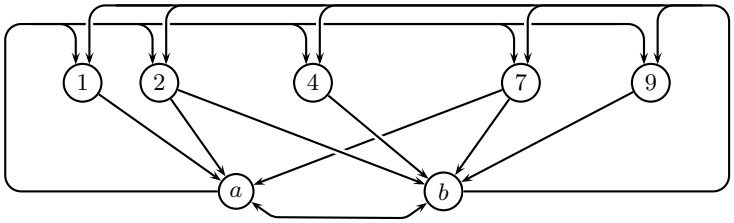

Fig. 1. The network $G_{x,y}$ for $x = 110000100, y = 010100101$ (i.e., $X = \{1, 2, 7\}, Y = \{2, 4, 7, 9\}$)

The Hamming distance $\Delta(x, y)$ is closely related to the size of $G_{x,y}$:

Lemma 1. *For all $(x, y) \in (\{0, 1\}^N)^2$, the graph $G_{x,y}$ is strongly connected, its diameter is 2, and its size is $|V_{x,y}| = (\|x\|_1 + \|y\|_1 + \Delta(x, y))/2 + 2$.*

Next we show that an efficient algorithm for approximating the size of diameter 2 networks leads to an efficient protocol for $\mathrm{GHD}_{N,\epsilon N}$.

Lemma 2. *Given an ϵ-approximate counting algorithm $\mathcal{A}$ which outputs a correct answer after t rounds with probability at least $1 - \delta$, one can construct a public-coin protocol for $\mathrm{GHD}_{N,\epsilon N}$ which exchanges a total of $O(Bt + \log N)$ bits and succeeds with probability $1 - \delta$.*

Proof. Given an instance (x, y), Alice and Bob simulate the execution of $\mathcal{A}$ in $G_{x,y}$ as follows. Alice locally simulates the nodes in $X \cup \{a\}$, and Bob locally simulates the nodes in $Y \cup \{b\}$. The shared random string is used to provide the randomness of all nodes in the network. (Since Alice and Bob do not initially know which of the nodes $\{1, \ldots, N\}$ are present, we interpret the shared random string as containing the randomness of each node $1, \ldots, N$ regardless of whether or not the node is in $X \cup Y$.) Notice that there can be some overlap, $X \cap Y$, which is simulated by both players independently.

The initial states of all nodes in $X \cup \{a\}$ and in $Y \cup \{b\}$ are known to Alice and Bob, respectively, because they depend only on the UIDs of these nodes and on the shared randomness. Each round of $\mathcal{A}$ is simulated as follows:

- Based on the states of their local simulations, Alice and Bob compute the messages sent by the nodes in $X \cup \{a\}$ and in $Y \cup \{b\}$, respectively.
- Alice sends to Bob the message sent by node a, and Bob sends to Alice the message sent by b. Following this exchange, Alice and Bob have all the messages received by each node they need to simulate.
- The players update the states of their local simulations by feeding to each node the messages it receives in $G_{x,y}$: the nodes of $X \cup Y$ receive the messages sent by a and b; node a receives the messages sent by nodes in $X \cup \{b\}$; and node b receives the messages sent by nodes in $Y \cup \{a\}$. (Note that Alice knows X and Bob knows Y, so the two players know which messages are supposed to be received by nodes a, b, respectively.)

Although Alice and Bob do not directly exchange information about the states of nodes in $X \cap Y$ — indeed, they do not *know* which nodes are in $X \cap Y$, and this is what makes the problem difficult — still their local simulations agree on the states of these nodes.

With probability at least $1-\delta$, after t rounds of the simulation node a halts and outputs an approximate count $\tilde{n}$ which satisfies $|\tilde{n}-n| \leq \epsilon n$. When node a halts, Alice sends $\tilde{n}$ to Bob, and in addition Alice and Bob send each other $|X| = \|x\|_1$ and $|Y| = \|y\|_1$ (respectively). Let $\tilde{\Delta} = 2(\tilde{n}-2) - \|x\|_1 - \|y\|_1$. Both players output 0 if $\tilde{\Delta} < N/2$, and 1 if $\tilde{\Delta} \geq N/2$. (If node a fails to halt after t rounds, the players output an arbitrary answer.)

If $|\tilde{n}-n| \leq \epsilon n$ then Lemma 1 shows that $|\tilde{\Delta} - \Delta(x,y)| = 2|\tilde{n}-n| \leq 2\epsilon n \leq 2\epsilon N$. Hence, with probability at least $1-\delta$, the players output the correct answer: if $\Delta(x,y) \geq N/2 + 2\epsilon N$ then $\tilde{\Delta} \geq N/2$, and if $\Delta(x,y) < N/2 - 2\epsilon N$ then $\tilde{\Delta} < N/2$.

The total number of bits sent during the protocol is $2Bt + 2\log(N)$. In addition, to transform the protocol into a private-coin protocol we require $O(\log N)$ additional bits. The communication complexity is therefore $O(Bt + \log N)$. □

Although our reduction is stated in terms of the upper bound N (we reduce from $\mathrm{GHD}_{N,\epsilon N}$), the "hard" instances are the ones where n is roughly linear in N; it is always possible to solve GHD by exchanging the coordinates of indices i such that $x_i = 1$ or $y_i = 1$, and hence when $|X \cup Y| = n$ the problem can easily be solved in $O(n \log N)$ bits. It is therefore more informative to state our lower bound in terms of the actual size n of the network. From Theorem 3 and the reduction above, we obtain the following lower bound.

Theorem 4. *If $B = \Omega(\log N)$, a randomized algorithm for computing an ϵ-approximate count requires $\Omega((\min\{n, 1/\epsilon^2\}/B)$ rounds to succeed with probability 2/3 in networks of diameter 2.*

Remarks. The *deterministic* communication complexity of $\mathrm{GHD}_{N,g}$ is $\Omega(N)$ even when $g = c \cdot N$ for a sufficiently small constant c [2]; therefore deterministically computing an ϵ-approximate count for ϵ a sufficiently small constant requires $\Omega(n/B)$ rounds. As for *exact* counting (deterministic or randomized), computing the exact count is as hard as computing a $(1/n)$-approximate count, so $\Omega(n/B)$ rounds are required.

The lower bound of Theorem 4 is nearly tight if the diameter of the network is known. An algorithm for ϵ-approximate counting is given in [11]; the algorithm of [11] sends messages containing real numbers, but using a rounding scheme to bound the size of messages (see [9]), one obtains an $\tilde{O}(D + \min\{n, 1/\epsilon^2\}/B)$-round algorithm for networks of known diameter D. For the case where the diameter is unknown, we obtain a stronger lower bound in the next section.

Finally, the reduction from Lemma 2 also shows that finding a rooted spanning tree in directed networks is hard even when the diameter of the network is known *a priori* to be 2. In the network $G_{x,y}$, the nodes of $X \cup Y$ are not connected to each other; therefore any rooted spanning tree of $G_{x,y}$ has diameter at most 3, as each node of $X \cup Y$ except possibly the root must have either a or b as its parent in the tree. If one can find a rooted spanning tree of $G_{x,y}$ in t rounds, then an exact count can be computed in $t+3$ rounds by finding such a tree and then "summing up the tree" (convergecast). Since exact counting requires $\Omega(n/B)$ rounds, so does computing a rooted spanning tree. In the full version of this paper we show that this lower bound continues to hold when the size of the network is known *a priori*, provided that the UID space is of size at least $(1+\epsilon)n$ for some arbitrarily small constant ϵ.

5 Computing Sensitive Functions

In this section we study the complexity of computing sensitive functions, such as the minimum or maximum input value. In contrast to the previous section, here we are interested in instances where the diameter of the network is not known *a priori* to be small, but the algorithm is deployed in a network that *does* in practice have a small diameter. We will show that in such cases it is not possible to exploit the small diameter of the network; the worst-case running time of the algorithm must be $\Omega(D + \sqrt{n/B})$. We also give a nearly-matching algorithm for computing simple sensitive functions.

Let f be a globally-sensitive function, and let $\bar{x}$ be an input assignment under which changing any node's input changes the value of f (i.e., for all $\bar{y} \neq \bar{x}$ we have $f(\bar{x}) \neq f(\bar{y})$). In any execution where the input is $\bar{x}$, at time t, a node v can only know the value of f if $N^t(v) = V$, that is, if t rounds are sufficient for a message from any node in the network to reach node v; otherwise there is some node whose input node v cannot know at time t, and this node's input may determine the value of f. Similarly, if f is ϵ-sensitive, there exists an input assignment under which no node can know the value of f at time t unless $|N^t(v)| > (1-\epsilon)n$. This motivates us to study the following problem:

Definition 1 (Hearing from m nodes). *In the* Hear-from-m-nodes *problem, denoted* HF_m*, each node v in the network must halt at some time t such that $|N^t(v)| \geq m$.*

The worst-case time complexity of computing a globally-sensitive function is at least the worst-case time complexity of solving HF_n, and similarly for ϵ-sensitive functions and $\mathrm{HF}_{(1-\epsilon)n}$. (In fact, computing an ϵ-sensitive function can require hearing from *strictly more* than $(1-\epsilon)n$ nodes.) Of course, HF_n can easily be solved by having all nodes wait until time $n-1$; however, we are interested here in efficient solutions, which terminate faster in networks with smaller diameter (recall, however, that the diameter is not known in advance).

5.1 Lower Bounds on Computing a Sensitive Function

In this section we show that even when the diameter of the network is 2, *learning* that the diameter is 2 requires $\Omega(\sqrt{n/B})$ rounds in the worst case. More formally, we show that when the size of the network is known, the UID space is $1, \ldots, n$, and no *a priori* bound on the diameter is known,

(a) Any randomized algorithm for HF_n requires $\Omega(\sqrt{n/B})$ rounds to succeed with constant probability, even when executed in a network of diameter 2; and
(b) For any $\epsilon \in (0, 1/2)$, any deterministic algorithm for $\mathrm{HF}_{(1-\epsilon)n}$ requires $\Omega(\sqrt{n/B})$ rounds, again when executed in networks of diameter 2.

(Of course, in networks of diameter 2 we have $|N^2(v)| = n$ for all nodes v, so $t = 2$ is sufficient; however, this fact is not known to the algorithm in advance.)

Fix an algorithm $\mathcal{A}$ for HF_m and a network size $n \geq m$. We describe a reduction from Set Disjointness or Gap Set Disjointness, which we will use to show both the hardness of HF_n for randomized algorithms and the hardness of $\mathrm{HF}_{(1-\epsilon)n}$ for deterministic algorithms.

As in Section 4, in the reduction we construct a network G based on the instance of Set Disjointness given to Alice and Bob. The two players then simulate the execution of $\mathcal{A}$ in G, and output an answer to Set Disjointness (or Gap Set Disjointness) based on the behavior of $\mathcal{A}$ in G — in this case, based on the time when $\mathcal{A}$ terminates. We now describe the construction of the network and the simulation used by Alice and Bob.

The construction has several parameters. First, let $t_{\mathcal{A}}$ be the number of rounds such that when $\mathcal{A}$ is executed in a network of size n with node UIDs $1, \ldots, n, a, b$ (as before we add UIDs a, b for convenience), with probability at least $2/3$ all nodes halt by time $t_{\mathcal{A}}$. Based on $t_{\mathcal{A}}$ and on m, we choose a *segment length* $s \geq t_{\mathcal{A}} + 1$ which will be fixed later. Informally, in the reduction nodes must distinguish diameter 2 networks from diameter $s + 2$, and we will show that this requires $\Omega(n/s)$ rounds in the worst-case.

Assume for simplicity that s divides n. We divide the nodes $1, \ldots, n$ into *segments* $S_1, \ldots, S_{n/s}$, each of size s, where $S_i := \{(i-1) \cdot s + 1, (i-1) \cdot s + 2, \ldots, i \cdot s\}$. Each segment S_i is further subdivided into two parts: a *back end* S_i^B containing nodes $(i-1) \cdot \cdots + 1, \ldots, i \cdot s - t_{\mathcal{A}}$, and a *front-end* S_i^F containing the remaining nodes, $i \cdot s - t_{\mathcal{A}} + 1, \ldots, i \cdot s$. In the sequel we implicitly use wrap-around (i.e., $\bmod n$ arithmetic) for node indices, so that $-1 \equiv n$, $-2 \equiv n - 1$, and so on.

We are now ready to describe the reduction itself. The reduction is from $\text{DISJ}_{n/s}$, that is, Set Disjointness (or Gap Set Disjointness) with a universe of n/s elements; each segment S_i represents a single element of the universe. Given an instance (x, y) of $\text{DISJ}_{n/s}$, we define a network $G_{s,x,y} := (\{1, \ldots, n, a, b\}, E_{s,x,y})$ (see Fig. 2), where

- Nodes a, b have edges to all nodes of the graph.
- Nodes $1, \ldots, n$ are connected in a directed cycle: for each $i \in [n]$ we have $(i, i+1) \in E_{s,x,y}$.
- In each segment S_i, the last node (node $i \cdot s$) is connected to node a. (This is to ensure strong connectivity and a bound of $s + 2$ on the diameter.)
- For all $i \notin X$ and for all $v \in S_i$ we have $(v, a) \in E_{s,x,y}$; similarly, for all $i \notin Y$ and for all $v \in S_i$ we have $(v, b) \in E_{s,x,y}$.

Here, X and Y are the sets whose characteristic vectors are x, y respectively.

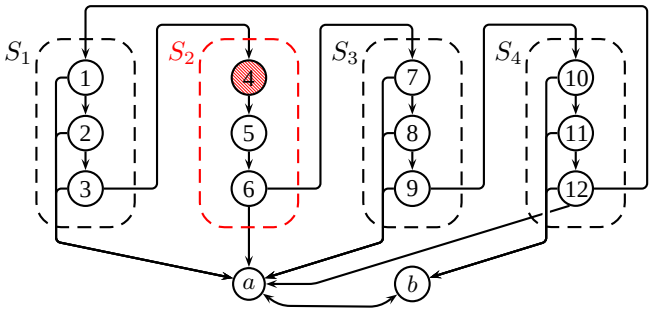

Fig. 2. The network $G_{s,x,y}$ from Thm. 5, with $n = 12, t_{\mathcal{A}} = 2, s = t_{\mathcal{A}} + 1 = 3$. Edges from a, b to nodes $1, \ldots, 12$ are omitted for clarity. The DISJ_4 instance shown here is $X = \{2, 4\}, Y = \{1, 2, 3\}$. Since $2 \in X \cap Y$, all S_2 nodes except the last (node 6) are not connected to a or to b. Therefore $4 \notin N^{t_{\mathcal{A}}}(a)$, i.e., two rounds are not sufficient for node a to hear from node 4.

With the exception of the last node in each segment (which is always connected to node a), the nodes in segment S_i are connected to node a iff Alice did *not* receive i in her input, and connected to node b iff Bob did not receive i in his input. Therefore, if there exists an element i in the intersection $X \cap Y = \overline{\overline{X} \cup \overline{Y}}$, the nodes of the corresponding segment S_i, with the exception of the last node, will not be connected to either node a or node b. These nodes are only connected to the rest of the graph by the cycle edges $(i-1) \cdot s + 1 \to (i-1) \cdot s + 2 \to \ldots \to i \cdot s$. Consequently the diameter of the graph is $s + 2 > t_{\mathcal{A}}$ in this case. In $t_{\mathcal{A}}$ rounds, nodes a and b can only hear from the last $t_{\mathcal{A}}$ nodes of segment S_i, i.e., only from the front-end S_i^F; for each segment S_i such that $i \in X \cap Y$, $|S_i^B| = s - t_{\mathcal{A}}$ nodes are missing from $N^{t_{\mathcal{A}}}(a)$.

On the other hand, if $X \cap Y = \emptyset$ (or equivalently, $\overline{X} \cup \overline{Y} = \{1, \ldots, n/s\}$), all nodes in all segments are connected to either node a or node b, and the diameter of the graph is 2.

Lemma 3. *For any $x, y \in \{0,1\}^n$,*
(a) The graph $G_{s,x,y}$ is strongly connected,
(b) For all $i \in X \cap Y$ and for all $v \in S_i^B$ we have $v \notin N^{t_{\mathcal{A}}}(a)$ and $v \notin N^{t_{\mathcal{A}}}(b)$,
(c) If $X \cap Y = \emptyset$, the diameter of $G_{s,x,y}$ is 2, and
(d) $|N^{t_{\mathcal{A}}}(a)| \leq n - |X \cap Y| \cdot (s - t_{\mathcal{A}})$ (and similarly for b).

Alice and Bob simulate the execution of $\mathcal{A}$ in $G_{s,x,y}$ in a slightly different manner than in Lemma 2; here both players simulate nodes $1, \ldots, n$ regardless of the input instance, and in addition Alice simulates node a and Bob simulates node b. The remainder of the simulation is the same as in Lemma 2, and we omit the details here.

Proposition 1. *Given inputs x and y respectively, and a shared string representing the randomness of all nodes, Alice and Bob can each simulate nodes $\{a, 1, \ldots, n\}$ and $\{b, 1, \ldots, n\}$ (respectively) throughout rounds $1, \ldots, t_{\mathcal{A}}$ of the execution of $\mathcal{A}$ in $G_{s,x,y}$.*

It remains only to put the pieces together to obtain the following lower bounds.

Theorem 5. *If the diameter of the network is not known initially, any randomized algorithm for computing a globally-sensitive function requires $\Omega(\sqrt{n/B})$ rounds with probability at least $2/3$ when executed in networks of diameter 2.*

Proof. As explained above, it is sufficient to show the corresponding bound for HF_n.

Fix an algorithm $\mathcal{A}$, and let $t_{\mathcal{A}}$ be defined as above. Fix a segment length of $s := t_{\mathcal{A}} + 1$ (so that the back-end of each segment contains exactly one node).

Given an instance (x, y) of $\textsc{Disj}_{n/s}$, Alice and Bob jointly simulate the first $t_{\mathcal{A}}$ rounds in the execution of $\mathcal{A}$ in $G_{s,x,y}$ as in Proposition 1. After $t_{\mathcal{A}}$ rounds, Alice informs Bob whether or not node a has halted in the simulation. If node a has halted, the players output "$X \cap Y = \emptyset$"; otherwise they output "$X \cap Y \neq \emptyset$".

As we saw in Lemma 3, if $X \cap Y = \emptyset$ then the diameter of $G_{t_{\mathcal{A}}+1,x,y}$ is 2, so with probability at least $2/3$ all nodes halt after $t_{\mathcal{A}}$ rounds and Alice and Bob output "$X \cap Y = \emptyset$". On the other hand, if $X \cap Y \neq \emptyset$, then by time $t_{\mathcal{A}}$ node a has not heard from all nodes, as Lemma 3 shows that at least $(s - t_{\mathcal{A}}) \cdot |X \cap Y| = |X \cap Y| > 0$ nodes are missing from $N^{t_{\mathcal{A}}}(a)$. Consequently, with probability at least $2/3$, node a does not halt by time $t_{\mathcal{A}}$ and the players output "$X \cap Y \neq \emptyset$".

The total number of bits exchanged by the players in the protocol above is $2B \cdot t_{\mathcal{A}} + 1$, because Alice and Bob only send each other the messages output by nodes a and b, plus one bit needed for Alice to inform Bob whether node a has halted. An additional $O(\log(n/t_{\mathcal{A}}))$ bits are required to obtain a private-coin protocol. Since the randomized communication complexity of $\text{DISJ}_{\lfloor n/(t_{\mathcal{A}}+1)\rfloor}$ is $\Omega(n/t_{\mathcal{A}})$, we must have $2B \cdot t_{\mathcal{A}} + 1 = \Omega(n/t_{\mathcal{A}})$, or in other words, $t_{\mathcal{A}} = \Omega(\sqrt{n/B})$. □

Theorem 6. *If the diameter of the network is initially unknown, any deterministic algorithm for computing an ϵ-sensitive function, where $\epsilon \in (0, 1/2)$ is constant, requires $\Omega(\sqrt{n/B})$ rounds when executed in networks of diameter 2.*

Proof (sketch). We prove that $\Omega(\sqrt{n/B})$ rounds are required to solve $\text{HF}_{(1-\epsilon)n}$ deterministically for any $\epsilon \in (0, 1/2)$, even in networks of diameter 2. The proof is similar to that of Thm. 5, except that we now reduce from $\text{GAP-DISJ}_{\lfloor n/s\rfloor,\epsilon'\lfloor n/s\rfloor}$ for an appropriately chosen constant $\epsilon' \in (0, 1/2)$, and the segment length s is also chosen differently.

Fix a deterministic algorithm $\mathcal{A}$ for $\text{HF}_{(1-\epsilon)n}$, and let $t_{\mathcal{A}}$ be the maximal time at which the algorithm halts in any network of diameter 2. We must now choose a segment length $s = \Theta(t_{\mathcal{A}})$ so that the following conditions hold:

(a) If $X \cap Y = \emptyset$, then the diameter of $G_{s,x,y}$ is 2. This ensures that in "yes" instances, all nodes halt by time $t_{\mathcal{A}}$.

(b) If $|X \cap Y| \geq \epsilon'\lfloor n/s\rfloor$ then we have $|N^{t_{\mathcal{A}}}(a)| < (1-\epsilon)(n+2)$. This ensures that in "no" instances, node a *cannot* halt by time $t_{\mathcal{A}}$.

These conditions suffice for the protocol from Thm. 5 to solve $\text{GAP-DISJ}_{\lfloor n/s\rfloor,\epsilon'\lfloor n/s\rfloor}$ as well. From Lemma 3 we see that condition (a) holds regardless of our choice of s. As for condition (b), from part (d) of Lemma 3, it is sufficient to choose $s := \alpha t_{\mathcal{A}}, \epsilon'$ so that

$$n - \epsilon' \left\lfloor \frac{n}{\alpha t_{\mathcal{A}}} \right\rfloor \cdot (\alpha - 1)t_{\mathcal{A}} < (1-\epsilon)(n+2).$$

There exist constants $\alpha > 1, \epsilon' \in (0, 1/2)$ satisfying this constraint (we omit the details for lack of space). For this choice of s, ϵ', the reduction from Thm. 5 yields a protocol with communication complexity $2Bt_{\mathcal{A}} + 1$ for $\text{GAP-DISJ}_{n',\epsilon'n'}$, where $n' = \lfloor n/s\rfloor = O(n/t_{\mathcal{A}})$. Because GAP-DISJ is linearly hard for deterministic protocols even when the gap is linear in the universe size (Theorem 2), we must have $2Bt_{\mathcal{A}} + 1 = \Omega(n/t_{\mathcal{A}})$, i.e., $t_{\mathcal{A}} = \Omega(\sqrt{n/B})$. □

Remarks. The construction in this section can be modified to show a few related results.

In Theorems 5 and 6 we assumed that no upper bound on the diameter of the network is known in advance. Suppose now that some upper bound $\bar{D}$ on the diameter is known in advance. We can show that any randomized algorithm for computing a globally-sensitive function, and any deterministic algorithm for computing an ϵ-sensitive function for $\epsilon \in (0, 1/2)$, requires $\Omega(\min\left\{\bar{D}, \sqrt{n/B}\right\})$ rounds when executed in networks of diameter 2.

To see this, observe that the diameter of $G_{s,x,y}$ never exceeds $s + 2$. Suppose that $\bar{D} = o(\sqrt{n/B})$ and we are given an HF_n-algorithm (or similarly, a deterministic $\text{HF}_{(1-\epsilon)n}$-algorithm) $\mathcal{A}$ with $t_{\mathcal{A}} < \bar{D} - 2$. If we use a segment length of $s = t_{\mathcal{A}} + 1 \leq \bar{D} - 2$, as in Thm. 5, the diameter upper bound is not violated in $G_{s,x,y}$. For this choice of

s, the reduction from Thm. 5 allows us to solve $\text{DISJ}_{\lfloor n/s \rfloor}$, where $\lfloor n/s \rfloor \geq \lfloor n/(\bar{D}-2) \rfloor$, using less than $2(\bar{D}-2)B+1$ bits. We must have $2(\bar{D}-2)B+1 = \Omega(n/\bar{D})$, that is, $\bar{D} = \Omega(\sqrt{n/B})$, contradicting our assumption that $\bar{D} = o(\sqrt{n/B})$.

Next, consider the problem of finding an approximate count when the diameter is not known in advance. (Our lower bound from Section 4 allows the diameter to be known in advance, but the following requires it to be unknown.) Let N be the best upper bound known in advance on the count. We will show that in order to distinguish a network of size n from a network of size N, nodes a, b must solve a Set Disjointness instance of size $O(n/t_{\mathcal{A}})$, so that again $t_{\mathcal{A}} = \Omega(\sqrt{n/B})$ rounds are required.

Recall that in $G_{s,x,y}$, the distance from any node $(i-1)\cdot s+1$ where $i \in X \cap Y$ to nodes a and b is $s > t_{\mathcal{A}}$. Thus, when $X \cap Y \neq \emptyset$, we can choose a node $v := (i-1)\cdot s+1$ where $i \in X \cap Y$, and "hide" nodes $n+1, \ldots, N$ behind it, adding edges from nodes $n+1, \ldots, N$ to v and from nodes a, b to nodes $n+1, \ldots, N$. Let $G'_{s,x,y}$ be the resulting network. Since the distance from node v to nodes a, b exceeds $t_{\mathcal{A}}$, and the new nodes $n+1, \ldots, N$ are connected only to node v, $t_{\mathcal{A}}$ rounds are insufficient for nodes a, b to distinguish $G_{s,x,y}$ from $G'_{s,x,y}$. Therefore, if $X \cap Y \neq \emptyset$, an algorithm for distinguishing networks of size $n+2$ from networks of size N cannot terminate by time $t_{\mathcal{A}}$ in $G_{s,x,y}$ (except with small probability). This is sufficient to carry out the reduction from Thm. 5 exactly as before, obtaining an $\Omega(\sqrt{n/B})$ lower bound on any non-trivial approximation of the count.

5.2 A $(D + \tilde{O}(\sqrt{n/B}))$-Round Algorithm for HF_n

We now give an algorithm that solves HF_n in nearly-optimal time. If, for example, the minimum input value heard so far is forwarded alongside the messages of our algorithm, this allows nodes to compute the global minimum. The algorithm does not use UIDs, and it only requires an polynomially loose upper bound $N \geq n$ on the count.

High-level overview of the algorithm. Initially, each node computes a sequence of independent Bernoulli variables, and stores the indices of the variables that turned up one. These indices are called *tokens*. The tokens are then forwarded throughout the network by all nodes. If a node does not receive any new tokens for a sufficiently long period of time, it concludes that it has heard from all nodes, and halts. The waiting period is long enough so that if at the end t of the period we do not have $N^t(v) = V$, then during the waiting period the tokens of many new nodes are received by v, and the probability that none of these nodes generated a token that was not previously known is very small.

Detailed description. Since nodes do not know the exact size n, we use exponentially-increasing guesses 2^k for $k = \lceil \log\log N \rceil, \ldots, \lceil \log N \rceil$. We refer to each value of k as a *level*. On level k, each node computes ℓ_k independent Bernoulli variables X_k^i with $\Pr[X_i = 1] = 1/2^{k+2}$, where $\ell_k = \tilde{\Theta}(\sqrt{2^k B})$ (the exact value will be fixed later). We denote by $L_k := \sum_{i=1}^{k} \ell_i$ the total number of variables computed on levels $k' \leq k$.

At the beginning of the algorithm, the indices of the variables that turned up one on each level are collected in a set $Tokens = \{(k,i) \mid X_k^i = 1\}$. The tokens are ordered lexicographically — first by level and then by index. Each token can be represented using $\log n + \log\log N$ bits; for simplicity we assume that each message can fit β

for $k = \lceil \log\log N \rceil, \lceil \log\log N \rceil + 1, \ldots, \lceil \log N \rceil$ **do**
  Compute independent $X_k^1, \ldots, X_k^{\ell_k} \sim \text{Bernoulli}(2^{-(k+2)})$
  $last\_update_k \leftarrow 0$
$Tokens \leftarrow \{(k,i) \in \mathbb{N}^2 \mid X_k^i = 1\}$, $Sent \leftarrow \emptyset$
for $r = 1, 2, \ldots$ **do**
  $X \leftarrow$ select the β smallest tokens in $Tokens \setminus Sent$
  broadcast X and set $Sent \leftarrow Sent \cup X$
  receive tokens Y from neighbors
  for all $y = (k,i) \in Y \setminus Tokens$ **do** $\forall k' \geq k : last\_update_{k'} \leftarrow r$
  $Tokens \leftarrow Tokens \cup Y$
  if $\exists k : (|\{(k,i) \in Tokens\}| \leq 2\ell_k/3) \wedge (r - last\_update_k \geq 2\tau_k)$ **then halt**

Algorithm 1: A $(D + \tilde{O}(\sqrt{n/B}))$-round algorithm for HF_n

tokens, that is, $B = \beta(\log n + \log\log N)$ where β is an integer. Pseudocode for the algorithm is given by Algorithm 1. In the sequel, let $\tau_k := \lceil L_k/\beta \rceil$.

After generating an initial set of tokens, the tokens are disseminated in batches of β tokens each, with lower-level tokens taking precedence over higher-level tokens. Each node halts as soon as on some level k, fewer than $2\ell_k/3$ tokens have been received in total, and in the past $2\tau_k = 2\lceil L_k/\beta \rceil$ rounds no new token was received.

The algorithm relies on *pipelining* [18] to quickly disseminate small tokens throughout the network. Because we forward small tokens before large ones, the progress of a token (k,i) can only be impeded by tokens on its own level (k) or lower levels ($k' < k$); there are at most $L_k = \sum_{i=1}^{k} \ell_i$ such tokens, and β of them can be sent per message. Thus the "latency" of token (k,i) is at most $\lceil L_k/\beta \rceil = \tau_k$. More formally, for a set $S \subseteq V$ of nodes, let $A_k(S) := \bigcup_{v \in S} \{(k,i) \mid (k,i) \in Tokens_v(0)\}$ be the level-k tokens generated by the nodes of S. Let $Tokens_v(t)$ stand for the value of the local variable $Tokens$ at node v and time t. The latency of level-k tokens is bounded by the following lemma.

Lemma 4. *For all $v \in V$ and $t \geq \tau_k$, $A_k(N^{t-\tau_k}(v)) \subseteq Tokens_v(t) \subseteq A(N^t(v))$.*

We can now bound the round complexity of the algorithm in terms of the "correct" value of k, which is roughly $\log(n)$.

Lemma 5. *Let $\hat{k} := \min\{\lceil \log\log N \rceil, \lceil \log n \rceil\}$. In graphs of diameter D, the algorithm terminates in $D + 3\lceil L_{\hat{k}}/\beta \rceil$ rounds with probability at least $1 - e^{-\ell_{\hat{k}}/9}$.*

Proof (sketch). It is not difficult to show that the expected number of level-k tokens generated by all the nodes together is at most $\ell_{\hat{k}}/3$. A Chernoff bound shows that w.h.p., the total number of level-k tokens does not exceed $(2/3)\ell_{\hat{k}}$, so the second part of the termination condition is satisfied for $k = \hat{k}$. For the first part of the condition we rely on pipelining: Lemma 4 shows that $A_{\hat{k}}(N^D(v)) \subseteq Tokens_v(D + \tau_k)$ for all nodes v; since $N^D(v) = V$, at time $D + \tau_k$, each node v has already received all tokens generated anywhere in the network. After this time no node can receive any new tokens, so all nodes halt no later than time $D + 3\tau_k$. □

Next we show that w.h.p., nodes do not halt before they have heard from all n nodes.

Lemma 6. *If the level-k termination condition holds at node v at time t, then with probability at least $1 - e^{-\ell_k^2/(3\cdot 2^{k+3}\beta)}$ we have $N^t(v) = V$.*

Proof (sketch). The level-k termination condition asserts that no new level-k tokens are received during the time interval $[t - 2\tau_k, t]$. Assume that $N^t(v) \neq V$, and set $S := N^{t-2\tau_k}(v), S' := N^{t-\tau_k}(v)$. From Lemma 4 we see that

(a) $A_k(S') \subseteq Tokens_v(t)$, that is, all tokens generated by the nodes of S' are known to v at time t; and

(b) $Tokens_v(t-2\tau_k) \subseteq A_k(S)$, i.e., at time $t-2\tau_k$ node v only knows tokens generated by the nodes of S.

Since no new tokens were added to $Tokens_v$ between time $t - 2\tau_k$ and time t, we must have $A_k(S') = A_k(S)$; in other words, the nodes of $S' \setminus S$ did not generate any tokens that were not already generated by the nodes of S. We will show that this is unlikely.

From the level-k termination criterion, at least $\ell_k/3$ tokens were not generated by the nodes of S. Each of these tokens is generated by each node of $S' \setminus S$ with probability $1/2^{k+2}$. Because we assumed that $N^t(v) \neq V$ and the graph is strongly connected, $|S' \setminus S| \geq \tau_k$. Hence, for each token $(k, i) \notin A_k(S)$, we can show that $\Pr[(k, i) \in A_k(S' \setminus S)] \geq \tau_k/2^{k+3}$ independently of the other tokens. It follows that $\Pr[A_k(S') = A_k(S)] \leq \left(1 - \tau_k/2^{k-3}\right)^{\ell_k/3} \leq e^{-\ell_k^2/(3\cdot 2^{k+3}\beta)}$. □

Combining the two lemmas, we see that choosing ℓ_k as $\Theta(\sqrt{2^k \beta \ln N})$ yields a polynomially small probability of any node not halting at time $D + O(L_{\hat{k}}/\beta) = D + \tilde{O}(\sqrt{n/B})$, or halting before it has heard from all n nodes.

Theorem 7. *For any constant c, if $\ell_k \geq \sqrt{3(c+2)\beta \cdot 2^{k+3} \ln N}$, then with probability at least $1 - 1/N^c$ each node v halts at a time $t = D + O(L_k/\beta) = \tilde{O}(D + \sqrt{n/B})$ such that $N^t(v) = V$.*

6 Conclusion

Data aggregation problems are traditionally studied in models that feature symmetric point-to-point communication. However, wireless networks can have *asymmetric* communication topologies, due to the effects of local interference and heterogeneous power assignments. This motivates our interest in directed networks with communication by local broadcast.

Our results show that the traditional strategy of first computing a spanning tree, and then solving various distributed tasks using the tree, is not always optimal for directed networks; for example, while computing a rooted spanning tree can require $\Omega(D + n/B)$ rounds (as we saw in Section 4), certain data aggregates can be computed or approximated in $\tilde{O}(D + \sqrt{n/B})$ rounds. Our lower bounds also imply that it is not possible to quickly compute a small-diameter symmetric spanning subgraph of a directed network with diameter 2. In general it seems that "topology-oblivious" algorithms, such as the algorithm in Section 5.2 and gossip algorithms [6, 11], may be better suited for directed networks.

We leave open the question of finding a tight bound on the deterministic time complexity of computing a sensitive function; is there a *deterministic* algorithm that matches the $\Omega(D+\sqrt{n/B})$ lower bound, or can the lower bound be strengthened? For technical reasons, it seems unlikely that a two-party reduction of the style we used in this paper will yield a stronger lower bound, but perhaps multi-party communication complexity lower bounds could be used.

References

1. B. Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems (detailed summary). In *Proc. 19th ACM Symp. on Theory of Computing (STOC)*, pages 230–240, 1987.
2. A. Chakrabarti and O. Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. In *Proc. 43rd ACM Symp. on Theory of Computing (STOC)*, pages 51–60, 2011.
3. G. N. Frederickson and N. A. Lynch. The impact of synchronous communication on the problem of electing a leader in a ring. In *Proc. 16th ACM Symp. on Theory of Computing (STOC)*, pages 493–503, 1984.
4. P. Indyk and D. Woodruff. Tight lower bounds for the distinct elements problem. In *Proc. 44th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 283 – 288, oct. 2003.
5. B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992.
6. D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 482–491, 2003.
7. F. Kuhn, T. Locher, and S. Schmid. Distributed computation of the mode. In *Proc. 27th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 15–24, 2008.
8. F. Kuhn, T. Locher, and R. Wattenhofer. Tight bounds for distributed selection. In *Proc. 19th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 145–153, 2007.
9. F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proc. 42nd ACM Symp. on Theory of Computing (STOC)*, pages 513–522, 2010.
10. E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.
11. D. Mosk-Aoyama and D. Shah. Fast distributed algorithms for computing separable functions. *IEEE Transactions on Information Theory*, 54(7):2997–3007, 2008.
12. A. Negro, N. Santoro, and J. Urrutia. Efficient distributed selection with bounded messages. *IEEE Trans. Parallel and Distributed Systems*, 8(4):397–401, 1997.
13. B. Patt-Shamir. A note on efficient aggregate queries in sensor networks. In *Proc. 23rd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 283–289, 2004.
14. A. A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106:385–390, December 1992.
15. N. Santoro, M. Scheutzow, and J. B. Sidney. On the expected complexity of distributed selection. *J. Parallel and Distributed Computing*, 5(2):194–203, 1988.
16. N. Santoro, J. B. Sidney, and S. J. Sidney. A distributed selection algorithm and its expected communication complexity. *Theoretical Computer Science*, 100(1):185–204, 1992.
17. L. Shrira, N. Francez, and M. Rodeh. Distributed k-selection: From a sequential to a distributed algorithm. In *Proc. 2nd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 143–153, 1983.
18. D. M. Topkis. Concurrent broadcast for information dissemination. *IEEE Trans. Softw. Eng.*, 11:1107–1112, October 1985.