Artificial Intelligence 276 (2019) 79–104

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Distributed monitoring of election winners ☆

Arnold Filtser *,[1], Nimrod Talmon *,[2]

Ben-Gurion University of the Negev, Israel

ARTICLE INFO

Article history:
Received 6 May 2018
Received in revised form 21 July 2019
Accepted 31 July 2019
Available online 5 August 2019

Keywords:
Voting
Distributed streams
Sublinear algorithms

ABSTRACT

We consider distributed elections, where there is a center and k sites. In such distributed elections, each voter has preferences over some set of candidates, and each voter is assigned to exactly one site such that each site is aware only of the voters assigned to it. The center is able to directly communicate with all sites. We are interested in designing communication-efficient protocols, allowing the center to maintain a candidate which, with arbitrarily high probability, is guaranteed to be a winner, or at least close to being a winner. We consider various single-winner voting rules, such as variants of Approval voting and scoring rules, tournament-based voting rules, and several round-based voting rules. For the voting rules we consider, we show that, using communication which is logarithmic in the number of voters, it is possible for the center to maintain such approximate winners; that is, upon a query at any time the center can immediately return a candidate which is guaranteed to be an approximate winner with high probability. We complement our protocols with lower bounds. Our results are theoretical in nature and relate to various scenarios, such as aggregating customer preferences in online shopping websites or supermarket chains and collecting votes from different polling stations of political elections.

© 2019 Elsevier B.V. All rights reserved.

1. Introduction

Elections are extensively used to aggregate preferences of voters. Some elections are centralized, but others are carried out in distributed settings. Consider, for example, a supermarket chain consisting of a large number of stores. Each store collects data on the purchases made in it, and the managers at the chain headquarters might want to aggregate this data, to identify, for example, the most popular items being sold. One solution would be to have a central database, collecting all data from all stores, and to compute the most popular items on this centralized database. As the number of customers might be huge, however, it might not be practical to do so. Further, as the communication between the stores and the headquarters might be expensive, a more efficient solution would be to have some computations being made locally at each store, and to develop a protocol for efficient communication between the stores and the headquarters, to allow the managers at the headquarters to know, at each point in time, what are the most popular items that are being sold throughout the chain.

☆ A preliminary version of this paper was presented at the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '17) [21]. This full version contains all proofs, has improved upper bounds, considers more voting rules, studies further lower bounds, and discusses several issues in more detail.

* Corresponding authors.
E-mail addresses: arnoldf@cs.bgu.ac.il (A. Filtser), talmonn@bgu.ac.il (N. Talmon).

[1] The research was supported in part by ISF grant (1718/18) and by BSF grant 2015813.
[2] Nimrod Talmon was supported by a postdoctoral fellowship from I-CORE ALGO.

https://doi.org/10.1016/j.artint.2019.07.007
0004-3702/© 2019 Elsevier B.V. All rights reserved.

A similar situation happens in online shopping websites, where buyers from all around the world make purchases. As the design of modern websites is based on data centers, aggregating the data concerning all buyers involves communicating in a distributed setting. Specifically, in order to identify the current trends, and as communication between data centers might be expensive, it is of interest to develop protocols for those data centers to communicate with a central entity.

Our model also captures scenarios of political polls and political elections. That is, in political elections and in TV polls, it is usually the case that there are several polling stations, spread around the country or the region. Then, in order to compute the results of the election (or the intermediate results during the day when the poll is being held), the voters' preferences from all those polling stations are aggregated at some central station. For example, in the general political elections held in Brazil in 2014, there were roughly 500,000 polling stations, with an average of 300 voters per station. In this situation, it is beneficial to have a protocol allowing the polling stations to efficiently communicate with a central entity, allowing the central entity to maintain a good estimate on the nation-wide (or region-wide) state of affairs.

In this paper, we model such situations as follows. We are considering an election whose electorate is distributed into k sites. Assuming some common axis of time,[3] we have that at each point in time, a new voter arrives and votes, and her vote is assigned to one of those k sites,[4] There is some center which is able to directly communicate with each of the k sites. With respect to a voting rule $\mathcal{R}$, the goal of the center is to maintain, at any point in time, a candidate which is an $\mathcal{R}$-winner of the whole election (given an election E and a voting rule $\mathcal{R}$, an $\mathcal{R}$-winner of E is a winner of E under $\mathcal{R}$). More specifically, we are interested in designing communication-efficient protocols, where the center is able, upon request at any time, to return a candidate which, with high probability, is an $\mathcal{R}$-winner.

As we are interested in sublinear communication, in addition to allowing mistakes to accrue with some low probability, we will also use approximation. We call a candidate an ϵ-winner with respect to a voting rule $\mathcal{R}$, if by adding up to an ϵ-fraction of new voters, it can become an $\mathcal{R}$-winner. A more formal description of our model and a discussion on our notion of approximation is given in Section 2. Previous works were concerned with bribery (where we are allowed to change an ϵ-fraction of the voters), and margin of victory (where we are guaranteed that by changing an ϵ-fraction of the voters, the outcome of the election shall remain unchanged), see Section 1.1 for additional details on these notions. These notions are appropriate to deal with noisy data, or to be used in scenarios where some external agent can influence the voters, thus change their votes. Here, however, we are concerned with monitoring an election while minimizing the communication, and the source of our errors is lack of information (rather than noise). Our approximation notion fits better to our scenario, as a candidate is an ϵ-winner if it might become a winner under full information. Furthermore, in monitoring an election we do expect more voters to come, thus, in this aspect, an ϵ-winner is a candidate who might become a winner very shortly. Finally, as we consider an ongoing election, changing previous votes is not an option. However, the information on whether a candidate is an ϵ-winner is very valuable for making, e.g., real-time election policy decisions.

We concentrate on single-winner voting rules, and consider various voting rules, ranging from approval-based rules and scoring rules, to tournament-based rules and round-based voting rules; while we naturally cannot cover all voting rules available, we choose some of the more popular and representative ones as well as aim at choosing representative voting rules. Further, we develop some general techniques for designing protocols for maintaining approximate winners in distributed elections, which might be applicable to other voting rules and settings as well. We show how to apply these techniques for the rules we consider. We discuss the effect of several parameters on the communication complexity of the protocols we design; specifically, the effect that the number n of voters, the number m of candidates, the required approximation ϵ, and the number k of sites have on the amount of communication used by our protocols. We complement our communication-efficient protocols with lower bounds.

As a by-product of our lower bounds for maintaining an approximate Plurality winner in distributed elections, we have two contributions which might be useful in other contexts. First, we improve the state-of-the-art lower bound on the Count-tracking problem, which is a central problem in distributed streams; this result is discussed in detail in Remark 5. In short, in the Count-tracking problem, the task is to maintain a value which approximates the number of items in a given distributed stream. In the regime where $k \geq 1/\epsilon^2$, we improve the lower bound for Count-tracking from $\Omega(k)$, proved by Huang et al. [22, Theorem 2.3], to $\Omega(k \log n / \log k)$ (see Remark 5). Second, we define a novel problem in multiparty communication complexity and show a tight lower bound for it; in this problem, which we call the *No Strict Majority* problem, we have several players, each with its own private binary string, and, by communicating bits, the players should decide whether there is some index for which a majority of the players have 1 in it. We prove a lower bound on the *No Strict Majority* problem, showing that the naive protocol for this problem is essentially optimal: asymptotically, all the bits have to be transmitted. See Section 5 for further details on our lower bounds and their implications to continuous distributed monitoring and to multiparty communication complexity.

1.1. Related work

We first review related work on sublinear algorithms in computational social choice, as the current paper fits naturally within this line of research. Then we review papers on compilation complexity, vote elicitation, and mention some connec-

[3] To avoid confusion, let us mention that, while we indeed speak about "time", we do not consider any external clocks (or, importantly, clocks accessible to the sites or the center). In particular, the voters can be assumed to come at fixed intervals, whose speed is not known to the sites nor to the center.

[4] For convenience, we refer to voters as females, while the candidates are males.

tions between our notion of approximation to work on control and bribery in elections (as well as to the concept of margin of victory). Finally, we give an overview on the available literature on the continuous distributed monitoring model, which is the computational model we use in the current paper (its formal definition is given in section 2).

Sublinear social choice. As the amount of data in general, and data concerning preferences in particular, is consistently increasing, the study of identifying election winners using time or space which is sublinear in the number of voters is receiving increasing attention. Specifically, the size of some elections might be too big to process in linear time, thus algorithms with sublinear time and/or space complexity are of interest.

We first mention a follow-up to the conference version of this paper which was recently published [12]. That paper provides a different protocol for winner tracking which is based on our checkpoints technique. Their technique achieves an improvement of a $(1 + \frac{\log k}{\log n/k})$ over our checkpoints-based protocols. Notably, they also performed computer simulations to evaluate the practical communication complexity of theirs and our protocols.

In two papers, Bhattacharyya and Dey [15,2] study sampling algorithms for winner determination as well as winner determination in the streaming model. In fact, some of our sampling-based protocols are inspired by Bhattacharyya and Dey [15]. In their model, they assume that they are given an election in which the margin of victory is at least ϵn (where n is the number of voters); this means that the winner is guaranteed to remain such even if an adversary is allowed to change ϵn votes. Given such elections, they evaluate the number of vote samples needed in order to identify the winner with high probability. In our current paper, we have a different notion of approximation and we do not assume such margins of victory (we formally describe our notion of approximation in Section 2).

Remark 1. There is a mistake in the preliminary version of this work [21], which claims that the sampling-based protocols are implied by the work of Bhattacharyya and Dey [15,2]. This is incorrect as our notion of approximation is different than theirs, specifically due to this margin of victory assumption which in particular means that, while an approximate winner under our definition always exists, this does not necessarily hold in their model.

In a recent paper, Dey et al. [18] study winner determination for several multiwinner voting rules aiming at proportional representation. Dey and Narahari [17] study sampling algorithms for estimating the margin of victory. These works deal with centralized elections, while the current paper considers distributed elections. Another paper worth mentioning in this context is the paper of Lee et al. [25] which argues for the importance of developing fast communication-efficient protocols for computing winner in (centralized) streams; they also provide a simple sampling-based algorithm for approximating Borda winners.

Not strictly considering sublinear social choice, but nonetheless concentrating on "huge elections", in a recent paper, Csar et al. [8] study winner determination using the MapReduce framework which may allow processing such elections efficiently by distributing the computation among clusters of machines.

Compilation complexity. In a series of papers, Chevaleyre et al. [7,6] and Xia and Conitzer [31] define and study the compilation complexity of various voting rules; in their model, the electorate is partitioned into two parts, and the general concern is the amount of communication which needs to be transmitted between the two parts, in order to determine an election winner. In compilation complexity there are no rounds of communication, as only one message is being passed between the two parts. This stands in contrast to our protocols, which use small amounts of communication due to their use of several rounds of communication between the center and the sites.

Vote elicitation. There is quite an extensive literature which deal with vote elicitation [16,13,25,24]; these works provide algorithms for finding approximate winners under various voting rules, by elicitating the voters' preference orders. Conitzer and Sandholm [14] study communication complexity for various voting rules. In their model, each voter acts as a site. Conitzer and Sandholm were interested in finding exact winners. In a follow up paper, Service and Adams [29] allowed approximation. For example, for Borda, they define the approximation ratio of a candidate c to be $\frac{\mathrm{sc}(c)}{\mathrm{sc}(w)}$, where w is the true winner and sc is the Borda score function. Their goal is to find a candidate with $1 - \epsilon$ approximation ratio; approximation ratios for other voting rules are defined similarly. However, as each voter acts as a site, their upper bounds are quite high: In particular, they depend linearly on the number of voters. It is also interesting to mention that our notion of approximation is strictly stronger than theirs, at least for Borda (that is, an ϵ-Borda winner has $O(\epsilon)$ approximation ratio under the definition used by Service and Adams [29]).

Approximate winners, margin of victory, and election control. In the current paper we do not require our protocols to maintain exact winners, but are satisfied with approximate winners. We formally define our notion of approximation in Section 2; roughly speaking, we consider a candidate to be an approximate winner if it can become a winner if we are allowed to add a small number of additional voters (where we can set their votes as we wish). Our notion of approximation somehow resembles the vast amount of research done on electoral control and bribery in elections (see, e.g., the survey by Faliszewski and Rothe [20]). In electoral control by adding voters, there is usually a set of unregistered voters, and the question is whether it is possible to change the outcome of the election, e.g., to have some predefined, preferred candidate to become a winner in a new election, where a small number of those unregistered voters are added to the election.

In bribery problems, such as shift bribery and swap bribery [19], an external agent can change the way some voters vote in order to have some predefined, preferred candidate to become a winner. As observed by Xia [33], the number of such

changes that needs to be done in order to make a specific candidate to become a winner (the so-called margin of victory), is a natural notion of this candidate's closeness to be a winner. Indeed, in this sense, our approximation notion is related to those notions of control and bribery in elections.

Continuous distributed monitoring. The model of computation which we study in the current paper is called the *continuous distributed monitoring* model, and is usually studied within theoretical computer science and database systems. There is a fairly recent survey about this model [11], as well as quite extensive line of work studying various problems in this model, such as sampling-based protocols [10,30], protocols for approximating moments [9,1], protocols for counting with deletions [26] (interestingly, that paper specifically mentions elections as a motivation, but do not study it explicitly), heuristic protocols for monitoring most-frequent items [3], and randomized protocols for counting the number of items in a distributed stream and finding frequent items [22]. In the current paper we complement this line of work by studying winner determination in this model.

2. Preliminaries

We begin by providing preliminaries regarding elections and voting rules, continue by describing our notion of approximation, and finish by discussing our model concerning continuous monitoring of distributed streams. We use standard notions from computational complexity. For $n \in \mathbb{N}$, we denote the set $\{1, \ldots, n\}$ by $[n]$.

2.1. Elections and voting rules

An *election* $E = (C, V)$ consists of a set of *candidates* $C = \{c_1, \ldots, c_m\}$ and a collection of *voters* $V = (v_1, \ldots, v_n)$. We consider both *approval* elections, where voters cast approval ballots, and *ordinal* elections, where voters cast ordinal ballots.

Specifically, in approval elections, each voter is associated with her set of approved candidates, such that $v_i \subseteq C$. We say that v_i *approves* candidate c if $c \in v_i$ (and *disapproves* him otherwise). In ordinal elections each voter is a total order $\succ_{v_i}$ over C. A *single-winner voting rule* $\mathcal{R}$ is a function that gets an election E and returns a set $\mathcal{R}(E) \subseteq C$ of co-winners of that elections, such that c is a winner of the election E under $\mathcal{R}$ if $c \in \mathcal{R}(E)$.

Next we define our voting rules of interest. We ignore issues of tie-breaking; specifically, we assume an arbitrary tie-breaking order which works in our favor, such that a candidate c is a winner if there is some fixed tie-breaking that makes him a winner.

We begin with approval-based voting rules and scoring rules, continue with tournament-based voting rules, and then discuss round-based voting rules.

2.1.1. Approval-based rules and scoring rules

Plurality, *t*-Approval, and Approval. Under *Approval*, each voter approves a subset of the candidates (that is, it is held in approval elections), and the score of a candidate is the number of voters approving him. The candidates with the highest score tie as co-winners. *t-Approval* is similar to Approval, but with the restriction that each voter shall approve exactly t candidates (that is, $|v_i| = t$; we assume that $t \leq m/2$). *Plurality* is a synonym for 1-Approval, that is, where each voter approves exactly one candidate.

Borda. Borda is the archetypical scoring rule. Under *Borda*, a voter ranking a candidate at position j is giving him $m - j$ points, and the candidates with the highest score tie as co-winners. Scoring rules in general are defined similarly, but with scoring vectors other than the one used by Borda.

2.1.2. Tournament-based voting rules

Cup. The *Cup* voting rule is defined via a balanced binary tree T with m leaves, such that there is exactly one leaf for each candidate. Starting from the leaves, in a bottom-up fashion, each non-leaf node is associated with the candidate which wins in the pairwise election held with only the two candidates corresponding to the two children of that node. Finally, the candidate which gets assigned to the root of T is declared the winner of the election.

Copeland and Condorcet. The *Copeland score* of a candidate c is the number of other candidates $c' \neq c$ for which a majority of voters prefer c to c'. Under *Copeland*, the candidates with the highest Copeland score tie as co-winners. A *Condorcet winner* is a candidate with Copeland score $m - 1$. Under *Condorcet*, a Condorcet winner is selected as a winner if it exists; otherwise, all candidates tie as co-winners.

2.1.3. Round-based voting rules

Plurality with run-off. *Plurality with run-off* proceeds in two rounds. In the first round, it selects two candidates with the highest Plurality scores, where the Plurality score of a candidate is defined as the number of voters ranking him first. In the second round, it considers only those two candidates selected in the first round and selects as a winner the one which is preferred to the other by majority of voters.

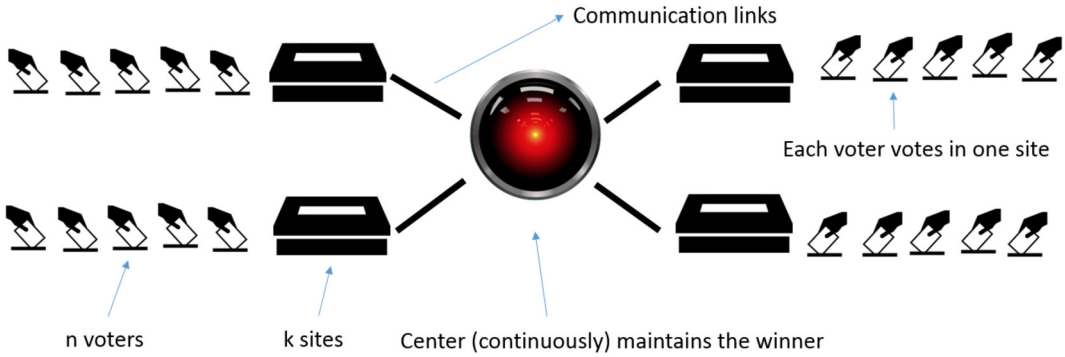

Fig. 1. Illustration of our model.

Bucklin. *Bucklin* also proceeds in rounds. In round $i \in [m]$, it computes, for each candidate c, the number of voters ranking c among their top i choices. Then, if there is a candidate with a strict majority of the voters ranking him among their top i choices, then such a candidate is selected as a winner; otherwise, a new round begins.

2.2. Our notion of approximation

Since we will be interested in designing protocols where the center cannot see the full election, it will not be possible to guarantee that our protocols will find exact winners; therefore, we will be satisfied with protocols which are guaranteed to find approximate winners. There are several possibilities for defining approximate winners of elections; in this paper we consider ϵ-winners. Roughly speaking, an ϵ-winner is a candidate which is not far from being the winner of the election in the sense that he might become a winner after the arrival of only a few additional new voters. A more formal definition follows.

Definition 1 *(ϵ-winner).* For $\epsilon \in (0, 1)$, a candidate c is an *ϵ-winner* in an election E (with n voters) under some voting rule $\mathcal{R}$ if it can become a winner under $\mathcal{R}$ by adding at most ϵn additional voters to E. That is, if there exists an election E', where $E \subseteq E'$ and $|E' \setminus E| \leq \epsilon \cdot n$ such that $c \in \mathcal{R}(E')$.

Indeed, we view the definition of an ϵ-winner as a definition of approximation, as the lower ϵ is, the closer an ϵ-winner is to a real winner. As we will design our protocols to compute ϵ-winner, the lower ϵ would be, their guaranteed results would become closer and closer to real winners.

Our approximation notion seems particularly relevant to our setting (as compared to, e.g., the notion used by Bhattacharyya and Dey [15,2]), for the following reasons. First, we do not assume a margin-of-victory, namely that some candidate is a clear winner. Second, in distributed vote streams we expect more voters to arrive in the future, thus we are interested in identifying a candidate which might become a winner in the near future: such candidates are exactly the ϵ-winners.[5]

2.3. Our model of computation

In our computational model, we have one center and k sites. The center and the sites are arranged in a star-shaped network, centered at the center, such that the center has a direct communication link to each site but two sites cannot communicate directly.

We assume some axis of time, $t_1, \ldots, t_n$, and a stream of voters $v_1, \ldots, v_n$, such that voter v_i comes at time t_i. Each voter is assigned to exactly one site, such that each site is aware only of the subset of voters which are assigned to it. We stress that the time is not known to either the center or the sites. Such a stream is called a *distributed stream*. Fig. 1 illustrates the model. Table 1 provides an example of a distributed stream.

We mention that our model of computation might be seen as the model of computation assumed in the study of *Continuous Distributed Monitoring*, when instantiated for vote streams (and not general, abstract streams). See the Related Work section for more details on this subject.

We are interested in designing communication-efficient protocols, whose goals are to allow the center to declare, at any point in time, a candidate c which is, with constant probability (say, 0.9), an ϵ-winner (see Section 4 for a discussion on higher probabilities). Formally, for a voting rule $\mathcal{R}$, we are interested in finding efficient protocols for the $\mathcal{R}$-WINNER-TRACKING problem.

[5] As a side note, we mention that in political elections such a knowledge might worth much to these candidates, as it can help them decide on when to spend their campaigning funds.

Table 1
An example of a distributed stream corresponding to a Plurality election over the candidate set $\{a, b\}$. Rows correspond to the exist of time, $t_1, \ldots, t_n$, while each column corresponds to a site; e.g., at time 3, a voter voting for b is assigned to site S_2. Notice that the situation at a certain time can be viewed as the current election; for example, the election at time 4 consists of 3 votes for 1 and 1 vote for b. For this election, notice that, while a is the winner, and thus an ϵ-winner for any $0 \le \epsilon \le 1$, b is not an ϵ-winner even for $\epsilon = 0.5$, as even adding $\epsilon n = 0.5 \cdot 4 = 2$ votes for b would not make b a Plurality winner.

t	S_1	S_2	S_3
1	a		
2	a		
3		b	
4			a
5			a
6		b	
7		b	

A protocol is defined via the messages which the center and the sites send to each other, and can consist of several rounds. The protocol shall be correct not only at the end of the stream (which is usually the case in streaming algorithms), but shall be correct at any point in time. As it is the custom in protocols operating on distributed streams, we describe our upper bounds in terms of words of communication, where we assume that a word contains $\log n$ bits.

We measure the communication complexity of a protocol with respect to the expected number of words communicated in its run. Using Markov inequality, this expectation can be replaced by high probability (or even by probability 1 in the expense of slight increase in the failure probability).

2.4. Useful results from probability theory

For the sampling based protocols, we will use the following bound.

Theorem 1 *(Chernoff Bound). Let $X_1, \ldots, X_s$ be a sequence of s i.i.d random variables in $[0, 1]$. Let $X = \sum_i X_i$ and let $\mu = \mathbb{E}[X]$. Then, for any $0 \le \gamma \le 1$:*

$$\Pr[|X - \mu| \ge \gamma\mu] < 2\exp(-\gamma^2\mu/3) .$$

The following corollary will be the main building block for our sampling-based protocols.

Lemma 1. *Let $X_1, \ldots, X_s$ be i.i.d. random variables in $[0, 1]$ with mean p. Let $X = \sum_i X_i$ and let $q = \frac{1}{s}X$. Then, for $s \ge \frac{3}{\epsilon^2}\log(\frac{2}{\delta})$ it holds that*

$$\Pr[|q - p| \ge \epsilon] < \delta .$$

Proof. Set $\mu = \mathbb{E}[X] = s \cdot p$. Using the Chernoff Bound (i.e., Theorem 1), it follows that:

$$\Pr[|q - p| \ge \epsilon] = \Pr\left[|X - \mu| \ge \frac{\epsilon}{p} \cdot \mu\right] \le 2\exp\left(-\left(\frac{\epsilon}{p}\right)^2 \cdot \mu/3\right) \le 2\exp\left(-\epsilon^2 s/3\right) \le \delta . \qquad \square$$

3. Algorithmic techniques

The naive protocol, where each site sends to the center a message for every voter that arrives to it, clearly solves our problem, however it uses communication which is linear in the number of voters. For example, for ordinal ballots, it communicates $O(n \cdot m \log m)$ bits, since $m \log m$ bits are sufficient for sending a single vote. In this paper we are interested in protocols which use significantly less communication, namely communication which is polylogarithmic in the number of voters.

In this section we provide high level descriptions of three algorithmic techniques which are useful for developing protocols for maintaining approximate winners in distributed vote streams. Accordingly, in Section 4 we demonstrate how to realize and instantiate these algorithmic techniques as concrete protocols for maintaining approximate winners for various specific voting rules.

3.1. Protocols based on counting frequencies

In the Frequency-tracking problem, we are given a distributed stream where, instead of voters, the items of the stream come from a known universe of items. The goal is for the center to maintain, for each item type in the distributed stream, a value which approximates the frequency of that item type. More formally, let us denote the items of the stream by $v_1, \ldots, v_n$ and consider m different item types, such that item $i \in [n]$ is of type $j \in [m]$ if $v_i = j$. Let us denote the frequency of item type j by $f(j) = |\{i : v_i = j\}|$. A protocol solving the Frequency-tracking problem guarantees that, with constant probability, simultaneously for every item type j, the center can maintain a value $f'(j)$ such that $f'(j) \in f(j) \pm \epsilon n$.

Estimating the frequencies of item types is a fundamental problem in distributed streams (in fact, also in centralized streams). A deterministic protocol for Frequency-tracking, using $O(\epsilon^{-1} k \log n)$ words of communication, is known [34]; in fact, it is also known to be tight. Moreover, there is a randomized protocol which uses $O((\epsilon^{-1}\sqrt{k} + k) \log n \log \frac{1}{\delta})$ words of communication, and succeeds with probability $1 - \delta$ [22].[6] Formally, the protocol guarantees that for every $j \in [m]$ and every n, after the arrival of n items, $\Pr[f'(j) \in f(j) \pm \epsilon n] \geq 1 - \delta$. In particular, by setting $\delta = 1/\text{poly}(m)$ and applying the union bound, we get that for every n, $\Pr[\forall j, \quad f'(j) \in f(j) \pm \epsilon n] \geq 1 - \frac{1}{\text{poly}(m)}$. The communication complexity in this case is $O((\epsilon^{-1}\sqrt{k} + k) \log n \cdot \log m)$.

Many voting rules operate by counting points for candidates, thus, it can be seen as if these voting rules actually count frequencies of, say, approvals of each candidate. It turns out that, indeed, it is sometimes possible to reduce the problem of maintaining an ϵ-winner under such voting rules to the problem of maintaining approximate frequencies. For more concrete intuition, consider the following distributed stream for the Frequency-tracking problem.

Example 1. Consider the set of $m = 3$ item types $\{1, 2, 3\}$ and a distributed stream containing the following $n = 10$ items: $[2, 2, 1, 3, 3, 3, 3, 3, 2, 1]$. Then, the frequencies f of the item types is: $f(1) = 1$, $f(2) = 3$, and $f(3) = 5$. Observe how the above stream might, roughly speaking, correspond to a distributed vote stream for Plurality-winner-tracking containing 10 votes over the set of candidates $\{1, 2, 3\}$, and how, in essence, Plurality simply counts the frequencies of the corresponding items.

During the description of our results for specific voting rules, in Section 4, we will usually use the randomized version of the Frequency-tracking protocol, the only exception being the hybrid protocol for Runoff, for which we will use the deterministic version.

3.2. Protocols based on checkpoints

Protocols based on checkpoints are deterministic in nature, and the general idea behind such protocols is as follows. Assume that the center knows an ϵ-winner c of the election containing the first n voters. Then, the crucial observation is that, until the number of voters reaches $(1 + \epsilon)n$, the center can declare c as an $O(\epsilon)$-winner. This suggests protocols where the center only updates its declared candidate whenever the number of voters multiplies by a $(1 + \Omega(\epsilon))$-fraction. Such points of time will be called *checkpoints*. Between two checkpoints, the center will declare the previous estimation as the current ϵ-winner. This intuition is formulated in the following lemma, the proof of which appears in Appendix A.[7] Algorithms 1 and 2 provide pseudocode for a general protocol based on checkpoints.

Algorithm 1: Protocol based on checkpoints, for center.

```
Initiate and maintain an approximate count n' using a Count-tracking protocol with λ = ε/12;
if n' ≥ (1 + λ)^i for some i, for the first time for this i then
    Initiate a static subprotocol to identify an ε/4-winner c;
Upon query: declare c;
```

Algorithm 2: Protocol based on checkpoints, for site.

```
Participate in the Count-tracking protocol initiated by the center;
Participate in each static subprotocol initiated by the center;
```

[6] Notice that Huang et al. [22] consider only situations where $k \leq \epsilon^{-2}$, thus their bounds read differently; nevertheless, $O((\epsilon^{-1}\sqrt{k} + k) \log n \cdot \log \frac{1}{\delta})$ is the communication complexity of their protocol.

[7] While some of the ideas in the proof might fit naturally in the main text, the proof considers each voting rule studied in this paper separately, and thus it is slightly repetitive, and thus deferred to the appendix.

Lemma 2. *Let $\mathcal{R}$ be some voting rule described in Section 2.1. Let $E = \{v_1, \ldots, v_n\}$ and $E' = E \cup \{v_{n+1}, \ldots, v_{n+q}\}$, where $q \le \frac{\epsilon}{4}n$, be two elections. If candidate c is an $\frac{\epsilon}{4}$-winner w.r.t. E, then c is an ϵ-winner w.r.t E'.*

In order to identify the checkpoints, the center shall be able to count the number of voters arriving so far. Fortunately, for an approximation factor $\lambda \in (0, 1)$, there is an efficient deterministic protocol for solving the Count-tracking problem, which uses $O(\lambda^{-1}k \log n)$ words [34]; in the Count-tracking problem, the center maintains a value n' such that $n' \in n \pm \lambda n$, where n is the actual number of items in the distributed stream.

Now we have all the ingredients for our generic protocol. Specifically, the center will maintain a value n' using a Count-tracking protocol with precession parameter $\lambda = \frac{\epsilon}{12}$. Each time n' exceeds $(1+\lambda)^i$ for the first time,[8] for some i, the center will initiate a *static* subprotocol to identify an $\frac{\epsilon}{4}$-winner c of the election so far. The center will declare c as ϵ-winner until the next checkpoint. We argue that c is indeed an ϵ-winner. Consider a step in time n. Then the center's estimation n' of the number of voters is at least $(1-\lambda)n$. In particular, it necessarily had a "checkpoint" at time n'', for $n'' \ge \frac{1-\lambda}{1+\lambda}n$. Thus $n \le (1+3\lambda)n'' = (1+\frac{\epsilon}{4})n''$. By Lemma 2, as c was $\frac{\epsilon}{4}$-winner at time n'', it is also ϵ-winner at time n.

As the estimation n' is upper bounded by $(1+\lambda)n$, the number of checkpoints is bounded by $\log_{1+\lambda}((1+\lambda)n) = O(\log n/\lambda) = O(\log n/\epsilon)$. Assuming that it is possible to compute an $\frac{\epsilon}{4}$-winner using $O(z)$ words, a protocol based on checkpoints would then need $O((k+z)\epsilon^{-1}\log n)$ words of communication. As z will be at least $\Omega(k)$, we would get $O(z \cdot \epsilon^{-1}\log n)$.[9] We conclude with the following:

Corollary 1. *Let $\mathcal{R}$ be a voting rule from those described in Section 2.1. Suppose that there is a static deterministic protocol that computes an $\frac{\epsilon}{4}$-winner using z words of communication, for $z \ge k$. Then, there is a protocol for $\mathcal{R}$-winner-tracking which uses $O(z \cdot \epsilon^{-1}\log n)$ words.*

During the description of our results for specific voting rules, in Section 4, we will describe only the static protocol in each protocol based on checkpoints. For simplicity of presentation, we will compute ϵ-winner instead of $\frac{\epsilon}{4}$-winner as actually needed.

3.3. Protocols based on sampling

Instead of sending all voters to the center, as the naive protocol does, it is natural to let each site send only some of the voters arriving to it. Specifically, we would like the center to have a uniform sample of the voters. Cormode et al. [10] describe a protocol for maintaining a sample of s items chosen uniformly at random from a distributed stream; its communication complexity is $O((k+s)\log n)$. Since we are sampling voters, we need to take into account the communication needed to send each of the sampled voters. Specifically, in approval elections (where the voters cast approval ballots), we need m bits per voter. Since we count the communication complexity in words, each of which contains $\log n$ bits, we need $\lceil \log 2^m/\log n \rceil \le 1 + m/\log n$ words per voter. Similarly, in ordinal elections (where the voters cast ordinal ballots), we need $(\log m!)$ bits per voter, thus $\lceil \log m!/\log n \rceil \le 1 + m\log m/\log n$ words per voter.

But how many samples are needed in order to determine an ϵ-winner with high probability? Our main building block will be Lemma 1 (see Section 2) and our general framework is as follows. For each voting rule, we will use Lemma 1 to argue that, with s samples, chosen uniformly with repetitions, we can determine an ϵ-winner with high probability. Then, assuming that we need w words of communication for each voter, using an efficient sampling protocol [10], as discussed above, we will get a communication protocol with complexity $O((k+s)w \cdot \log n)$. (As we use asymptotic analysis, it will be enough to find an $O(\epsilon)$-winner and to adjust the parameters accordingly.)

4. Communication-efficient protocols

Our upper bounds are summarized in Table 2. We begin with approval-based rules and scoring rules, continue with tournament-based rules, and then discuss round-based rules. Before we present our specific upper bounds, the following remark, concerning the success probability of our protocols, is in place.

Remark 2. Notice that we state our results for protocols which are correct with some constant probability, say 0.9. One can always achieve arbitrary high probability $1-\delta$, as follows, and depending on the general technique used:

- For protocols based on counting frequencies, following the discussion in Section 3.1, one can get failure probability δ by replacing the $\log m$ term with a $\log \frac{m}{\delta}$ term in the communication complexity.
- Protocols based on checkpoints are deterministic anyhow.

[8] In fact, the Count-tracking protocol of [34] only increases its estimation as time goes by.

[9] Huang et al. [22] provide a randomized protocol for Count-tracking which uses $O(\sqrt{k}\epsilon^{-1}\log n)$ bits of communication. As z will be greater than $\Omega(k)$, using randomization will not reduce the total asymptotic communication.

Table 2
Overview of our results. ϵ is the required approximation, k is the number of sites, m is the number of candidates, and n is the number of voters. There are three columns of upper bounds, where the first is for protocols based on counting frequencies, the second is for protocols based on checkpoints, and the third is for sampling-based protocols. The results in the first column and in the third column correspond to randomized protocols, while the results in the second column correspond to deterministic protocols. For Plurality with run-off, the second protocol is actually a hybrid between checkpoints to (deterministic) frequency count. For Cup and Condorcet, one might also use the checkpoints protocol of Copeland. In our randomized lower bound we make the usual assumptions that there is no spontaneous communication.

Upper bounds			
Voting rule	Frequencies	Checkpoints	Sampling
Plurality	$O((\epsilon^{-1}\sqrt{k}+k)\log n\cdot\log m)$		
t-Approval	$O((\epsilon^{-1}\sqrt{kt}+k)\log tn\cdot\log m)$	$O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon}+\log n)\right)$	$O(\epsilon^{-2}\log(2t)+k)(\log\binom{m}{t}+\log n)$
Approval	$O((\epsilon^{-1}\sqrt{km}+k)\log mn\cdot\log m)$	$O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon}+\log n)\right)$	$O((\epsilon^{-2}\log m+k)(m+\log n))$
Borda	$O((\epsilon^{-1}\sqrt{km}+k)\log mn\cdot\log m)$	$O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon}+\log n)\right)$	$O((\epsilon^{-2}\log m+k)(m\log m+\log n))$
Condorcet	$O((\epsilon^{-1}\sqrt{km^2}+k)\log mn\cdot\log m)$	$O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon}+\log m\cdot\log n)\right)$	$O((\epsilon^{-2}\log m+k)(m\log m+\log n))$
Copeland	$O((\epsilon^{-1}\sqrt{km^2}+k)\log mn\cdot\log m)$	$O\left(\frac{k}{\epsilon}(m^2\log\frac{k}{\epsilon}+\log n)\right)$	$O((\epsilon^{-2}\log m+k)(m\log m+\log n))$
0 Cup	$O((\epsilon^{-1}\sqrt{km^2}+k)\log mn\cdot\log m)$	$O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon}+\log m\cdot\log n)\right)$	$O((\epsilon^{-2}\log m+k)(m\log m+\log n))$
Run Off	$O((\epsilon^{-1}\sqrt{km^2}+k)\log mn\cdot\log m)$	$O\left(\frac{k}{\epsilon}\log n\right)$	$O((\epsilon^{-2}+k)(m\log m+\log n))$
Bucklin	$O\left((\epsilon^{-1}\sqrt{km}\log^2 m+k)\log mn\cdot\log m\right)$	$O(\frac{k\log m}{\epsilon}(m\log\frac{k}{\epsilon}+\log n))$	$O((\epsilon^{-2}\log m+k)(m\log m+\log n))$
Lower bounds			
All rules	$\Omega((\epsilon^{-1}\sqrt{k}+k)\log n/\log k)$		Already for $m=2$.
Approval	$\Omega\left(\epsilon^{-1}km\cdot\log(n/k)\right)$		For deterministic protocols.

Algorithm 3: Protocol for PLURALITY-WINNER-TRACKING, for center.

Initiate and maintain approximate frequencies $f'(c)$ for each candidate using a FREQUENCY-TRACKING protocol with $\epsilon'=\epsilon/2$;
Upon query: declare $\hat{c}$ for which $f'(\hat{c})$ is the highest;

Algorithm 4: Protocol for PLURALITY-WINNER-TRACKING, for site.

Participate in the FREQUENCY-TRACKING protocol initiated by the center by reducing a vote for candidate j to an item of type j;

- For protocols based on sampling, we mention that, as can be seen from the corresponding proofs, the increase of the required sampling size needed for increasing the success probability is quite small. Specifically, the number of samples will increase: in t-APPROVAL to $O(\epsilon^{-2}\log(\frac{2t}{\delta}))$, in (PLURALITY WITH) RUN OFF to $O(\epsilon^{-2}\log(\frac{1}{\delta}))$, and in all other voting rules to $O(\epsilon^{-2}\log(\frac{m}{\delta}))$.

4.1. Approval-based rules and scoring rules

Let us begin with Plurality, as it is arguably the simplest voting rule. In Plurality, a vote in a distributed vote stream is associated with one candidate out of the m candidates participating in the election, and the goal is for the center to maintain a candidate c such that the highest number of voters vote for c, or at least it is at most ϵn-far from being such a candidate. Equivalently, a distributed stream for Plurality contains m item types (one item type for each candidate). Given an approximate frequency for each type (that is, an approximate number of voters voting for each candidate), the center can safely declare the candidate with the highest approximate frequency.

The next result follows by realizing a straight-forward protocol based on counting frequencies, as described in Section 3.1.

Theorem 2. *There is a protocol for* PLURALITY-WINNER-TRACKING *which uses* $O((\epsilon^{-1}\sqrt{k}+k)\log n\cdot\log m)$ *words.*

Proof. We use the efficient protocol for FREQUENCY-TRACKING [22] with $\epsilon'=\epsilon/2$. This allows the center to maintain, for each candidate c, a value which is guaranteed to be at most $\frac{\epsilon}{2}n$-far from the real number of voters voting for c. The center would declare the candidate $\hat{c}$ for which the approximate frequency is the highest. A pseudocode of the protocol is given in Algorithms 3, 4.

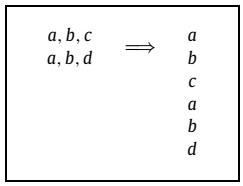

Fig. 2. Reducing t-Approval to Plurality, for $t = 3$. Notice that two t-Approval voters become six Plurality voters.

Let us denote the real frequency of a candidate c by $f(c)$ (which equals its Plurality score), and its approximate frequency computed by the Frequency-tracking protocol by $f'(c)$. For each $c \neq \hat{c}$, it holds that

$$f(c) \leq f'(c) + \frac{\epsilon}{2}n \leq f'(\hat{c}) + \frac{\epsilon}{2}n \leq f(\hat{c}) + \epsilon n$$

where the first and third inequalities follows from the $\epsilon/2$-approximation and the second from our choice of $\hat{c}$. Therefore, we conclude that $\hat{c}$ is an ϵ-winner, as required. □

We go on to consider t-Approval, where each voter specifies t candidates which she approves. We provide three protocols, based on counting frequencies, checkpoints, and sampling, respectively.

Remark 3. Notice that here, as well as for other protocols which we discuss later on, we provide a different upper bound on the communication complexity of each of the protocols we describe. Note that these upper bounds are incomparable, in the sense that each of them is better for different specific values of the parameters. To not clutter the text too much, we defer a discussion on the interplay between the parameters, which relates to the decision of which protocol to choose for a specific scenario to Section 6.

Theorem 3. *There are three protocols for t-Approval-winner-tracking, for $t \leq m/2$. Respectively, the protocols use $O((\epsilon^{-1}\sqrt{k}t + k)\log tn \cdot \log m)$, $O\left(\frac{k}{\epsilon}(m \log \frac{k}{\epsilon} + \log n)\right)$, and $O(\epsilon^{-2}\log(2t) + k)(\log \binom{m}{t} + \log n)$ words of communication.*

Proof. For the first protocol, we reduce t-Approval to Plurality, as follows, and as depicted in Fig. 2. Each site, upon receiving a voter v which approves t candidates, instead of considering the voter v, creates and considers t voters, $v_1, \dots v_t$, such that voter v_i (for $i \in [t]$) is set to approve the ith approved candidate of v. For example, a voter approving $\{a, b, d\}$ would be reduced to three voters, approving a, b, and d, respectively.

The reduced election has $n' = nt$ voters, and will be executed with precision parameter $\epsilon' = \epsilon/2t$. Consider a candidate c which is an ϵ'-winner in the reduced election; we argue that c is an ϵ-winner in the original election. Indeed, we can add ϵn voters, each approving c, while for each other candidate c', at most $\epsilon n/2$ of them approve c' (as $t \leq m/2$); thus, the relative score of c increases by $\epsilon n/2 = \epsilon' n'$. As c is ϵ'-winner in the reduced election, this is sufficient. By Theorem 2, the communication used is $O((\epsilon'^{-1}\sqrt{k} + k)\log n' \cdot \log m) = O((\epsilon^{-1}\sqrt{k}t + k)\log tn \cdot \log m)$.

The second protocol is based on checkpoints. We describe the static protocol for computing an ϵ-winner. The center initiates communication with all sites, asking from each site to send an approximate score for each candidate. That is, each site, for each candidate c, sends the number of voters approving c, rounded to the closest multiplication of $\epsilon n/k$. Such rounding is enough, since, summing up the possible errors from all k sites, the center would have a value which is at most $\epsilon n/2$-far from the real score. Thus, the candidate c with the highest approximated score will indeed be an ϵ-winner. Each site should communicate $\log(\frac{k}{\epsilon})$ bits per candidate. Thus, the total communication is bounded by $k\lceil \frac{m \log \frac{k}{\epsilon}}{\log n} \rceil \leq O(k(1 + \frac{m \log \frac{k}{\epsilon}}{\log n}))$. The bound follows.

For the third protocol, we will show that $s = \frac{24}{\epsilon^2} \ln \frac{2t}{\delta}$ sampled voters, chosen uniformly at random (with repetitions), are enough to determine an ϵ-winner with failure probability at most δ. As we can communicate each voter using $\log \binom{m}{t}$ bits, the bound follows. Consider such a sample of s voters, and, for a candidate c, let X_i^c be an indicator for the event that the i-th sampled voter approved c. Let $X^c = \frac{n}{s}\sum_{i=1}^{s} X_i^c$, and denote by Y^c the actual number of voters that approved c in the original election. Set $\mu = \mathbb{E}\left[\sum_i X_i^c\right] = s \cdot \frac{Y^c}{n}$. Using the Chernoff bound (Theorem 1 in Section 2), we have that:

$$\begin{aligned}\Pr\left[\left|X^c - Y^c\right| \geq \frac{\epsilon}{2}n\right] &= \Pr\left[\left|\sum_i X_i^c - s \cdot \frac{Y^c}{n}\right| \geq \frac{\epsilon}{2}s\right] \\ &= \Pr\left[\left|\sum_i X_i^c - \mu\right| \geq \frac{\epsilon}{2}\frac{n}{Y^c}\mu\right]\end{aligned}$$

$$\leq 2\exp\left(-\left(\frac{\epsilon}{2}\frac{n}{Y^c}\right)^2 \cdot \mu/3\right)$$
$$= 2\exp\left(-\frac{\epsilon^2}{12}\cdot\frac{ns}{Y^c}\right) .$$

By union bound, we have that:

$$\Pr\left[\exists c \text{ s.t. } \left|X^c - Y^c\right| \geq \frac{\epsilon}{2}n\right] \leq 2\sum_c \exp\left(-\frac{\epsilon^2}{12}\cdot\frac{ns}{Y^c}\right) \leq 2t\cdot e^{-\frac{\epsilon^2 s}{12}} \leq \delta ,$$

where the second inequality follows from Claim 1 below, by setting $\lambda = \frac{\epsilon^2 \cdot ns}{12}$ and noting that $(Y^{c_1}, \ldots, Y^{c_m})$ lies in the convex hull of the set A described there. The center will return a candidate c with maximal X^c. Correctness follows by the same arguments as in the frequency-count protocol. □

Claim 1. *Consider the set $\mathbb{N}^m$ of points with m integer coordinates. Let $A \subset \mathbb{N}^m$ contain exactly those points in $\mathbb{N}^m$ for which the value of exactly t coordinates is n, while the value of all their other $m-t$ coordinates is 0. Let $\lambda \geq 2n$. Then, for any arbitrary point $(x_1, \ldots, x_m)$ in the convex hull of A, it holds that:*

$$\sum_{i=1}^{m} e^{-\frac{\lambda}{x_i}} \leq t\cdot e^{-\frac{\lambda}{n}} .$$

Proof. Consider the function $f(x) = e^{-\frac{\lambda}{x}}$ and notice that its second derivative is

$$(f(x))'' = \left(e^{-\frac{\lambda}{x}}\right)'' = \left(\frac{\lambda}{x^2}\cdot e^{-\frac{\lambda}{x}}\right)' = -\frac{2\lambda}{x^3}\cdot e^{-\frac{\lambda}{x}} + \frac{\lambda^2}{x^4}\cdot e^{-\frac{\lambda}{x}} = \frac{\lambda}{x^3}\cdot e^{-\frac{\lambda}{x}}\cdot\left(\frac{\lambda}{x} - 2\right) .$$

Hence, f is convex in the domain $[0, n] \subseteq [0, \lambda/2]$. Set $\hat{f}(x_1, \ldots, x_n) = \sum_{i=1}^{n} f(x_i)$. As sum of convex functions is also convex, $\hat{f}$ is convex in the domain $[1, n]^n$, which in particular contains the convex hull of A. Since $\hat{f}$ is convex function, the maximum value in the convex hull achieved in a point of A. We conclude that:

$$\sum_{i=1}^{m} e^{-\frac{\lambda}{x_i}} = \hat{f}(x_1, \ldots, x_m) \leq \max_{(y_1,\ldots,y_m)\in A} \hat{f}(y_1, \ldots, y_m) = t\cdot e^{-\frac{\lambda}{n}} . \qquad \square$$

For Approval, where the set of approved candidates of each voter can be arbitrary, thus upper bounded by the number m of candidate, we proceed similarly to t-Approval. Naturally, we have m-factors instead of t-factors in our bounds. (Specifically, in the first protocol the size of the reduced election is $n' = mn$ and in the second protocol we sample slightly more voters.)

Theorem 4. *There are three protocols for* Approval-winner-tracking. *Respectively, the protocols use $O((\epsilon^{-1}\sqrt{k}m + k)\log mn \cdot \log m)$, $O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon} + \log n)\right)$, and $O((\epsilon^{-2}\log m + k)(m + \log n))$ words of communication.*

We go on to consider ordinal elections. Specifically, next we consider the Borda rule, for which we describe three protocols.

Theorem 5. *There are three protocols for* Borda-winner-tracking. *Respectively, the protocols use $O((\epsilon^{-1}\sqrt{k}m + k)\log mn \cdot \log m)$, $O(\epsilon^{-1}k(m\log(k/\epsilon) + \log n))$, and $O((\epsilon^{-2}\log m + k)\ (m\log m + \log n))$ words of communication.*

Proof. We start by discussing the impact of adding voters for elections whose results are decided by Borda. For an arbitrary candidate c, consider two voters where one voter is ranking c first and then ranks the other candidates in an arbitrary order, and another voter is ranking c first and then ranks the other candidates in reverse order. Adding these two voters causes an increase to the score of c by $2(m-1)$ while the score of all other candidates increases by $m-2$. Thus, by adding ϵn voters, we can increase the relative score of c by $\epsilon nm/2$.

The first protocol is based on reducing Borda to Plurality, similarly to the first protocol stated in Theorem 3. Specifically, we begin by reducing Borda to Plurality, as follows, and as depicted in Fig. 3: Each site, upon receiving a voter v with preference order $c_1 \succ \ldots \succ c_m$, instead of considering the voter v, creates and considers $\sum_{j\in[m]} m - j < m^2$ voters, such that for $j \in [m]$, it creates $m - j$ voters, each approving c_j. For example, a voter $v : a \succ b \succ d$ would be transformed into three voters, approving a, a, b, respectively.

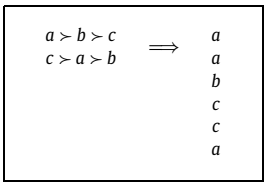

Fig. 3. Reducing Borda to Plurality. Notice that two Borda voters become six Plurality voters.

In the reduced election we have $n' < m^2 n$ voters, where n is the number of voters in the original election. We use the protocol for Plurality described in Theorem 2 with $\epsilon' = \epsilon/(4m)$. Let us denote the real frequency of a candidate c in the reduced election by $f(c)$ and its computed approximate frequency by $f'(c)$. The error is bounded by $\left|f'(c) - f(c)\right| \le \epsilon' n' < \frac{\epsilon}{4m} \cdot nm^2 = \frac{\epsilon nm}{4}$. Since by adding ϵn voters we can increase the relative score of the chosen candidate by $\epsilon nm/2$, we are done.

The second protocol is based on checkpoints, thus below we describe the static subprotocol used in each checkpoint. Similarly to the second protocol in Theorem 3, each site sends an approximation of the Borda score of each candidate rounded to the closest multiplication of $\epsilon nm/k$. Hence the subprotocol uses $O(k(1 + (m \log \frac{k}{\epsilon})/(\log n)))$ words, while the combined error for the Borda score estimation of each candidate is $\epsilon nm/2$.

For the third protocol, we will show that $s = O(\epsilon^{-2} \log \frac{m}{\delta})$ sampled voters, chosen uniformly at random (with repetitions), are enough to determine an ϵ-winner with failure probability at most δ. As we can communicate each voter using $\log(m!)$ bits, the bound follows. For a candidate c, let $X_i^c = \frac{\alpha_i}{m}$, where α_i is the score that candidate c gets from the i's sampled voter. Let $X^c = \frac{n \cdot m}{s} \sum_{i=1}^{s} X_i^c$, and denote by Y^c the score of the candidate c in the election. Set $\mu = \mathbb{E}\left[\frac{1}{s} \sum_i X_i^c\right] = \frac{1}{n \cdot m} Y^c$. Using Lemma 1 we have that

$$\Pr\left[\left|X^c - Y^c\right| \ge \frac{\epsilon}{4} \cdot n \cdot m\right] = \Pr\left[\left|\frac{1}{s} \sum_i X_i^c - \mu\right| \ge \frac{\epsilon}{4}\right] \le \frac{\delta}{m},$$

and hence by union bound it follows that $\Pr\left[\exists c \text{ s.t. } \left|X^c - Y^c\right| \ge \frac{\epsilon}{4} \cdot n \cdot m\right] \le \delta$. The center will return a candidate c with maximal X^c. The accuracy of the protocol follows from arguments given in the analysis of the frequency-count protocol. □

Remark 4. Results for other scoring rules, at least those corresponding to scoring vectors whose values are polynomially bounded, can be achieved by similar techniques. As the corresponding reductions to plurality are quite technical and do not provide new insights to the problem, we do not consider them here.

4.2. Tournament-based rules

In this section we consider Condorcet winners and the Copeland voting rule. The rules we consider below are built upon the tournament defined over the election by considering head-to-head contests between all pairs of candidates. The first protocol for Copeland proceeds by approximating, for each pair of candidates c_1 and c_2, the number of voters preferring c_1 to c_2. Having these approximate counts, we will be able to identify an ϵ-winner under Copeland. If there is a candidate c which is preferred to all other candidates, then the center shall declare c as the Condorcet winner.

Theorem 6. *There are three protocols for* Copeland-winner-tracking. *Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{k}m^2 + k) \log mn \cdot \log m)$, $O\left(\frac{k}{\epsilon}(m^2 \log \frac{k}{\epsilon} + \log n)\right)$, *and* $O((\epsilon^{-2} \log m + k)(m \log m + \log n))$ *words.*

Proof. For the first protocol, we reduce each voter, corresponding to a total order over the candidates, to $O(m^2)$ items; specifically, the reduced distributed stream will contain items of $O(m^2)$ item types, where for each pair of candidates c_1 and c_2 we have a different type, denoted by (c_1, c_2). The reduction proceeds as follows. Each site, upon receiving a voter v which specifies a linear order, instead of considering the voter v, creates and considers $\binom{m}{2}$ items, such that if v prefers c_1 to c_2, then we create an item (c_1, c_2) (notice that this is an ordered tuple). The reduction is depicted in Fig. 4. For example, a voter $v : a \succ b \succ d$ would be transformed into three items, (a, b), (a, d), and (b, d).

The reduced distributed stream has $n' = \binom{m}{2} \cdot n$ items and $O(m^2)$ types of items. For two candidates c_1 and c_2, let $N(c_1, c_2)$ denote the number of voters preferring c_1 to c_2. Now we can use a protocol based on counting frequencies (see Section 3.1), with $\epsilon' = \epsilon/m^2$, to let the center maintain, for each pair of candidates c_1 and c_2, a value $N'(c_1, c_2)$ such that $N'(c_1, c_2) \in N(c_1, c_2) \pm \epsilon' n' \subseteq N(c_1, c_2) \pm \epsilon n/2$.

Let $\mathrm{Sc}'(c, E)$ be the number of candidates c' such that $N'(c, c') \ge n/2 - \epsilon n/2$ in the election E. We denote by $\mathrm{Sc}(c, E)$ the (real) Copeland score of candidate c in election E. The center declares as an ϵ-winner a candidate $\hat{c}$ with the highest value of $\mathrm{Sc}'(\hat{c}, E)$. Note that, for every candidate c', it holds that $\mathrm{Sc}(c', E) \le \mathrm{Sc}'(c', E)$; this is so since the error in the computed frequency is bounded by $\epsilon n/2$, while for the declared winner $\hat{c}$, it holds that there are at least $\mathrm{Sc}'(\hat{c}, E)$ candidates c' such that $N'(\hat{c}, c') \ge n/2 - \epsilon n$.

$a \succ b \succ c$ $a \succ c \succ b$	$\Longrightarrow$	(a, b) (a, c) (b, c) (a, c) (a, b) (c, b)

Fig. 4. Reducing a linear order to frequencies.

Next we argue that $\hat{c}$ is indeed an ϵ-winner. We add $\epsilon n/2$ voters which rank $\hat{c}$ on top and then the other candidates in arbitrary order, and another $\epsilon n/2$ voters which rank $\hat{c}$ on top and then the other candidates in reverse order. Denote the modified election, with these additional voters, by E'. Then, for every c', $N(\hat{c}, c')$ increased by ϵn; thus, $\mathrm{Sc}(\hat{c}, E') \geq \mathrm{Sc}'(\hat{c}, E)$. Moreover, the number of wins of any other candidate c' does not increase. Hence $\mathrm{Sc}(c', E') \leq \mathrm{Sc}'(c', E) \leq \mathrm{Sc}'(\hat{c}, E) \leq \mathrm{Sc}(\hat{c}, E')$.

The communication complexity follows by the discussion given in Section 3.1; specifically, it is $O((\epsilon'^{-1}\sqrt{k} + k)\log n' \cdot \log m) = O((\frac{m^2}{\epsilon}\sqrt{k} + k)\log(nm) \cdot \log m)$.

The second protocol is based on checkpoints, and thus below we describe the static subprotocol used in each checkpoint. For every pair of candidates, c_1 and c_2, every site sends, to the center, the number of voters preferring c_1 over c_2, rounded to the closest multiplication of $\epsilon n/2k$. In each checkpoint, a candidate achieving estimated score higher that $\frac{n}{2} - \frac{\epsilon n}{2}$ for the maximal number of times (that is, for the largest number of other candidates) is declared a winner. As the error in each head-to head contest is upper-bounded by $k \cdot \frac{\epsilon n}{2k} = \frac{\epsilon n}{2}$, correctness follows by similar lines as given above in the proof of the frequency-count protocol. As there are m^2 quantities to estimate, each site sends $O\left(1 + \frac{m^2 \log \frac{k}{\epsilon}}{\log n}\right)$ words. The total communication follows.

For the third protocol, we will show that $s = O(\epsilon^{-2} \log \frac{m}{\delta})$ sampled voters, chosen uniformly at random (with repetitions), are enough to determine an ϵ-winner with failure probability at most δ. As we can communicate each voter using $\log(m!)$ bits, the bound follows. For two candidates c, c', let $X_i^{(c,c')}$ be an indicator for the event that the i's sampled voter prefers c over c'. Let $N'(c, c') = \frac{n}{s}\sum_{i=1}^{s} X_i^{(c,c')}$, and denote by $N(c, c')$ the actual number of voters preferring c over c' in the original election. Set $\mu = \mathbb{E}\left[\frac{1}{s}\sum_i X_i^{(c,c')}\right] = \frac{1}{n}N(c, c')$. Using Lemma 1 it follows that

$$\Pr\left[\left|N'(c, c') - N(c, c')\right| \geq \frac{\epsilon}{2} \cdot n\right] = \Pr\left[\left|\frac{1}{s}\sum_i X_i^{(c,c')} - \mu\right| \geq \frac{\epsilon}{2}\right] \leq \frac{\delta}{m^2}.$$

By union bound, with probability at least $1 - \delta$, for every pair of candidates we have that

$$\left|N'(c, c') - N(c, c')\right| < \frac{\epsilon}{2} \cdot n.$$

Let $\mathrm{Sc}'(c, E)$ be the number of candidates c' such that $N'(c, c') \geq n/2 - \epsilon n/2$ in the election E. The center declares as an ϵ-winner a candidate c with the highest value of $\mathrm{Sc}'(c, E)$. The accuracy of the protocol follows from arguments given in the analysis of the frequency-count protocol. □

We go on to consider the Cup rule, which differs from Copeland in several aspects. The first aspect is that, in order to prove that some estimated candidate c is indeed an ϵ-winner, it is not enough to add c arbitrary voters ranking c last, but rather a more subtle construction of voters is needed. The second aspect is that, intuitively, while in Copeland we had to send communication regarding all pairs of candidates, in Cup it is enough to send communication only regarding some pairs of candidates, as given by the binary tree corresponding to the "head-to-head" contests performed for finding the winner under Cup.

Theorem 7. *There are three protocols for* Cup. *Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{k}m^2 + k)\log mn \cdot \log m)$, $O\left(\frac{k}{\epsilon}(m \log \frac{k}{\epsilon} + \log m \cdot \log n)\right)$, *and* $O((\epsilon^{-2}\log m + k)(m \log m + \log n))$ *words.*

Proof. Let T be an implementation of the binary tree of the Cup election: There are $n - 1$ ordered pairs P of candidates (corresponding to the head-to-head "contests"), such that the winning candidate in each such pair goes up in the tree. In particular, every election E which agrees with the tree T on P, will have the root of T as its Cup-winner. We argue that there is an order π_P over the candidates such that, if $(c, c') \in P$, then c will appear before c' in π_P. Indeed, consider a directed graph G with the candidates as its vertices and P as its edges. G is acyclic and thus a topological order of G will provide us with the desired order π. Later we will use this order π as a preference order. Now we will proceed to describing the protocols.

Our first protocol is based on counting frequencies, and is similar to the corresponding Copeland protocol. We estimate the frequencies of *all* head-to-head contests (using the same precision and communication). To return a winner, we simply

run a Cup tournament (with the appropriate, given tree), using the estimations $N'(c, c')$ instead of the real values $N_E(c, c')$. As a result, we have a set P of $n-1$ ordered pairs. To prove correctness, it will be enough to show that by adding additional ϵn votes it will hold, for every $(c, c') \in P$, that $N_{E'}(c, c') \geq N_{E'}(c', c)$. Indeed, following the analysis of the frequency count of Copeland, with high probability for every pair of candidates c, c' we have that $|N'(c, c') - N_E(c, c')| \leq \epsilon n/2$. Recall the order π_P described at the beginning of the proof, and notice that by adding ϵn voters with preference orders as π_P it will hold, for every $(c, c') \in P$, that

$$N_{E'}(c, c') = N_E(c, c') + \epsilon n \geq N_E(c', c) = N_{E'}(c', c) ,$$

as required.

The second protocol is based on checkpoints,[10] thus below we describe the static subprotocol carried-out in each checkpoint. The subprotocol has $\log m$ rounds, corresponding to the height of the binary tree associated with the Cup protocol. In each round, the center asks each site to provide approximate values of the pairs currently at interest. Supplied with these approximate values, the center then computes the winner of each head-to-head contest, and continue to the nodes further up the tree. At the end, the center declares the winner of the highest node in the tree.

More concretely, for every pair of candidates of interest c, c', each site sends the center the number of voters preferring c over c', rounded to the closest multiplication of $\epsilon n/2k$. As the error in each head-to-head contest is upper-bounded by $k \cdot \frac{\epsilon n}{2k} = \frac{\epsilon n}{2}$, correctness follows by similar lines as given above in the proof of the frequency-count protocol described above. There are $\log m$ rounds, where at round i, each site sends $2^{\log m - i}$ values, each requiring $\log \frac{2k}{\epsilon}$ bits. Thus, the total number of words in a checkpoint is:

$$k \cdot \sum_{i=1}^{\log m} \left\lceil \frac{2^{\log m - i} \cdot \log \frac{2k}{\epsilon}}{\log n} \right\rceil \leq k \cdot \sum_{i=1}^{\log m} \left(1 + \frac{2^i \cdot \log \frac{2k}{\epsilon}}{\log n}\right) = O\left(k \cdot \left(\log m + \frac{m \cdot \log \frac{k}{\epsilon}}{\log n}\right)\right),$$

and total communication follows.

The third protocol is based on sampling and is similar to the Copeland sampling protocol. We use the same communication, and hence we ensure that with high probability, for every pair of candidates c, c' it holds that $|N'(c, c') - N(c, c')| < \frac{\epsilon}{2} \cdot n$. Correctness now follows by similar lines as in the frequency-count protocol. □

Finally, we consider the Condorcet voting rule. In order to declare a candidate c as a Condorcet ϵ-winner, it is enough to ensure that, by adding ϵn voters, every other candidate $c' \neq c$ loses to at least one other candidate in the head-to-head contest (and thus, either c can become a Condorcet winner in this way, or there will be no Condorcet winner at all, in which case c can be returned). A candidate c which is either Copeland or Cup ϵ-winner has this property. We conclude that every protocol for Copeland as well as every protocol for Cup is in particular a protocol for Condorcet.

Corollary 2. *There are three protocols for* Condorcet-winner-tracking. *Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{k}m^2 + k)\log mn \cdot \log m)$, $O\left(\frac{k}{\epsilon}(m \log \frac{k}{\epsilon} + \log m \cdot \log n)\right)$, *and* $O((\epsilon^{-2}\log m + k)(m \log m + \log n))$ *words.*

4.3. Round-based rules

In this section we consider two round-based voting rules; we begin with Plurality with run-off and then continue to Bucklin. For Plurality with run-off we provide three protocols, one of which is a "hybrid" protocol, specifically combining checkpoints and sampling. Intuitively, hybrid protocols fit naturally with round-based voting rules, which, informally speaking, are themselves "hybrids" of voting rules.

Theorem 8. *There are three protocols for* Plurality-with-run-off-winner-tracking. *Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{k}m^2 + k)\log mn \cdot \log m)$, $O\left(k\epsilon^{-1}\log n\right)$ *and* $O((\epsilon^{-2} + k)(m \log m + \log n))$ *words.*

Proof. The first protocol is based on counting frequencies. We combine the protocol for Plurality, described in the proof of Theorem 2, with the protocol for Copeland, described in the proof of Theorem 6. Specifically, the Plurality protocol maintains a frequency count for the plurality score of each candidate with accuracy $\frac{\epsilon}{6}$. The Condorcet protocol, for every two candidates c_1, c_2, maintains a frequency count for the number of times c_1 wins c_2 with accuracy $\frac{\epsilon}{3}$. Following the analysis in Theorem 2 and Theorem 6, the communication needed is $O((\epsilon^{-1}\sqrt{k} + k)\log mn \cdot \log m) + O((\epsilon^{-1}\sqrt{k}m^2 + k)\log mn \cdot \log m) = O((\epsilon^{-1}\sqrt{k}m^2 + k)\log mn \cdot \log m)$. When calculating the winner, the center identifies two candidates c_1, c_2 with the highest

[10] The protocol described here is useful if we assume that $\log m \cdot \log n \geq m^2 \log \frac{k}{\epsilon}$. If this is not the case, then we can use instead the communication protocol of Copeland.

estimated Plurality scores $f'(c)$ using the protocol for Plurality. Denoting by $f(c)$ the real Plurality score, for every $c' \neq c_1, c_2$ it holds that

$$\text{For } i \in \{1,2\}, \quad f(c') \leq f'(c') + \frac{\epsilon}{6}n \leq f'(c_i) + \frac{\epsilon}{6}n \leq f(c_i) + \frac{\epsilon}{3}n\,. \tag{1}$$

Next, the center uses the protocol for Condorcet to decide which of these two candidates it shall declare as an ϵ-winner. Assume, without loss of generality, that it declares c_1 as the winner. Then, by adding $\frac{2}{3}\epsilon n$ (resp. $\frac{1}{3}\epsilon n$) voters ranking c_1 (resp. c_2) on top, we can guarantee that c_1 and c_2 indeed have the highest Plurality score while c_1 wins c_2 in the head-to-head contest.

The second protocol is a "hybrid" protocol which combines checkpoints and frequency count. During the protocol we maintain estimated frequencies of the Plurality score of each candidate as in the first protocol (which we execute with precision $\frac{\epsilon}{6}$). Next we describe the subprotocol executed in each checkpoint. At each checkpoint, we use the Plurality protocol to identify two candidates c_1 and c_2 with the highest (approximated) Plurality score. Given c_1 and c_2, the center collects from all sites the *exact* number of voters preferring c_1 over c_2, and declares as a winner the one which is preferred by more voters. Correctness follows as by adding $\frac{\epsilon}{2}n$ voters ranking c_1 on top, and $\frac{\epsilon}{2}n$ voters ranking c_2 on top, we guarantee that c_1 and c_2 indeed have the highest plurality score (formally, this follows from equation 1) while the winner between the two remains unchanged. The subprotocol uses $2k$ words of communication, thus the total communication in all the checkpoints is $O\left(k\epsilon^{-1}\log n\right)$. For the frequency count we will use the deterministic protocol with $O\left(k\epsilon^{-1}\log n\right)$ communication. Therefore, in total we have a deterministic protocol with $O\left(k\epsilon^{-1}\log n\right)$ communication.

For the third protocol, we will show that $s = O(\epsilon^{-2}\log\frac{1}{\delta})$ sampled voters, chosen uniformly at random (with repetitions), are enough to determine an ϵ-winner with failure probability at most δ. We will use two sets of independent samples, S_1 and S_2, each of size $s/2 = O(\epsilon^{-2}\log\frac{1}{\delta})$. According to the proof of Theorem 3 for the case of $t = 1$, the set of sampled voters S_1 is sufficient for us to determine the plurality score of each candidate with accuracy $\frac{\epsilon}{6}$. Let c_1 and c_2 by the two candidates with the highest plurality score in S_1. Next we use S_2 to determine the number of times c_1 wins c_2 (with accuracy $\frac{\epsilon}{3}$ as in our protocol for Copeland; see Theorem 6), and return the candidate who wins in the head-to-head contest (in S_2). Correctness follows by similar lines to our frequency-count protocol. □

For Bucklin, we suggest three protocols; one is based on counting frequencies, the second is based on checkpoints, while the third is a sampling-based protocol.

Theorem 9. *There are three protocols for* Bucklin-winner-tracking. *Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{k}m\log^2 m + k)\log mn\cdot\log m)$, $O\left(\epsilon^{-1}\cdot k\cdot\log m\cdot\left(\log n + m\log\frac{k}{\epsilon}\right)\right)$, *and* $O((\epsilon^{-2}\log m + k)(m\log m + \log n))$ *words.*

Proof. To make the proof ideas more clear, for simplicity we assume that m is a power of 2 (we mention that if this is not the case, than we can add less than m dummy candidates, that always will be ranked after the real candidates). By the pigeonhole principle, a Bucklin winner is necessarily found within the first $m/2$ rounds. This is so, since a candidate which is not a Bucklin winner appears less than $n/2$ times within these $n\cdot m/2$ positions, thus if no candidate is a Bucklin winner, then at least one position in the $n\cdot m/2$ positions, constituting the $m/2$ first positions of the n voters, is not filled by any candidate.

We start with a discussion regarding the impact of adding voters. Let c be an arbitrary candidate and consider adding two voters, each ranking c on top, and ranking the other candidates in reverse orders. As a result, the score of c increases by 2 for each level $j \leq m/2$, while the status of each candidate $c' \neq c$ is only weaker (thus, if c' does not have a majority at level j before the addition, then it will also not have a majority after the addition).

The first protocol is based on counting frequencies. It begins by reducing the distributed vote stream into a different distributed stream. Intuitively, the idea is to consider binary divisions of the positions between 1 to m; for example, if we know the frequency of some candidate c in the first half positions (between position 1 and position $m/2$) as well as the frequency of it in the positions between position $m/2$ and position $3m/4$, then we know its frequency between position 1 and $3m/4$. Thus, we will have a different distributed stream for each binary division of the m positions, and we will use these to know the (approximated) Bucklin score of each candidate.

Formally, we will be estimating the frequencies of items of the form (c, i, j), where c is a candidate, $i \in [0, \log m]$, and $j \in [0, m/2^i - 1]$. To this end, each site, upon receiving a voter v, increases (by 1; this is equivalent to producing those entries and feeding them to the frequency count protocol) various entries in the frequency count protocol. Specifically, for each candidate c that was ranked at position $\ell \in [m]$, the site increases the items $(c, i, \left\lfloor\frac{\ell-1}{2^i}\right\rfloor)$, for all $i \in [0, \log m - 1]$ (alternatively (c, i, j) for $j \in [0, m/2^i - 1]$, such that $(j\cdot 2^i + 1) \leq \ell \leq ((j+1)\cdot 2^i)$). The idea is that we can recover the approximate number of voters ranking each candidate c at the first j positions using $\log m$ approximate counters of these items. See Fig. 5 for an illustrating example.

The protocol initiates a Frequency-tracking protocol on the reduced distributed stream with $\epsilon' = \epsilon/(2m\log^2 m)$. (To arrive to the term above, notice that one log factor comes from the fact that each voter is replaced by $m\log m$ votes, while the second log factor comes to compensate for the fact that we estimate up to $\log m$ values.) This will give us

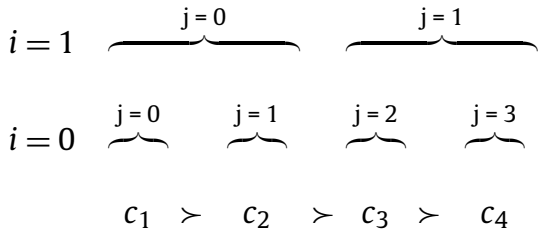

Fig. 5. An example for the reduction performed in the protocol for Bucklin. Specifically, a voter $v : c_1 \succ c_2 \succ c_3 \succ c_4$ is considered, which is reduced to the following stream elements: $(c_1, 0, 0)$, $(c_1, 1, 0)$, $(c_2, 0, 1)$, $(c_2, 1, 0)$, $(c_3, 0, 2)$, $(c_3, 1, 1)$, $(c_4, 0, 3)$, $(c_4, 1, 1)$.

Algorithm 5: Frequencies based protocol for Bucklin-winner-tracking, for center.

Initiate and maintain a Frequency-tracking protocol with precision parameter $\epsilon' = \epsilon/(2m \log^2 m)$, where the items are all triples (c, i, j) such that c is a candidate, $i \in [0, \log m]$ and $j \in [0, m/2^i - 1]$. The estimated frequency count of (c, i, j) is denoted $f'(c, i, j)$;
foreach *candidate* c *and* $k \in \{1, \ldots, m\}$ **do**
  Let $a_0, a_1, \ldots, a_{\log m - 1} \in \{0, 1\}$ such that $k = \sum_{i=0}^{\log m - 1} a_i \cdot 2^i$;
  Maintain $N'(c, k) = \sum_{i=0}^{\log m - 1} a_i \cdot f'\left(c, i, \left\lfloor \frac{k-1}{2^i} \right\rfloor\right)$;
Upon query: Find the minimum k for which there is at least one candidate c for which $N'(c, k) \geq \frac{n}{2} - \frac{\epsilon n}{2}$, and return the corresponding c;

Algorithm 6: Frequencies based protocol for Bucklin-winner-tracking, for site.

Participate in the Frequency-tracking protocol initiated by the center;
Upon receiving a vote $v : c_1 \succ \ldots \succ c_m$ reduce it to the following items:
for $\ell \in \{1 \ldots, m\}$ **do**
  for $i \in \{0, \ldots, \log m - 1\}$ **do**
    Simulate insertion of item $\left(c_\ell, i, \left\lfloor \frac{\ell-1}{2^i} \right\rfloor\right)$ to the Frequency-tracking protocol;

approximate values on the number of items of each type in our reduced distributed stream. Let us denote, for a candidate c and position $k \in [m]$, the number of voters ranking c at any position $k' \leq k$ by $N(c, k)$. Then, we can approximate each of the values $N(c, k)$ by adding $\log m$ different approximated frequencies, computed by the Frequency-tracking protocol (on the reduced stream). This is, informally, the reason why we reduced each original voter in to the items we reduced to: Given those items, it is enough to add $\log m$ different approximated frequencies in order to approximate the value of $N(c, k)$; then, as we will see below, bounding the error can be done in a finer way, since the error is accumulated only in $\log m$ different frequencies, and not in m such (which would be the case otherwise).

Formally, for candidate c, index $i \in [m]$, and $j \in \{0, \ldots, \frac{m}{2^i} - 1\}$, denote by $f(c, i, j)$ the number of voters that ranked c between the $(j \cdot 2^i + 1)$'th position and the $((j + 1) \cdot 2^i)$'th position. For an index $k \in [m]$, let $a_0, a_1, \ldots, a_{\log m - 1} \in \{0, 1\}$ such that $k = \sum_{i=0}^{\log m - 1} a_i \cdot 2^i$. It holds that $N(c, k) = \sum_{i=0}^{\log m - 1} a_i \cdot f\left(c, i, \left\lfloor \frac{k-1}{2^i} \right\rfloor\right)$. Denote by $f'(c, i, j)$ the frequency count protocol estimation for $f(c, i, j)$. Set $N'(c, k) = \sum_{i=0}^{\log m - 1} a_i \cdot f'\left(c, i, \left\lfloor \frac{k-1}{2^i} \right\rfloor\right)$ to be the approximations of $N(c, k)$.

We are now able to simulate Bucklin; specifically, the center finds the minimum k for which there is at least one candidate c for which $N'(c, k) \geq \frac{n}{2} - \frac{\epsilon n}{2}$, and declares c as an ϵ-winner.

Next we show correctness. The size of the reduced distributed stream is $n' = nm \log m$, since each voter is transformed into $m \log m$ items, specifically $\log m$ per each candidate. To approximate the value $N(c, k)$ we add up $\log m$ approximate frequencies, each of which can be wrong by at most $\epsilon' n' = \epsilon n/(2 \log m)$; thus, the value of $N'(c, k)$ can be wrong by at most $\epsilon n/2$. Therefore, in each level $j' < j$ where we do not find a winner, there is indeed no candidate with a majority. Finally, according to the discussion in the begging of the proof, ϵn additional voters can indeed make our chosen candidate a winner. A pseudocode of the protocol is given in Algorithms 5, 6.

The second protocol is based on checkpoints, and thus below we describe the static subprotocol carried-out in each checkpoint. Each checkpoint contains $\log m$ rounds, where in each of these $\log m$ rounds, the center is performing an approximate binary search to find the first j for which there is at least one candidate c_i for which the estimation of $N(c_i, j)$ is greater than $\frac{n}{2} - \frac{\epsilon n}{8}$, and declares this c_i as an ϵ-winner. In the round when some index j is considered, each site sends to the center the number of voters ranking each candidate c among the first j positions, rounded to the closest multiplication of $\epsilon n/4k$. Thus, the center can estimate each $N(c_i, j)$ with precision $\frac{\epsilon n}{8}$, as needed.

Let c be our declared candidate, which is declared at round j. Then, according to the discussion in the beginning of the proof, by adding $\frac{\epsilon}{4}$ votes, the declared candidate c will have majority of the votes at round j, while no $c' \neq c$ will have majority of the votes for $j' < j$. In particular the candidate c is an $\frac{\epsilon}{4}$-winner. Correctness follows by the discussion in

Section 3.2. As at most $k\lceil \frac{m \log 4k/\epsilon}{\log n} \rceil$ words of communication are required in each round of the sub-protocol, and there are at most $\log m$ rounds, the total communication is bounded by

$$O\left(\frac{\log n}{\epsilon} \cdot \log m \cdot k\lceil \frac{m \log 4k/\epsilon}{\log n} \rceil\right) \leq O\left(\frac{k \cdot \log m}{\epsilon} \cdot \left(\log n + m \log \frac{k}{\epsilon}\right)\right) .$$

For the third protocol, we will show that $s = O(\epsilon^{-2} \log \frac{m}{\delta})$ sampled voters, chosen uniformly at random (with repetitions), are enough to determine an ϵ-winner with failure probability at most δ. As we can communicate each voter using $\log(m!)$ bits, the bound would follow. So, for each candidate c and $j \in [m]$, let $X_i^{(c,j)}$ be an indicator for the event that the i's sampled voter ranks c among the top j positions. Set $N'(c, j) = \frac{n}{s} \sum_{i=1}^{s} X_i^{(c,j)}$ to be an estimation for $N(c, j)$ – the number of voters ranking c at among the top j positions. Using Lemma 1 we conclude that $\Pr\left[\left|N'(c, j) - N(c, j)\right| \geq \frac{\epsilon}{2} \cdot n\right] \leq \frac{\delta}{m^2}$. By union bound, with probability at least $1 - \delta$ for every all c, j, it holds that $\left|N'(c, j) - N(c, j)\right| < \frac{\epsilon}{2} \cdot n$. The center now finds the first j for which there is at least one candidate c for which $N'(c, j) \geq \frac{n}{2} - \frac{\epsilon n}{2}$, and declares this c as an ϵ-winner. Correctness follows by the same arguments as in the frequency count protocol. □

5. Lower bounds

In this section we provide lower bounds to complement the upper bounds derived above. Our main result is an almost tight lower bound (up to a factor of $\log k \cdot \log m$) for Plurality-winner-tracking. We mention that our lower bound holds already for Plurality with 2 candidates and that it also improves the state-of-the-art lower bound for Count-tracking (refer to Theorem 10 for our lower bound and to the remark which follows it for its application to Count-tracking). Later in this section we describe a lower bound for deterministic protocols for Approval-winner-tracking, which is of some interest mainly since it is almost tight for Approval-winner-tracking and also shows that some dependency on the number m of candidates is required.

5.1. Randomized lower bound for Plurality-winner-tracking

Before we describe the randomized lower bound for Plurality-winner-tracking, we mention that it is applicable to all other voting rules we consider, via the following reduction.

Lemma 3. *Let $\mathcal{R}$ be some voting rule described in Section 2.1. A protocol for $\mathcal{R}$-winner-tracking which uses C words of communication implies a protocol for Plurality with 2 candidates which uses C words of communication.*

Proof. Assuming a protocol for a voting rule $\mathcal{R}$, we can use it as a black-box for solving Plurality with 2 candidates; below we describe such a reduction.

Let $\mathcal{R}$ be a voting rule considered in this paper. Let P be a protocol for $\mathcal{R}$ which uses C words of communication. We construct a protocol P' for Plurality with two candidates, a and b, which uses P as a black-box. Specifically, we describe the operation of P' for the different $\mathcal{R}$'s considered in this paper; the general idea of the reduction is similar for all these voting rules, namely, given a Plurality election to construct a $\mathcal{R}$ election where the $\mathcal{R}$ winners are equivalent to the Plurality winners. The specifics of the reduction very slightly between the voting rules considered. We denote by a and b our two Plurality candidates.

If $\mathcal{R}$ is Approval, then, for each Plurality voter which arrives and approves some candidate, say a, we create a voter approving only a. Notice that the Approval winners are equivalent to the Plurality winners.

If $\mathcal{R}$ is one of {Borda, Condorcet, Copeland, Cup, Plurality with run-off, Bucklin}, then, for each Plurality voter which arrives and approves a, we create a voter ranking a on top and then b; similarly, for each Plurality voter which arrives and approves b, we create a voter ranking b on top and then a. Notice that, in these cases, the $\mathcal{R}$ winner are equivalent to the Plurality winners.

If $\mathcal{R}$ is t-Approval, then we shall create $2t - 2$ new candidates $c_1, \ldots, c_{2t-2}$, and, for each Plurality voter which arrives and approves a, we create one voter ranking $a, c_1, \ldots, c_{t-1}$ on top, and another voter ranking $a, b, c_t, \ldots, c_{2t-3}$ on top; similarly, for each Plurality voter which arrives and approves b, we create one voter ranking $b, c_1, \ldots, c_{t-1}$ on top, and another voter ranking $b, a, c_t, \ldots, c_{2t-2}$ on top; Notice that in this case also, the $\mathcal{R}$ winner are equivalent to the Plurality winners. □

We mention that in our lower bound for Plurality which we describe next, we assume, as it is usual in studying distributed streams, that there is no spontaneous communication; that is, the center can initiate communication only as a result of receiving a message from the sites, and each site can initiate communication only as a result of receiving a stream item or a message from the center.

Now we are ready to state our lower bound for Plurality; the proof of the corresponding theorem (that is, Theorem 10) appears at the end of the section, and is based on Lemma 4 and Lemma 5. Recall that for Plurality-winner-tracking, Theorem 2 provides an upper bound of $O((\epsilon^{-1}\sqrt{k} + k) \log n \cdot \log m)$.

Theorem 10. *Any randomized protocol for* Plurality-winner-tracking *uses at least* $\Omega((\epsilon^{-1}\sqrt{k}+k)\log n/\log k)$ *words of communication, even when there are only two candidates.*

The next lemma shows a lower bound when $k < \epsilon^{-2}$.

Lemma 4. *If* $k < \epsilon^{-2}$*, then any randomized protocol for* Plurality-winner-tracking *uses at least* $\Omega(\epsilon^{-1}\sqrt{k}\log n)$ *words of communication, even when there are only two candidates.*

Proof. We reduce Count-tracking to Plurality-winner-tracking. To this end, we assume, towards a contradiction, that there is a protocol for Plurality-winner-tracking with $o(\epsilon^{-1}\sqrt{k}\log n)$ communication complexity, and describe a protocol with the same communication complexity for Count-tracking. For $k < \epsilon^{-2}$ this leads to a contradiction, since there is a lower bound of $\Omega(\epsilon^{-1}\sqrt{k}\log n)$ for Count-tracking where $k < \epsilon^{-2}$ [22, Theorem 2.4].

The distributed stream for Count-tracking contains items of only one type, and a protocol for Count-tracking maintains a value n' such that $n' \in n \pm \epsilon n$, where n is the number of items in the distributed stream. We treat those items as voters, each of which is approving the candidate c_1.

So, our reduction would work as follows: We assume the existence of a protocol for Plurality-winner-tracking in order to design a protocol for Count-tracking. Specifically, in order to design a protocol for Count-tracking with k sites, we will use (the assumed) protocol for Plurality-winner-tracking with $k+1$ sites. The idea would be that the k sites for the Count-tracking protocol would be mapped to k sites of the Plurality-winner-tracking protocol, where an item received by a site of the Count-tracking protocol would be mapped to a vote for candidate c_1, delivered to a site of the Plurality-winner-tracking. Then, we center of the protocol for Count-tracking which we are designing would directly send votes for candidate c_2 to the remaining site, at certain times. This would cause the protocol for Plurality-winner-tracking with $k+1$ sites to swap winners back and forth from declaring c_1 to be a winner to declaring c_2 to be a winner; intercepting those flips, we would be able to approximate the number of message, thus indeed describe a protocol for Count-tracking.

Below we explain how does the center operate in our protocol for Count-tracking. The general idea of the reduction is as follows: For the center to simulate another site, called a *ghost site* (we use this name to emphasize that it is not a "real" site, but just a "virtual" site which is being simulated by the center, as part of the center's internal computation), to which the center will send *ghost voters* (again, not real voters, but only simulated by the center). The center will simulate a protocol for Plurality with voters approving c_1 going to the k "real" sites, and simulated voters approving c_2 going to the ghost site.

Specifically, the center has three parts (here, by parts we mean the implementation modules of the center; that is, the center is a center for Count-tracking which internally uses the following parts/modules in its internal computation). The first part is a center for a Plurality-winner-tracking protocol operating on $k+1$ sites. The second part is a site in a Plurality-winner-tracking protocol; this is the ghost site. The third part is for the center to inspect the Plurality-winner-tracking protocol from above, and (using knowledge about the current winner) to send voters approving the candidate c_2 to the ghost site.

Let us denote the number of voters voting for c_1 (c_2) by $s(c_1)$ (respectively, $s(c_2)$). Set $\delta = \epsilon/10$. The protocol for Plurality-winner-tracking will work with respect to approximation δ, and will consist of $k+1$ sites. Next we describe the logic of the third part of the center. The estimation for Count-tracking will be $\text{est} = (1+3\delta)s(c_2)$ (note that only the ghost site receives voters approving c_2, hence the center knows $s(c_2)$ exactly).

Before there is any communication from the (real) sites to the center, we set $s(c_2) = 0$. Then, at some point in time there will be some communication from the sites to the center indicating that some voters approving c_1 arrived; specifically, the first part of the center would declare c_1 as the winner of the election. More generally, our protocol works in phases, where a phase starts when the center "flip"s its estimation; that is, the (first part of the) center changes the estimation for the Plurality winner from c_2 to c_1. When such a flip occurs, the center sends some ghost voters (approving c_2) to the ghost site until $s(c_2) = (1+3\delta)^i$ (for some i) and a flip (from c_1 back to c_2) occurs. (That is, we send ghost voters until a flip occurs and then send some additional voters until we reach a power of $1+3\delta$; reaching this power of $1+3\delta$ is actually not needed, but it does not affect the communication complexity and it makes the analysis cleaner.) We assume, as is usually done in distributed streams, that communication and internal computation happens instantly. Thus, we have that c_2 is always the winner of the Plurality-winner-tracking protocol. This finishes the description of the reduction.

Next we argue that our estimation (for Count-tracking) is accurate. Specifically, we will show that $s(c_1) \le \text{est} \le (1+\epsilon)s(c_1)$. As c_2 is always the winner, it always holds that $s(c_2) + \delta(s(c_1)+s(c_2)) \ge s(c_1)$. Since $\delta < 1/10$ (as $\epsilon < 1$), it holds that:

$$s(c_1) \le \frac{1+\delta}{1-\delta} \cdot s(c_2) < (1+3\delta) \cdot s(c_2) = \text{est}\,.$$

Fix $s(c_2) = (1+3\delta)^i$. Note that when $s(c_2)$ was equal to $(1+3\delta)^{i-1}$, the protocol for Plurality-winner-tracking considered c_1 as the winner. Hence, $s(c_1) + \delta(s(c_1) + (1+3\delta)^{i-1}) \ge (1+3\delta)^{i-1}$; therefore,

$$s(c_1) \ge \frac{1-\delta}{1+\delta}(1+3\delta)^{i-1} \ge \frac{(1+3\delta)}{(1+3\delta)^3}(1+3\delta)^i \ge \frac{\text{est}}{1+\epsilon}\,.$$

Note that, until the next flip, $s(c_1)$ can only grow, while our estimation remains unchanged. Hence, it will still hold that $\text{est} \le (1+\epsilon)s(c_1)$. Finally, we have that the communication of our protocol is bounded by $o(\delta^{-1}\sqrt{k+1}\log(s(c_1)+s(c_2))) = o(\epsilon^{-1}\sqrt{k}\log n)$, which contradicts the lower bound for Count-tracking discussed above. □

The next lemma is especially interesting for $k \ge \epsilon^{-2}$.

Lemma 5. *Any randomized protocol for* Plurality-winner-tracking *uses at least* $\Omega(k \log n / \log k)$ *words of communication, even when there are only two candidates.*

Proof. We assume that $\epsilon < \frac{1}{3}$. Consider a protocol for Plurality-winner-tracking which is correct with probability $\frac{2}{3}$ on every input. Next we describe a distributed stream of voters which come to the sites. Specifically, the stream consists of s phases. Let $x_1 = 1$, $y_1 = 1$, $x_i = (1+3\epsilon)\cdot k \cdot y_{i-1}$, and $y_i = y_{i-1} + x_i$. During the i'th phase, x_i voters will go to each site and vote for the candidate $c_{(i \bmod 2)}$. Note that after the i'th phase, exactly y_i voters voted at each site. The total number of votes for $c_{(i \bmod 2)}$ is at least $k \cdot x_i$, while the total number of votes for $c_{(i-1 \bmod 2)}$ is at most $k \cdot y_{i-1}$. In particular, $c_{(i \bmod 2)}$ is a unique ϵ-winner.

Note that

$$y_i = y_{i-1} + x_i = y_{i-1} + (1+3\epsilon)\cdot k \cdot y_{i-1} = (1+(1+3\epsilon)\cdot k)\cdot y_{i-1} = (1+(1+3\epsilon)\cdot k)^{i-1}\cdot y_1 \le (3k)^{i-1} \ ,$$

thus the total number of voters is bounded by $n = k \cdot y_s < (3k)^s$. In particular, $s = \Omega(\frac{\log n}{\log k})$.

Next consider the j's site S_j during the phase i. Let $Y_{i,j}$ be the event that some communication between the center and S_j occurs. Let $Z_{i,j}$ be the event that the center initiates communication with S_j. Let $X_{i,j}$ be the event that S_j initiates communication with the center, conditioned on the event that the center does not initiate communication with S_j (that is, $Y_{i,j}$ conditioned on $\overline{Z_{i,j}}$). We argue that $\mathbb{E}[X_{i,j}] = \Omega(1)$. Before the i'th phases starts, $c_{(i-1 \bmod 2)}$ is the unique ϵ-winner.

Consider an alternative scenario where, after the end of the $i-1$'th phase, x_i voters come to S_j (and vote for $c_{(i \bmod 2)}$), while no additional voters arrive. In this alternative scenario the center will not initiate communication with S_j, as from its point of view nothing have changed since the end of the $(i-1)$'s phase (since it did not receive any new messages). Note also that in the alternative scenario, $c_{(i \bmod 2)}$ is the unique ϵ-winner. This is since

$$\begin{aligned} k\cdot y_{i-1} + \epsilon(k\cdot y_{i-1} + x_i) &= k\cdot y_{i-1} + \epsilon(k\cdot y_{i-1} + (1+3\epsilon)\cdot k\cdot y_{i-1}) \\ &= k\cdot y_{i-1}\,(1+\epsilon(1+(1+3\epsilon))) \\ &= k\cdot y_{i-1}\left(1+2\epsilon+3\epsilon^2\right) < x_i. \end{aligned}$$

Thus, if S_j will not initiate communication with the center, then, in the alternative scenario, the center would not hold the right estimation both at the end of the $i-1$'th phase and at the end of the i'th phase. This is so since it will have the same estimation, while there are different unique ϵ-winners at those times. Therefore, the probability that the center is right in both of these times is bounded by $\Pr\left[X_{i,j}\right]$. As the center has constant probability to have the right estimation twice, we conclude that $\mathbb{E}[X_{i,j}] = \Omega(1)$.

Let us go back to our original scenario. Note that, from the point of view of S_j, both scenarios are the same (unless the center initiates communication). In particular $\Pr\left[X_{i,j} \mid \overline{Z_{i,j}}\right] = \Omega(1)$. Set $\Pr\left[Z_{i,j}\right] = \alpha$. Then, we have that:

$$\begin{aligned} \mathbb{E}\left[Y_{i,j}\right] = \Pr\left[Y_{i,j}\right] &= \Pr\left[Z_{i,j}\right] + \Pr\left[\overline{Z_{i,j}}\right]\Pr\left[X_{i,j} \mid \overline{Z_{i,j}}\right] \\ &= \alpha + (1-\alpha)\cdot\Omega(1) = \Omega(1)\ . \end{aligned}$$

The total communication used during the whole protocol can be lower bounded by $\sum_{i=1}^{s}\sum_{j=1}^{k}\mathbb{E}\left[Y_{i,j}\right] = \Omega(sk) = \Omega\left(\frac{k\log n}{\log k}\right)$. □

We are ready to prove Theorem 10.

Proof of Theorem 10. If $k < \epsilon^{-2}$, then Lemma 4 provides us with a lower bound of $\Omega(\frac{\sqrt{k}}{\epsilon}\log n) = \Omega\left((\frac{\sqrt{k}}{\epsilon}+k)\frac{\log n}{\log k}\right)$. Otherwise ($k \ge \epsilon^{-2}$), using Lemma 5 we get a lower bound of $\Omega(\frac{k\log n}{\log k}) = \Omega\left((\frac{\sqrt{k}}{\epsilon}+k)\frac{\log n}{\log k}\right)$. □

Remark 5. Notice that Lemma 5 implies a $\Omega(\frac{k\log n}{\log k})$ lower bound for the Count-tracking problem. The Count-tracking problem is a central problem in distributed streams, where the goal is to continuously maintain a counter which is at most ϵn far from the actual number of items arriving to the stream. For the Count-tracking problem in the regime where $k \ge \epsilon^{-2}$, Huang et al. [22, Theorem 2.3] give a lower bound of $\Omega(k)$.

Lemma 5 relates to Count-tracking, As there is a reduction from Plurality-winner-tracking with two candidates to Count-tracking: to implement a protocol for Plurality-winner-tracking with two candidates it is sufficient to use two protocols for Count-tracking with $\epsilon' = \epsilon/2$, one for each candidate, and to report as winner the candidate corresponding to the larger counter.

Thus, we conclude that Lemma 5 implies a $\Omega(\frac{k \log n}{\log k})$ lower bound for the Count-tracking problem, thus improving the state of the art for this problem.

5.2. Deterministic lower bound for Approval-winner-tracking

Next we prove a lower bound on the communication of a deterministic protocol for Approval-winner-tracking. Recall the checkpoints-based deterministic protocol described within the proof of Theorem 4: the protocol has $O(\epsilon^{-1} \log n)$ checkpoints, and in each checkpoint, each site sends $\log(\frac{4k}{\epsilon})$ bits per candidate. Thus, if we measure the communication in bits (instead of words as in Theorem 4), we get that the total cost of that protocol is $O\left(\epsilon^{-1} \log n \cdot m \cdot k \cdot \log(\frac{4k}{\epsilon})\right) = O\left(\frac{m \cdot k \cdot \log(\frac{k}{\epsilon})}{\epsilon} \cdot \log n\right)$. In this section we prove Theorem 11, showing that protocol (the one from Theorem 4) to be almost optimal in the deterministic regime.

Theorem 11. *For $\epsilon < 1/16$, and for large enough m, any deterministic protocol for* Approval-winner-tracking *uses at least $\Omega\left(\frac{mk}{\epsilon} \cdot \log\left(\frac{n}{k}\right)\right)$ bits of communication.*

The proof of Theorem 11 is based on a reduction from a new problem in *communication complexity*; specifically, the variant of communication complexity which is sometimes referred to as *multiparty communication complexity*. In this variant we have k players, denoted by $P_1, \ldots, P_k$, and each player P_j possesses a (possibly different) string $x_j \in \{0,1\}^m$. The objective is to compute the outcome of a function $f : \{0,1\}^{m \times k} \to \{0,1\}$ on the combined inputs of the players (formally, on the concatenation of the x_j strings). The players follow some protocol, and can communicate by broadcasting bits. Specifically, when a player broadcasts a bit b, all other players receive b and we add 1 to the communication count. The cost of a protocol is the maximum number of exchanged bits, over all possible inputs. The deterministic communication complexity of the function f, denoted by $D(f)$, is the minimal cost of a deterministic protocol that computes f. For additional details and overview of the field we refer to the textbook of Kushilevitz and Nisan [23] or to the book chapter by Razborov [28].

Next we define the *No Strict Majority* problem: NSM, in short. In it, we have $2k$ players and a parameter $\epsilon > 0$. Each player P_j has an m-bit string $A_j \in \{0,1\}^m$. The objective is to figure out if there is an index i such that a strict majority of the players has 1 in that index. Formally,

$$\mathsf{NSM}_{2k,m,\epsilon}(A_1, \ldots, A_{2k}) = \begin{cases} 0 & \exists i \ \left|\left\{j \mid i \in A_j\right\}\right| \geq (1+\epsilon)k \\ 1 & \forall i \ \left|\left\{j \mid i \in A_j\right\}\right| \leq k \\ \text{Don't Care} & \text{Otherwise,} \end{cases}$$

where by "Don't Care", we mean that any outcome of the protocol is legitimate. The role of the "Don't Care" here is to allow us to reduce approximation problems to NSM, as strict boundaries will not allow for that.

We denote a conjunction of l instances of $\mathsf{NSM}_{2k,m,\epsilon}$ by $\bigwedge_{i=1}^{l} \mathsf{NSM}_{2k,m,\epsilon}$. That is, we have $2k$ players, each of which is given l strings of m bits each (formally, P_j gets $A_{j,1}, \ldots, A_{j,l}$); the outcome shall be 1 if and only if, for every index $s \in [1,l]$ and $i \in [1,m]$, it holds that $\left|\left\{j \mid i \in A_{j,s}\right\}\right| \leq k$. An equivalent way to think about $\bigwedge_{i=1}^{l} \mathsf{NSM}_{2k,m,\epsilon}$ is that each of the players gets a binary $l \times m$ matrix and we accept if there is no cell for which a majority of the players has a 1 in.

The proof of the following lemma could be found in Appendix B. We mention that, as far as we know, the $\bigwedge_{i=1}^{l} \mathsf{NSM}_{2k,m,\epsilon}$ problem was not considered in the literature, hence the following lemma is novel and might be useful in other contexts besides our current context, that of communication-efficient protocols for monitoring election winners.

Lemma 6. $D\left(\bigwedge_{i=1}^{l} \mathsf{NSM}_{2k,m,\frac{1}{4}}\right) = \Omega(mkl)$.

To prove Theorem 11, next we show how the communication complexity of $\bigwedge_{i=1}^{l} \mathsf{NSM}_{2k,m,\frac{1}{4}}$ implies a lower bound on the communication of Approval-winner-tracking. The general idea, similarly to the idea underlying the lower bound described in Lemma 5, is to exploit the fact that, in any point in time, the center should be able to produce an answer without any additional communication. Specifically, we will have $l = \Omega(\frac{km}{\epsilon} \log \frac{n}{k})$ rounds, such that by sampling the center in l different points of time we can determine $\bigwedge_{i=1}^{l} \mathsf{NSM}_{2k,m,\frac{1}{4}}$.

Proof of Theorem 11. For the sake of simplicity, during the proof we will consider also non-integer number of voters; this issue can easily be fixed by proper rounding, while introducing only a constant overhead to the number of voters.

Consider an instance of $\bigwedge_{i=1}^{l} \mathsf{NSM}_{2k,m,\frac{1}{4}}$, where the input of player P_j is $\left\{A_j^s\right\}_{s=1}^{l} \in \{0,1\}^{m\times l}$. We will use a protocol for Approval-winner-tracking with m candidates, $2k$ sites, and precision parameter ϵ to solve $\bigwedge_{i=1}^{l} \mathsf{NSM}_{2k,m,\frac{1}{4}}$. By Lemma 6, $\bigwedge_{i=1}^{l} \mathsf{NSM}_{2k,m,\frac{1}{4}}$ requires $\Omega(lmk)$ communication. This in turn will imply a lower bound for the communication complexity of Approval-winner-tracking.

Our reduction is as follows. Each player acts as a site, and will simulate the arrival of voters in some order, to be specified shortly. Player P_1 will act also as server (this is possible as we assume broadcast communication). We denote the number of voters that approve candidate i at site j by $(v_i)_j$, and the total number of voters, across all sites, approving candidate i by $v_i = \sum_j (v_i)_j$. The reduction has several phases. We first describe the first phase and later generalize it to describe how the rth phase is executed.

- **First phase:** Before the first phase starts, the situation is that each candidate is approved by 0 voters at each site. The first phase has 3 stages, as follows.
  - Vote simulation: each site j simulates that a voter approving A_j^1 arrives.
  - Validation: the center computes a winner q, then it collects $(v_q)_j$ from all the sites (players). If $v_q = \sum_{j=1}^{2k} (v_q)_j > k$, then it determines that the solution for the first instance is 0. Otherwise it determines that the solution is 1.
  - Reset: each site j simulates that a voter approving $\overline{A_j^1}$ comes.
- ***r*th phase:** Set $x_r = (1+32\epsilon)^{r-2}$ and $y_r = 32\epsilon x_r > \frac{16\epsilon}{1-8\epsilon} x_r$. Before the rth phase starts, the situation is that each site already received exactly $2 \cdot x_r$ voters, such that each candidate was approved by exactly x_r voters at each site. The rth phase has 3 stages:
  - Vote simulation: each site j simulates that y_r voters appeared, all approving A_j^r.
  - Validation: the center computes a winner q, then it collects $(v_q)_j$ from all the sites (players). If $v_q = \sum_{j=1}^{2k} (v_q)_j > 2k \cdot x_r + k \cdot y_r$, then it determines that the solution for the rth phase is 0. Otherwise it determines that the solution is 1.
  - Reset: each site j simulates that y_r voters appeared, all approving $\overline{A_j^r}$.

In total, the number of voters used throughout the protocol is $n = 2k \cdot 2 \cdot x_{l+1} = 4k \cdot (1+32\epsilon)^{l-1}$. In addition to the protocol for Approval-winner-tracking, we also used $O(k)$ communication in each phase to compute the number of votes the winner got; to see why $O(k)$ bits of communication suffices for each phase, notice that in phase r, in the validation stage, each site sends to the center the number of voters voted for the q'th candidate. As there are only two options for this number $(x_r, x_r + y_r)$, one bit of communication suffices for each site, thus we have $O(k)$ additional bits of communication in total for each phase. Thus, the total communication our protocol uses, in addition to the Approval-winner-tracking protocol, is $O(lk)$.

Next we argue that we indeed compute the right answer for each of the l instances of $\bigwedge_{i=1}^{l} \mathsf{NSM}_{2k,m,\frac{1}{4}}$. Note that, at the time of the second step in the rth phase, exactly $2x_r + y_r$ voters arrived at each site, accumulating to a total of $n_r = 2k \cdot (2x_r + y_r)$ voters. Fix some $r \in [1,l]$ and consider first the case where there exists an index i such that $\left|\left\{j \mid i \in A_r^j\right\}\right| \geq \left(1+\frac{1}{4}\right)k$. In particular, the i'th candidate was approved by at least $v_i \geq 2k \cdot x_r + (1+\frac{1}{4}) \cdot k \cdot y_r$ voters. Hence, the Approval protocol will return an index q s.t. $v_q + \epsilon \cdot n_r \geq v_i$. As $y_r > \frac{16\epsilon}{1-8\epsilon} x_r$ it holds that $\frac{1}{4} \cdot k \cdot y_r > \epsilon \cdot n_r$, and in particular $v_q \geq v_i - \epsilon \cdot n_r > 2k \cdot x_r + k \cdot y_r$. We conclude that, in this case, the algorithm will compute the correct answer in the rth phase.

Otherwise, if for every index i, we have that $\left|\left\{j \mid i \in A_j^r\right\}\right| \leq k$, then no matter which index q the algorithm for Approval-winner-tracking will return, since the center will check it and will find out that $v_q \leq 2k \cdot x_r + k \cdot y_r$. Hence, again, it will compute the right answer.

Note that the number of voters used throughout the protocol is $n = 4k \cdot (1+32\epsilon)^{l-1}$, hence $l = 1 + \log_{1+32\epsilon} \frac{n}{4k} = \Omega\left(\frac{1}{\epsilon} \log \frac{n}{k}\right)$. As, other than the protocol for Approval-winner-tracking, we used only $O(lk)$ bits, while we solved $\bigwedge_{i=1}^{l} \mathsf{NSM}_{2k,m,\frac{1}{4}}$, a problem requiring $\Omega(mkl)$ bits, we conclude that Approval-winner-tracking requires at least

$$\Omega(lmk) - O(lk) = \Omega(lmk) = \Omega\left(\frac{mk}{\epsilon} \cdot \log\left(\frac{n}{k}\right)\right),$$

bits. In the first equality we used the fact that m is large enough. □

6. Discussion and outlook

In this paper we studied communication-efficient protocols for maintaining approximate winners in distributed vote streams. We have shown several general techniques for designing such protocols (namely, sampling-based protocols, protocols based on checkpoints, and protocols based on counting frequencies), and demonstrated their usefulness for various single winner voting rules. Indeed, based on these general techniques, for each of the rules we considered here, we have

designed several communication-efficient protocols, and analyzed their communication complexity. We complemented our protocols with lower bounds.

As a further contribution, we view our paper as a bridge between issues and ideas from artificial intelligence (specifically, multiagent systems and computational social choice) and techniques and methods from theoretical computer science and database systems (specifically, streaming and sampling algorithms and distributed continuous monitoring). We hope that more fruitful research can be done by bridging between those fields.

Below we first discuss several aspects which are somehow hidden in the technical part of the part. Specifically, we begin with a discussion on deterministic protocols, showing that, while the technical part of the paper concentrates on randomized protocols, communication-efficient deterministic protocols for monitoring election winners in distributed streams exist as well. Then, as in this paper we developed several protocols for each voting rule considered, we provide a discussion on how to choose which protocol to use at which scenario, depending on the specific parameters of the problem at hand. We end this section by mentioning some directions for future research.

6.1. Deterministic protocols

While in this paper we concentrated on randomized protocols, it turns out that some of our protocols are already deterministic or can be made deterministic with some slight modifications. To us, this is quite surprising: for example, there are usually no efficient deterministic algorithms operating on centralized streams. Specifically, as we show next, while there are no natural deterministic equivalents to our sampling-based protocols (since, informally speaking, a deterministic equivalent to sampling would basically need to sample the whole electorate), our other protocols can generally be made deterministic.

Indeed, protocols based on checkpoints are already deterministic. Further, protocols based on counting frequencies can use a deterministic protocol for Frequency count which uses $O(\epsilon^{-1}k\log n)$ words of communication [34]. Correspondingly, the increase in the communication complexity is by at most a factor of $\sqrt{k}$. Notice that the corresponding deterministic protocols still maintain only approximate solutions.

6.2. Choice of protocol

A closer look at our upper bounds reveals that the choice of which protocol to use for which voting rule crucially depends on the relationships between the various parameters; specifically, as a rule of thumb, it looks as if the choice of which protocol to use depends on the relation between k and $1/\epsilon^{-2}$; specifically, if $k < 1/\epsilon^{-2}$, then protocols based on counting frequencies or on checkpoints shall be used, while if $k \geq 1/\epsilon^{-2}$, then sampling-based protocols achieve better communication complexity. We believe that both cases make sense; for example, in a supermarket chain with 4000 stores, requiring approximation of $\epsilon = 1/100$ would put us in the first case, while requiring $\epsilon = 1/10$ would put us in the second case.

6.3. Future directions

Below, we discuss several directions for future research.

6.3.1. Improved bounds and more rules

While we considered quite a variety of voting rules in this paper, there are further interesting rules to consider, ranging from single-winner voting rules such as Kemeny, Young, Dodgson, Schulze, Maximin, and Ranked pairs, to multiwinner voting rules such as committee scoring rules, including Chamberlin–Courant and Monroe. Further, there are still some gaps between our upper bounds and lower bounds; closing those gaps is a natural direction for future research.

6.3.2. Simulations and heuristics

Our focus in the current paper, besides bridging between the study of computational social choice within the field of artificial intelligence and the topic of continuous distributed monitoring within database systems and theoretical computer science, is a theoretic study of communication-efficient protocols for maintaining election winners in distributed elections.

We believe that a theoretic study is important but also appreciate the possibility of validating our theoretical findings by performing simulations. Thus we view an experimental follow-up to the current paper as an important and interesting future work. One shall be careful in choosing input instances and evaluation methods, and there is also some hope that efficient heuristics (for which the theoretical complexity might not be impressive) outperform our protocols for certain scenarios and distributions.

6.3.3. Constrained resources

In this paper, we measured the complexity of our protocols only in terms of their communication cost. It is natural to consider other resources, especially studying various trade-offs between space, time, and communication. We mention that, for example, our sampling-based protocols do extend to situations where the computational power of the sites is very limited, since sampling from a distributed stream can be done with sites which have only logarithmic space [10]. Our checkpoint-based protocols, however, generally assume linear space (in m) for each site.

6.3.4. Various restrictions

In this paper we have concentrated on worst-case notions: first, we assumed that voters are arbitrarily (thus, adversarially) assigned into the sites; second, we did not assume any structure on the electorate itself. Since there might be better real-world situations, it is natural to study protocols for elections drawn from, say, Mallow's model or the Urn model, as well as to study situations where the voters are, say, uniformly assigned into the sites. Of course, studying protocols for elections which adhere to some domain restrictions, such as single peaked elections and single crossing elections would be natural and interesting as well. Indeed, there is hope more efficient protocols exist for such restrictions.

6.3.5. Incomplete votes

In this paper we considered situations in which voters provide full votes; e.g., in the ordinal elections we consider, each voter is assumed to provide a full ranking. There are quite a few papers dealing with settings in which voters provide only partial rankings: Pini et al. [27] and Xia and Conitzer [32] study the complexity of computing *possible* winners (alternatives for which at least one completion of the votes makes them win the election) and *necessary* winners (alternatives for which all completions of the votes make them win the elections); Bentert and Skowron [4] and Caragiannis et al. [5] identify voting rules which are suitable to situations in which the vote elicitation is incomplete, in the sense that they approximate well the winners for the full elicitation. It would be interesting to try to generalize our communication protocols to such settings in which voters provide only partial ranking.

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

The authors thank Robert Krauthgamer for inspiring discussions.

Appendix A. Proof of Lemma 2

Proof of Lemma 2. Set $\delta = \frac{\epsilon}{4}$. As c is a δ-winner in E, it follows that there exists a set of voters $u_1, \ldots, u_{q'}$, where $q' \le \delta n$, such that $c \in \mathcal{R}(\tilde{E})$ for $\tilde{E} = E \cup \{u_1, \ldots, u_{q'}\}$: that is, adding those q' voters to E would make c a winner. The situation is that we have an additional q voters, $v_{n+1}, \ldots, v_{n+q}$, which might have a bad impact with respect to c. Thus, our goal is to describe an additional set of $q'' \le 3q$ voters, denoted by $W = \{w_1, \ldots, w_{q'' \le 3q}\}$ which will nullify the (possibly) bad impact of those q voters (which arrived after the last checkpoint) on c.

So, for each of the voting rules we consider in this paper, we will argue that $c \in \mathcal{R}(\tilde{E}')$ where $\tilde{E}' = E' \cup \{w_1, \ldots, w_{q''}\} \cup \{u_1, \ldots, u_{q'}\} = \tilde{E} \cup \{v_{n+1}, \ldots, v_{n+q}\} \cup \{w_1, \ldots, w_{q''}\}$. Thus, we will conclude that c is a $4\delta = \epsilon$-winner with respect to E'. Below we describe the set of voters W for each voting rule separately.

- **Plurality, *t*-Approval, Approval**: For $i \in [q]$, let w_i be a voter approving c, and such that w_i is not approving any candidate which was approved by v_{n+i} (recall that in the case of t-Approval we assume $t \le m/2$).
  As c is a winner in both the elections with voters $\{v_{n+1}, \ldots, v_{n+q}, w_1, \ldots, w_q\}$ and $\tilde{E}$, it holds that $c \in \mathcal{R}(\tilde{E}')$.
- **Borda**: For $i \in [q]$, let w_i be the "reverse" of v_{n+i} (e.g., if $v_{n+i} : a \succ b \succ c$, then $w_i : c \succ b \succ a$). Note that all candidates have the same Borda score with respect to the voters $\{v_{n+1}, \ldots, v_{n+q}, w_1, \ldots, w_q\}$. Thus $c \in \mathcal{R}(\tilde{E})$ implies $c \in \mathcal{R}(\tilde{E}')$.
- **Cup**: Denote the set of candidates by M and consider the election $\tilde{E}$. In order to compute a Cup-winner, we shall preform a series of $m-1$ "head-to-head" contests. That is, there is a set $P \subseteq M \times M$ of ordered pairs, of size $m-1$, such that for every $(c_1, c_2) \in P$, c_1 wins c_2 in an head-to-head contest. In fact, in any election $\hat{E}$ such that for every $(c_1, c_2) \in P$, c_1 wins c_2 in an head-to-head contest with respect to $\hat{E}$, it holds that c is a Cup-winner.
  In the beginning of the proof of Theorem 7 we argued that there is an order π_P over M, such that for every $(c_1, c_2) \in P$, c_1 precedes c_2 in π_P. Next we define $w_1, \ldots, w_q$. All these voters will order the candidates with respect to π_P: that is, the maximal candidate in π_P will be ranked first, the second will be ranked second, and so on. Now, for every $(c_1, c_2) \in P$, c_1 wins c_2 in the "head-to-head" contest with respect to $\{v_{n+1}, \ldots, v_{n+q}, w_1, \ldots, w_q\}$, hence c_1 wins c_2 in the "head-to-head" contest with respect to $\tilde{E}'$. We conclude that $c \in \mathcal{R}(\tilde{E}')$.
- **Copeland and Condorcet**: We prove the claim for Copeland first. For $i \in [q]$, let u_i be the "reverse" of v_i. For every two candidates c_1, c_2, a majority of the voters prefer c_1 to c_2 with respect to $\tilde{E}$ if and only if a majority of the voters prefer c_1 to c_2 with respect to $\tilde{E}'$. Thus, $c \in \mathcal{R}(\tilde{E})$ implies $c \in \mathcal{R}(\tilde{E}')$. The above proves the claim for Copeland; thus the claim for Condorcet follows, as every Copeland winner is in particular a Condorcet winner.
- **Bucklin**: For $i \in [q]$, let w_i be a voter ranking c on top, and such that every candidate $c' \ne c$, which is ranked at position j in v_{n+i}, will be ranked at position $m-j+1$ or $m-j+2$ in v_j. Note that, for every $j \le \frac{m}{2}$ and $c' \ne c$, the number of voters among $v_{n+1}, \ldots, v_{n+q}, w_1, \ldots w_q$ ranking c' among the first j positions is at most q, while the number of voters ranking c among the first j position is at least q.

Set $n' = n + q' + 2q$. Suppose that in the elections $\tilde{E}$, c wins at round j. Consider the election $\tilde{E}'$. Then, for every candidate $c' \neq c$ and $j' < j$, the number of voters ranking c' among the first j' positions is less than $\frac{n+q'}{2} + q = \frac{n'}{2}$, while the number of voters ranking c among the first j positions is at least $\frac{n+q'}{2} + q = \frac{n'}{2}$. We conclude that $c \in \mathcal{R}(\tilde{E}')$.

- **Run Off**: Let c' be a candidate such that c and c' get the highest plurality score in $\tilde{E}$, and such that c is winning c' in the "head-to-head" contest with respect to $\tilde{E}$. Set $w_1, \ldots, w_q$ to be voters ranking c' on top, and set $w_{q+1}, \ldots, w_{3q}$ to be voters ranking c on top. Note that with respect to the voters $v_{n+1}, \ldots, v_{n+q}, w_1, \ldots, w_{3q}$, c and c' have the highest plurality score, while c is winning over c' in the "head-to-head" contest. Thus this is also the situation in $\tilde{E}'$. We conclude that $c \in \mathcal{R}(\tilde{E}')$. □

Appendix B. Communication lower bound for No Strict Majority

A basic machinery for communication complexity lower bounds is fooling sets. Consider a function $f : \{0,1\}^{m \times k} \to \{0,1\}$. We have k players, each holding a string from $\{0,1\}^m$.

Definition 2 *(Fooling set).* A set $A = \{(x_1^1, \ldots, x_k^1), \ldots, (x_1^s, \ldots, x_k^s)\} \subseteq \{0,1\}^{m \times k}$ is called a *fooling set* for the function $f : \{0,1\}^{m \times k} \to \{0,1\}$, if there are some bit $b \in \{0,1\}$ such that:

1. For every i, $f(x_1^i, \ldots, x_k^i) = b$.
2. For every $i \neq j$, there is $(y_1, \ldots, y_k) \in \{x_1^i, x_1^j\} \times \cdots \times \{x_k^i, x_k^j\}$ such that $f(y_1, \ldots, y_k) \neq b$.

A fooling set is called a 1-*fooling set* if the bit b above is 1 (similarly, a 0-*fooling set*). The proof of the following fact can be found in the textbook by Kushilevitz and Nisan [23].

Fact 1. *Let $f : \{0,1\}^{n \times k} \to \{0,1\}$ be some function with fooling set A. Then, $D(f) \geq \log |A|$.*

We denote by $f^l = \bigwedge_{i=1}^{l} f : \{0,1\}^{n \times k \times l} \to \{0,1\}$ a function that gets as input l inputs for f and returns 1 if and only if the output of all the l instance is 1. Formally, $f^l\left((x_1^1, \ldots, x_k^1), \ldots, (x_1^l, \ldots, x_k^l)\right) = f(x_1^1, \ldots, x_k^1) \wedge f(x_1^2, \ldots, x_k^2) \wedge \cdots \wedge f(x_1^l, \ldots, x_k^l)$. The proof of the following lemma is straightforward, albeit we attach the proof for completeness.

Lemma 7. *Suppose f has a 1-fooling set of size s. Then f^l has a 1-fooling set of size s^l.*

Proof. Let $A = \{y^1 = (x_1^1, \ldots, x_k^1), \ldots, y^s = (x_1^s, \ldots, x_k^s)\} \subseteq \{0,1\}^{n \times k}$ be a 1-fooling set for f. We argue that A^l (l-wise Cartesian product of A with itself) is a 1-fooling set for $\bigwedge f^l$.

Indeed, for every $\left(y^{i_1}, \ldots, y^{i_l}\right) \in A^l$, it holds that

$$f^l\left(y^{i_1}, \ldots, y^{i_l}\right) = \bigwedge_{j=1}^{l} f(y^{i_j}) = \bigwedge_{j=1}^{l} 1 = 1 .$$

Moreover, take two different points $(y^{i_1}, \ldots, y^{i_l})$ and $(y^{j_1}, \ldots, y^{j_l})$ in A^l. There is some index $r \in [l]$ such that $y^{i_r} \neq y^{j_r}$. Set $y^{i_r} = (z_1, \ldots, z_k)$ and $y^{j_r} = (w_1, \ldots, w_k)$. As $y^{i_r}, y^{j_r} \in A$, there is $x \in \{z_1, w_1\} \times \cdots \times \{z_k, w_k\}$ such that $f(x) = 0$. In particular

$$f^l\left(y^{i_1}, \ldots, y^{i_{r-1}}, x, y^{i_{r+1}}, \ldots, y^{i_l}\right) = \bigwedge_{j \in [l] \setminus \{r\}} f(y^{i_j}) \wedge f(x) = \bigwedge_{[l] \setminus \{r\}} 1 \wedge 0 = 0,$$

as required. □

Now, we are ready to prove Lemma 6.

Proof of Lemma 6. Using Lemma 7 and Fact 1, it will be enough to show that $\mathsf{NSM}_{2k,m,\frac{1}{4}}$ has a 1-fooling set of size $\Omega(mk)$.

We start by defining n metrics over $\{0,1\}^{m \times 2k}$:

$$d_i\left((A_1, \ldots, A_{2k}), (B_1, \ldots, B_{2k})\right) = \left|\left\{j \mid i \in A_j \triangle B_j\right\}\right|$$

Here, $A_j \triangle B_j = \left(A_j \setminus B_j\right) \cup \left(B_j \setminus A_j\right)$ is the symmetric difference.[11]

[11] In fact d_i is just the Hamming distance after we project the strings to the i'th coordinate.

$$d\left((A_1,\ldots,A_{2k}),(B_1,\ldots,B_{2k})\right)=\max_i d_i\left((A_1,\ldots,A_{2k}),(B_1,\ldots,B_{2k})\right)\ .$$

It is straightforward to verify that d is indeed a metric.

Let $\mathcal{S}=\left\{(A_1,\ldots,A_{2k})\in\{0,1\}^{m\times 2k}\mid \forall i\in[m],\ \left|\left\{j\mid i\in A_j\right\}\right|=k\right\}$ be all the points such that every index $i\in[m]$ appears in exactly k sets. Note that $|\mathcal{S}|=\binom{2k}{k}^m$, and that $\forall x\in\mathcal{S}$, $\mathsf{NSM}_{2k,m,\frac{1}{4}}(x)=1$. We will construct a subset $\mathcal{S}'\subseteq\mathcal{S}$ in a greedy manner. In each phase we will choose an arbitrary $x\in\mathcal{S}$, which was not deleted yet, add it to $\mathcal{S}'$ and delete all of $B(x,k/2)$, i.e., all the points in $\mathcal{S}$ which are at distance at most $k/2$ from x (with respect to the metric d).

It holds that

$$|B(x,k/2)\cap\mathcal{S}|=\left(\sum_{i=0}^{k/4}\binom{k}{i}^2\right)^m\le\left(2\cdot\binom{k}{k/4}^2\right)^m\ .$$

To see the equality, denote $x=(A_1,\ldots,A_{2k})$. For each index $i\in[m]$, there are k sets containing i. We should choose $j\le k/4$ sets to remove i from, and j new sets to insert i into. All this is taken in power of m as we have m different indices. To see the inequity, note that for $i\le k/4$, $\binom{k}{i}/\binom{k}{i-1}=\frac{k-i+1}{i}>2$. Hence $\sum_{i=0}^{\frac{k}{4}-1}\binom{k}{i}^2<\binom{k}{k/4}^2$.

By the end of the process (when all the points in $\mathcal{S}$ were deleted), we have a set $\mathcal{S}'$ of size at least $\left(\frac{\binom{2k}{k}}{2\cdot\binom{k}{k/4}^2}\right)^m$ such that for every $x,y\in M'$, $d(x,y)\ge k/2$. We argue that $\mathcal{S}'$ is a 1-fooling set. As $\mathcal{S}'\subseteq\mathcal{S}$, it holds that $\forall x\in\mathcal{S}'$, $\mathsf{NSM}_{2k,m,\frac{1}{4}}(x)=1$. Consider $x\ne y\in\mathcal{S}'$, where $x=(A_1,\ldots,A_{2k})$ and $y=(B_1,\ldots,B_{2k})$. There is an index i such that $d_i(x,y)\ge k/2$. Therefore, $|\{j\mid i\in A_i\cup B_i\}|\ge\frac{5}{4}k$. In particular, there is $z\in\{A_1,B_1\}\times\cdots\times\{A_{2k},B_{2k}\}$ with at least $\frac{5}{4}$ sets containing i, implying $\mathsf{NSM}_{2k,m,\frac{1}{4}}(z)=0$.

Finally, we lower bound $|\mathcal{S}'|$. Recalling Stirling's formula, which says that $n!\approx\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$, and the identity $\binom{2k}{k}=\sum_{i=0}^{k}\binom{k}{i}^2$, we have that:

$$\frac{\binom{2k}{k}}{2\binom{k}{k/4}^2}\ge\frac{\binom{k}{k/2}^2}{\binom{k}{k/4}^2}=\frac{\left(\frac{k}{4}!\right)^2\left(\frac{3k}{4}!\right)^2}{\left(\frac{k}{2}!\right)^4}=\Omega(1)\cdot\frac{\left(\frac{k}{4}\right)^{\frac{1}{2}k}\left(\frac{3k}{4}\right)^{\frac{3}{2}k}}{\left(\frac{k}{2}\right)^{2k}}=\Omega(1)\cdot\left(\frac{3^{\frac{3}{2}}}{4}\right)^k\ .$$

We conclude that

$$D\left(\mathsf{NSM}_{2k,m,\frac{1}{4}}\right)\ge\log\left(\left|\mathcal{S}'\right|\right)\ge\log\left(\left(\Omega(1)\cdot\left(\frac{3^{\frac{3}{2}}}{4}\right)^k\right)^m\right)=\Omega(mk)\ .\qquad\square$$

References

[1] C. Arackaparambil, J. Brody, A. Chakrabarti, Functional monitoring without monotonicity, in: Automata, Languages and Programming, 2009, pp. 95–106.
[2] A. Bhattacharyya, P. Dey, Fishing out winners from vote streams, preprint, arXiv:1508.04522, 2015.
[3] B. Babcock, C. Olston, Distributed top-k monitoring, in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, CDM '03, 2003, pp. 28–39.
[4] Matthias Bentert, Piotr Skowron, Comparing election methods where each voter ranks only few candidates, preprint, arXiv:1901.10848, 2019.
[5] Ioannis Caragiannis, Xenophon Chatzigeorgiou, George A. Krimpas, Alexandros A. Voudouris, Optimizing positional scoring rules for rank aggregation, Artif. Intell. 267 (2019) 58–77.
[6] Y. Chevaleyre, J. Lang, N. Maudet, J. Monnot, Compilation and communication protocols for voting rules with a dynamic set of candidates, in: Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge, TARK '11, 2011, pp. 153–160.
[7] Y. Chevaleyre, J. Lang, N. Maudet, G. Ravilly-Abadie, Compiling the votes of a subelectorate, in: Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI '09, 2009, pp. 97–102.
[8] T. Csar, M. Lackner, R. Pichler, E. Sallinger, Winner determination in huge elections with MapReduce, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI '17, 2017, pp. 451–458.
[9] G. Cormode, S. Muthukrishnan, Ke. Yi, Algorithms for distributed functional monitoring, ACM Trans. Algorithms 7 (2) (2011) 21.
[10] G. Cormode, S. Muthukrishnan, K. Yi, Q. Zhang, Continuous sampling from distributed streams, J. ACM 59 (2) (2012) 10.
[11] G. Cormode, The continuous distributed monitoring model, SIGMOD Rec. 42 (1) (2013) 5–14.
[12] Chaturvedi Kamal, Shrisha Rao, Distributed elections using site-push majority winner monitoring, IEEE Syst. J. (2019).
[13] V. Conitzer, T. Sandholm, Vote elicitation: complexity and strategy-proofness, in: Proceedings of the 18th National Conference on Artificial Intelligence, AAAI '02, 2002, pp. 392–397.
[14] V. Conitzer, T. Sandholm, Communication complexity of common voting rules, in: Proceedings of the 6th ACM Conference on Electronic Commerce, EC' 05, 2005, pp. 78–87.
[15] Palash Dey, Arnab Bhattacharyya, Sample complexity for winner prediction in elections, in: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '15, 2015, pp. 1421–1430.
[16] Swapnil Dhamal, Y. Narahari, Scalable preference aggregation in social networks, in: Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing, HCOMP '13, 2013.

[17] P. Dey, Y. Narahari, Estimating the margin of victory of an election using sampling, in: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI '15, 2015, pp. 1120–1126.
[18] Palash Dey, Nimrod Talmon, Otniel van Handel, Proportional representation in vote streams, in: Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems, AAMAS 2017, 2017, pp. 15–23.
[19] E. Elkind, P. Faliszewski, A. Slinko, Swap bribery, in: Proceedings of the 2nd International Symposium on Algorithmic Game Theory, SAGT '09, October 2009, pp. 299–310.
[20] Piotr Faliszewski, Jörg Rothe, Control and bribery in voting, in: F. Brandt, V. Conitzer, U. Endriss, J. Lang, A.D. Procaccia (Eds.), Handbook of Computational Social Choice, Cambridge University Press, 2015, chapter 7.
[21] Arnold Filtser, Nimrod Talmon, Distributed monitoring of election winners, in: Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems, AAMAS 2017, 2017, pp. 1160–1168.
[22] Z. Huang, K. Yi, Q. Zhang, Randomized algorithms for tracking distributed count, frequencies, and ranks, in: Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems, PODS '12, 2012, pp. 295–306.
[23] Eyal Kushilevitz, Noam Nisan, Communication Complexity, Cambridge University Press, 1997.
[24] D.T. Lee, Efficient, private, and ϵ-strategyproof elicitation of tournament voting rules, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI '15, 2015.
[25] D.T. Lee, A. Goel, T. Aitamurto, H. Landemore, Crowdsourcing for participatory democracies: efficient elicitation of social choice functions, in: Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing, HCOMP' 14, 2014.
[26] Z. Liu, B. Radunovic, M. Vojnovic, Continuous distributed counting for non-monotonous streams, in: Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '12, 2012, pp. 307–318.
[27] Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, Toby Walsh, Incompleteness and incomparability in preference aggregation: complexity results, Artif. Intell. 175 (7–8) (2011) 1272–1289.
[28] Alexander A. Razborov, Communication complexity, in: Dierk Schleicher, Malte Lackmann (Eds.), An Invitation to Mathematics: From Competitions to Research, 2011.
[29] Travis C. Service, Julie A. Adams, Communication complexity of approximating voting rules, in: International Conference on Autonomous Agents and Multiagent Systems, 3 Volumes, AAMAS 2012, Valencia, Spain, June 4–8, 2012, pp. 593–602.
[30] S. Tirthapura, D.P. Woodruff, Optimal random sampling from distributed streams revisited, in: Proceeding of the 25th international conference on Distributed computing, DISC '11, 2011, pp. 283–297.
[31] L. Xia, V. Conitzer, Compilation complexity of common voting rules, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI '10, 2010, pp. 915–920.
[32] Lirong Xia, Vincent Conitzer, Determining possible and necessary winners given partial orders, J. Artif. Intell. Res. 41 (2011) 25–67.
[33] L. Xia, Computing the margin of victory for various voting rules, in: Proceedings of the 13th ACM Conference on Electronic Commerce, EC' 12, 2012, pp. 982–999.
[34] K. Yi, Q. Zhang, Optimal tracking of distributed heavy hitters and quantiles, Algorithmica 65 (1) (2013) 206–223.