Accelerating Voting by Quantum Computation

Ao Liu[1] **Qishen Han**[1] **Lirong Xia**[1] **Nengkun Yu**[2]

[1]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA
[2]Department of Computer Science, Stony Brook University, Stony Brook, NY, USA

Abstract

Studying the computational complexity and designing fast algorithms for determining winners under voting rules are classical and fundamental questions in computational social choice. In this paper, we accelerate voting by leveraging quantum computation: we propose a quantum-accelerated voting algorithm that can be applied to any anonymous voting rule. We show that our algorithm can be quadratically faster than any classical algorithm (based on sampling with replacement) under a wide range of common voting rules, including positional scoring rules, Copeland, and single transferable voting (STV). Precisely, our quantum-accelerated voting algorithm outputs the correct winner with high probability in $\Theta\left(\frac{n}{\text{MoV}}\right)$ time, where n is the number of votes and MoV is *margin of victory*, the smallest number of voters to change the winner. In contrast, any classical voting algorithm based on sampling with replacement requires $\Omega\left(\frac{n^2}{\text{MoV}^2}\right)$ time under a large class of voting rules. Our theoretical results are supported by experiments under plurality, Borda, Copeland, and STV.

1 INTRODUCTION

Driven by the critical public need of revolutionalizing modern democratic systems [Mancini, 2015, Brill, 2018], voting has been widely applied in many collective decision-making scenarios beyond political elections. Examples include search engines [Dwork et al., 2001], crowdsourcing [Mao et al., 2013], database management [Belardinelli and Grandi, 2019], and blockchain governance [Grossi, 2022], just to name a few. In such large-scale, high-frequency collective decision-making scenarios, it is desirable that the winner is computed as soon as possible, ideally in sub-linear time in the number of votes, and perhaps at a (small) cost of its correctness. In fact, the study of computational complexity and algorithmic aspects of voting rules has been a key topic of *computational social choice* [Brandt et al., 2016].

One natural approach is to randomly sample a subset of votes (with or without replacement) and compute the winner of the sampled votes. The idea can be dated back to Venetian elections in the 13th century [Walsh and Xia, 2012] and has recently attracted much attention from the computational social choice community [Bhattacharyya and Dey, 2021, Flanigan et al., 2020, 2021]. However, the performance of a sampling algorithm is restricted by the number of samples needed to guarantee a certain level of correctness, which determines its runtime. Specifically, the runtime of a sampling algorithm would be quadratically related to the number of votes (see Table 1). Is there a faster algorithm, for example, sub-linear to the number of votes, that still preserves a high probability of correctness?

Quantum computation appears to be a promising approach, as it has successfully accelerated many computational tasks such as search [Grover, 1996], optimization [Hogg and Portnov, 2000], and machine learning [Benedetti et al., 2016, Ajagekar and You, 2020, 2021]. However, we are not aware of previous work on accelerating voting using quantum computation. Thus, the following problem remains open.

Can voting be accelerated by quantum computation?

We address this question with YES both theoretically and experimentally. We accelerate voting by designing a sub-linear quantum-accelerated voting algorithm where a small probability of "errors" is allowed, which outperforms the classical sampling-based voting algorithms. Our contributions are three-fold. First, we propose the quantum-accelerated voting algorithm (Algorithm 1). Our algorithm leverages the widely-used techniques of quantum counting [Brassard et al., 1998] to generate the histogram of the votes. The simple architecture guarantees that our algorithm will be easy to implement in the future. Second, we theoretically prove that our algorithm is quadratically faster than any classical

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

sampling-based algorithm for many common voting rules, including positional scoring rule, STV, Copeland, and maximin (see Table 1). Third, we experimentally verify our theoretical results under the plurality, Borda, Copeland, and STV (Section 6). In Section 7, we provide heuristics that may further improve the performance of quantum-accelerated voting.

	Runtime	Space Requirement
Quantum (Thm. 2)	$\Theta\left(\frac{n}{\text{MoV}}\right)$	$\Theta\left(\log(\frac{n}{\text{MoV}})\right)$
Classical (Thm. 3)	$\Omega\left(\frac{n^2}{\text{MoV}^2}\right)$	$\Omega\left(\log(\frac{n^2}{\text{MoV}^2})\right)$

Table 1: Summary for the theoretical results, where n is the number of votes, and MoV is the margin of victory. All results in the table assume a constant error rate (*e.g.,* 1%).

Our quantum-accelerated voting algorithm accelerates voting most significantly in the case where the margin of victory is sub-linear in n, *e.g.,* $\text{MoV} = \Theta(n^c)$ with $c \in (0, 1)$. In this case, MoV is relatively small compared with n, and a $\Theta(\frac{n}{\text{MoV}})$ acceleration from classical to quantum algorithm is significant. The ratio $\frac{n}{\text{MoV}} = \Theta(n^{1-c})$ implies that the algorithm is sub-linear. See Section 5 for detailed discussions.

Related works and discussions. To the best of our knowledge, our work is the first to use quantum computation to accelerate voting. Vaccaro et al. [2007]introduced the idea of quantum computation to voting. The quantum voting algorithm in Vaccaro et al. [2007] provides security guarantees (against colluding attacks [Lian and Zhang, 2009]). Xue and Zhang [2017] improved the result in Vaccaro et al. [2007] by proposing a simpler voting protocol but with stronger security guarantees. Khabiboulline et al. [2021] focuses on achieving anonymity without losing security guarantees. However, all approaches above require $\Omega(n)$ quantum communication cost and thus take $\Omega(n)$ time.

There is a large literature on efficient (classical) algorithms for the winner determination problem. Wang et al. [2019] purposed fast algorithms to compute winners in ranked pairs and STV under parallel-universes tiebreaking [Conitzer et al., 2009], which is known to be NP-complete. Various papers have shown that the winner of Dodgson rule, while is NP-hard to compute in the worst case [Bartholdi et al., 1989], can be efficiently computed with high probability when the ranking is generated $i.i.d.$ [McCabe-Dansted et al., 2008, Homan and Hemaspaandra, 2009] and under semi-random models [Xia and Zheng, 2022]. This line of work focuses on designing algorithms for NP-hard winner determination problems, while our paper focuses on further accelerating voting whose winner determination problem is in P.

2 PRELIMINARIES

VOTING.
In voting, $n > 1$ voters cast their votes on $m > 1$ candidates. The candidates are denoted as $c_1, \cdots, c_m$. A vote represents a voter's preference towards the candidates, which is a full-ranking (linear order) over candidates. Since there are $m!$ types of full rankings for m candidates, a vote can be represented as an $m!$-dimensional unit vector. That means if the vote is the j-th type, the j-th dimension of the vector is 1, and all other dimensions are 0's. The vote of the i-th voter is denoted as $\boldsymbol{V}_i = (V_{i,1}, \cdots, V_{i,m!})$. For example, if the i-th vote is the j-th type, we have that $\boldsymbol{V}_{i,j} = 1$ and $V_{i,j'} = 0$ for all $j' \neq j$. A profile P is a collection of n voters' rankings. A voting rule r is a mapping from the profile P to the winner(s) among m candidates.

A histogram **hist** is an $m!$ dimension vector that records the number of each type of ranking in the profile. We use $r(\textbf{hist})$ to denote the winner under an anonymous voting rule r and a profile with histogram **hist**. In this paper, we assume that the voting rule r satisfies *anonymity* and *canceling out* [Liu et al., 2020]. An anonymous voting rule selects the winner only based on the histogram of the profile and does not depend on the identity of the voter. A voting rule satisfies canceling out if the winner does not change after adding one copy of each ranking to the profile. Most common voting rules satisfy both anonymity and canceling-out, such as plurality, Borda, Copeland, and STV.

Plurality. The candidate ranked top in the most number of votes is chosen as the winner.

Borda. Firstly, the rule calculates the Borda score of each candidate. A candidate ranked i-th in a vote gains a score of $(m-i)$ from that vote. For example, the Borda scores for the vote $[c_1 \succ c_2 \succ c_3 \succ c_4]$ are $\{c_1 : 3,\ c_2 : 2,\ c_3 : 1,\ c_4 : 0\}$. The Borda score of a candidate is the sum of scores it gains from each vote. Then, the candidate with the largest Borda score is chosen as the winner.

Copeland. Copeland compares each pair of candidates, where the winner gets 1 point, and the loser gets 0 point. If two candidates are tied, both get 0.5 points. After finishing all pairwise comparisons, the candidate receiving the most points is chosen as the winner.

Single transferable vote (STV). STV is an $(m-1)$-round voting rule. In each round, the candidate receiving the least number of top-ranked votes is eliminated. When all $(m-1)$ rounds are finished, the remaining candidate is chosen as the winner.

Margin of victory. *Margin of victory* (MoV) is the smallest number k such that there exist a set of k voters who can change the winner by voting differently.

QUANTUM COMPUTATION.

Quantum counting algorithm[1] **[Brassard et al., 1998]**. Quantum counting is one of the key parts of our proposed quantum-accelerated voting algorithm. It counts the number of solutions to a search problem. Given a binary function $f : \{0, 1, \cdots, 2^t - 1\} \to \{0, 1\}$, the quantum counting circuit for f computes the number of $x \in \{0, 1, \cdots, 2^t - 1\}$ such that $f(x) = 1$. A quantum counting circuit uses t quantum bits to encode the function and s quantum bits to calculate and record the output. More specifically, supposing n_1 is the number of x such that $f(x) = 1$. The quantum counting circuit with $t + s$ quantum bits outputs a binary decimal $\hat{\varphi} = 0.b_1b_2 \cdots b_s$, which estimates $\varphi = \arcsin\left(\sqrt{n_1 \cdot 2^{-t}}\right)/\pi$. For example, binary decimal 0.011 represents $(2^{-2} + 2^{-3}) = 3/8$. In the next lemma, we present three useful properties of quantum counting.

Lemma 1 (Properties of quantum counting)**.** *The quantum counting algorithm has the following three properties*
1 (Tail bound, Inequality (5.34) in [Nielsen and Chuang, 2010]). For any $\delta > 2^{-s}$,

$$\Pr[|\hat{\varphi} - \varphi| \geq 2^{-s} + \delta] \leq \frac{1}{2(\delta \cdot 2^s - 1)}.$$

2. Its runtime is $\Theta(2^s)$.
3. Its space requirement is $(t + s)$ quantum bits (qubits).

The first property in Lemma 1 says that a larger s provides a stronger theoretical guarantee on the accuracy of quantum counting. However, a larger s also corresponds to longer runtime. The detailed implementation of quantum counting and the reasoning behind why it accelerates can be found in Appendix A.

Applying quantum counting to voting. We apply the quantum counting algorithm to estimate the histogram of a profile. For any $j \in \{1, \cdots, m!\}$, we estimate the j-th type of votes by setting the following binary function:

$$f_j(x) = \begin{cases} 1 & \text{if candidate } x\text{'s vote} \\ & \text{is of the } j\text{-th type} \\ 0 & \text{otherwise} \end{cases}. \quad (1)$$

The number of x's with $f_j(x) = 1$ is the number of votes of j-th type, *i.e.*, $\mathbf{hist}_j$. The histogram of the votes is generated by enumerating all the j's. We assume that the information in $\mathbf{hist}_j$ is stored in quantum RAM, which means the information can be efficiently encoded into quantum circuits [Giovannetti et al., 2008a,b, Park et al., 2019].

3 QUANTUM-ACCELERATED VOTING ALGORITHM

Formal definition of quantum-accelerated voting. We formally define quantum-accelerated voting in Algorithm 1.

[1]This paper adopts the same notation system as Nielsen and Chuang [2010], which is a textbook about quantum computation.

In classical voting, votes are usually sent to an "aggregator", who is responsible for aggregating the votes and announcing the winner. Our quantum-accelerated voting follows a similar procedure, where the "aggregator" uses an algorithm accelerated by quantum computation. Basically, Algorithm 1 repeats the quantum counting algorithm by K rounds. In each round, quantum counting estimates the histogram of the profile **hist** and applies the voting rule to the estimated histogram $\widehat{\mathbf{hist}}$ to compute an estimated winner $c^{(k)} = r(\widehat{\mathbf{hist}})$. Then, the classical plurality voting rule is used to aggregate the estimated winners of K rounds, *i.e.* $c^{(1)}, \cdots, c^{(K)}$. That is: the "aggregator" announces the candidate who is the estimated winner of the most rounds as the (final) winner. A tie-breaking rule (*e.g.,* random tie-breaking) is applied when there are multiple candidates with the most rounds as the estimated winner. In Section 4, we will show that Algorithm 1 (a combination of quantum and classical algorithms) has a better runtime than the quantum algorithm only.

Algorithm 1: Quantum-Accelerated Voting Algorithm

Inputs: n voters' votes $\boldsymbol{V}_0, \cdots, \boldsymbol{V}_{n-1}$, a voting rule r, number of iteration K, and the number of qubits $s \geq 2$
Initialization: Construct the binary functions $f_1, f_2, \cdots, f_{m!}$ based on $\boldsymbol{V}_0, \cdots, \boldsymbol{V}_{n-1}$
for $k \in \{1, \cdots, K\}$ **do**
  Initialize an $m!$-dimensional vector $\widehat{\mathbf{hist}}$
  for $j \in \{1, \cdots, m!\}$ **do**
    Construct and apply quantum counting circuit for f_j with s qubits. Denote the output of the quantum counting circuit as $0.b_1 \cdots b_s$.
    Set the j-th component of $\widehat{\mathbf{hist}}$ as $2^t \cdot \sin^2(\pi \cdot 0.b_1 \cdots b_s)$
  end for
  Set $c^{(k)} = r(\widehat{\mathbf{hist}})$ as the winner of the k-th round
end for
Output the (classical) plurality winner of $c^{(1)}, \cdots, c^{(K)}$.

Construct binary functions. We construct a binary function f_j to store the information about the j-th type of vote. $f_j(x) = 1$ if and only if the x-th voter casts the j-th type of vote. Formally, $f_j : \{0, 1, \cdots, 2^t - 1\} \to \{0, 1\}$ is defined according to Equation (1), where $t = \lceil \log n \rceil$ is the number of bits to encode n, and the voters are numbered from 0 to $(n - 1)$. For $x \in \{n, \cdots, 2^t - 1\}$, we just set $f_j(x) = 0$ for padding. Then, the number of x such that $f_j(x) = 1$ is exactly the number of votes of the j-th type, *i.e.* $\mathbf{hist}_j$.

Count the histogram. For each type j, a quantum counting circuit is constructed and used to estimate $\mathbf{hist}_j$. The output of the counting circuit is $\hat{\varphi} = 0.b_1b_2 \cdots b_s$, which estimates $\varphi = \arcsin\left(\sqrt{\mathbf{hist}_j \cdot 2^{-t}}\right)/\pi$. Therefore, $\widehat{\mathbf{hist}}_j = 2^t \cdot \sin^2(\pi \cdot 0.b_1 \cdots b_s)$ estimates $\mathbf{hist}_j$. By enumerating all types j, we get $\widehat{\mathbf{hist}}$, which estimates the histogram **hist**.

Decide the winner. The quantum counting procedure (and the voting rule) runs for K rounds. Each round's estimated winner is computed according to the estimated histogram $\widehat{\mathbf{hist}}$. Finally, the quantum-accelerated voting algorithm outputs the candidate that wins in the most number of rounds.

4 THEORETICAL ANALYSIS OF QUANTUM-ACCELERATED VOTING

In this section, we provide theoretical guarantees about Algorithm 1's $\Pr[\text{correct}]$ [2], runtime, and space requirements. In our analysis, the number of candidates m is fixed. Figure 1 illustrates a roadmap of the results in this section.

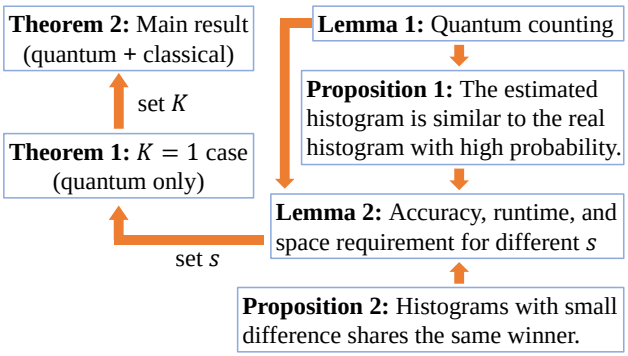

Figure 1: The logical chain of the theorems, lemmas, and propositions about Algorithm 1. For example, the arrow from Theorem 1 to Theorem 2 represents that Theorem 2 is proved based on Theorem 1.

In Theorem 1, we show the theoretical guarantee of Algorithm 1 when $K = 1$. Note that when $K = 1$, the outer for loop in Algorithm 1 only runs for one round, which means the classical plurality voting rule directly outputs $c^{(1)}$. In other words, Algorithm 1 only contains quantum counting (and the voting rule r) when $K = 1$. To simplify notations, we define a function $\sigma(\cdot)$ as follows.

$$\sigma(\varepsilon) = 2 + \left\lceil t + \log\left(\frac{m!}{2\varepsilon} + 2\right) + \log\left(\pi \cdot (m!)\right) - \log(\text{MoV}) \right\rceil = \Theta\left(\log\left(\frac{n}{\varepsilon \cdot \text{MoV}}\right)\right).$$

Throughout this paper, the logarithm function $\log(\cdot)$ represents $\log_2(\cdot)$ and $\ln(\cdot)$ represents $\log_e(\cdot)$.

Theorem 1 (Theoretical guarantee of Algorithm 1 when $K = 1$)**.** *For any given $\varepsilon \in (0, 1)$, Algorithm 1 with $K = 1$ and $s = \sigma(\varepsilon)$ has the following three properties.*
1. Its $\Pr[\text{correct}] \geq 1 - \varepsilon$.
2. Its runtime is $\Theta\left(\frac{n}{\varepsilon \cdot \text{MoV}}\right)$.
3. Its space requirement is $\Theta\left(\log(\frac{n}{\varepsilon \cdot \text{MoV}})\right)$.

[2]Throughout this paper, we use $\Pr[\text{correct}]$ to represent the probability (for an algorithm) to output the correct winner. Formally, $\Pr[\text{correct}] \triangleq \Pr[\text{the algorithm's output} = r(P)]$, where P is the voting profile and r is the voting rule.

For readability, we will present the proof of Theorem 1 after the proof of Theorem 2. In Theorem 2, we will present the theoretical guarantee of Algorithm 1 when parameter K is set properly. Under this setting, Algorithm 1 contains both quantum and classical aggregation methods. Comparing Theorem 1 with Theorem 2, we know the (classical) plurality voting can reduce the $(1/\epsilon)$ term in runtime to $\log(1/\epsilon)$.

Theorem 2 (Theoretical guarantee of Algorithm 1)**.** *For any given $\varepsilon \in (0, 1)$, Algorithm 1 with $K = \lfloor 24 \ln(1/\varepsilon) \rfloor + 1$ and $s = \sigma(1/4) = \Theta\big(\log(n/\text{MoV})\big)$ has the following three properties.*
1. Its $\Pr[\text{correct}] \geq 1 - \varepsilon$.
2. Its runtime is $\Theta\left(\frac{n \log(1/\varepsilon)}{\text{MoV}}\right)$.
3. Its space requirement is $\Theta\left(\log(\frac{n \log(1/\varepsilon)}{\text{MoV}})\right)$.

Proof. Setting $s = \sigma(1/4)$, we know that Algorithm 1 outputs the correct winner in each round with at least $p = \frac{3}{4}$ probability according to Theorem 1. By Chernoff bound,

$$\Pr[\text{correct}] \geq \Pr[r(P) \text{ wins in more than } K/2 \text{ rounds}] > 1 - \exp\left(-\left(1 - \frac{1}{2p}\right)^2 \cdot K \cdot \frac{p}{2}\right).$$

Thus, $\Pr[\text{correct}] \geq 1 - \varepsilon$ holds when

$$K > \frac{2p \cdot \ln(1/\varepsilon)}{(p - 1/2)^2} = 24 \ln(1/\varepsilon).$$

Then, Theorem 2 follows by the monotonicity of Chernoff bound towards K. □

Now, we have presented the two main theorems about Algorithm 1 and are ready to show the full proof of Theorem 1.

Proof of Theorem 1. Theorem 1 follows by setting $s = \sigma(\varepsilon)$ for the following lemma, which bounds $\Pr[\text{correct}]$, runtime, and space complexity of quantum-accelerated voting for different settings on the numbers of qubits s.

Lemma 2. *For any $s \geq 2$, quantum-accelerated voting (Algorithm 1) with $K = 1$ has the following three properties.*
1. Its $\Pr[\text{correct}] \geq 1 - \frac{m!}{2(\delta \cdot 2^s - 1)}$ *for any* $\delta \in (2^{-s}, \frac{\text{MoV}}{2^{t+1}\pi \cdot (m!)} - 2^{-s})$
2. Its runtime is $\Theta(2^s)$
3. Its space requirement is $\Theta(\log(n) + s)$.
Note that Property 1 holds only when the feasible region of δ is non-empty, which sets a constraint for s.

Proof. **Runtime and space requirement.** The runtime and the space requirement properties come from the corresponding properties of quantum counting (see Lemma 1). As the anonymous rule r computes the winner based on an $m!$ dimension histogram, its runtime and space requirement will only depend on the number of candidates m, which is $\Theta(1)$

in our analysis. Thus, when $K = 1$, Algorithm 2 has the following two properties.

$$\text{runtime} = \underbrace{\Theta(1)}_{\text{the voting rule } r} + \underbrace{\Theta(2^s)}_{\text{quantum counting}} = \Theta(2^s).$$

$$\begin{aligned}\text{space requirement} &= \underbrace{\Theta(1)}_{\text{the voting rule } r} + \underbrace{t+s}_{\text{quantum counting}} \\ &= \Theta(1) + \lfloor \log(n) \rfloor + s \\ &= \Theta(\log(n) + s).\end{aligned}$$

Pr[correct]. To simplify the notation, let $\overline{\textbf{hist}}_j$ be the histogram by adding (or removing) the same fraction of each ranking to $\widehat{\textbf{hist}}$ so that the sum of the histogram is n: for all dimension j, $\overline{\textbf{hist}}_j = \widehat{\textbf{hist}}_j + (||\textbf{hist}||_1 - ||\widehat{\textbf{hist}}||_1)/(m!)$ (recall that $||\textbf{hist}||_1 = n$). By the assumption of canceling-out (see the second paragraph in Section 2 for its definition), we know that $r(\overline{\textbf{hist}}) = r(\widehat{\textbf{hist}})$.

The proof of the $\Pr$[correct] property is obtained by the following two propositions. Proposition 1 guarantees that the ℓ_1-norm between **hist** and $\overline{\textbf{hist}}$ is bounded by MoV with at least $1 - \frac{m!}{2(\delta \cdot 2^s - 1)}$ probability. Proposition 2 guarantees that any histogram in the neighborhood of **hist** with ℓ_1-norm smaller than 2MoV shares the same winner with **hist**. Therefore, with at least $1 - \frac{m!}{2(\delta \cdot 2^s - 1)}$ probability, $\overline{\textbf{hist}}$ (and therefore $\widehat{\textbf{hist}}$) leads to the correct winner.

The next proposition shows that $\overline{\textbf{hist}}_j$ is close to $\textbf{hist}_j$ with high probability.

Proposition 1. *For all $\delta \in (2^{-s}, \frac{\text{MoV}}{2^{t+1}\pi \cdot (m!)} - 2^{-s})$, the probability that $\left|\left|\textbf{hist} - \overline{\textbf{hist}}\right|\right|_1 < \text{MoV}$ is at least $1 - \frac{m!}{2(\delta \cdot 2^s - 1)}$.*

Proof. This proof fixes an arbitrary pair of s and δ satisfying the condition. We prove a stronger result that the probability that for all dimension j, $|\overline{\textbf{hist}}_j - \textbf{hist}_j| < \frac{\text{MoV}}{m!}$ is at least $1 - \frac{m!}{2(\delta \cdot 2^s - 1)}$. When the stronger result holds,

$$\left|\left|\textbf{hist} - \overline{\textbf{hist}}\right|\right|_1 \leq \sum_{j=1}^{m!} |\textbf{hist}_j - \overline{\textbf{hist}}_j| < m! \cdot \frac{\text{MoV}}{m!} = \text{MoV}.$$

Therefore, the probability that for all dimension j, $|\overline{\textbf{hist}}_j - \textbf{hist}_j| < \frac{\text{MoV}}{m!}$ is at least $1 - \frac{m!}{2(\delta \cdot 2^s - 1)}$ directly implies that the probability that $\left|\left|\textbf{hist} - \overline{\textbf{hist}}\right|\right|_1 < \text{MoV}$ is at least $1 - \frac{m!}{2(\delta \cdot 2^s - 1)}$.

By applying the union bound on Lemma 1 for all dimensions j, we know that for all dimension j, the probability that $|\hat{\varphi}_j - \varphi_j| < 2^{-s} + \delta$ is at least $1 - \frac{m!}{2(\delta \cdot 2^s - 1)}$. Then we show that $|\hat{\varphi}_j - \varphi_j| < 2^{-s} + \delta$ for all dimension j implies $|\overline{\textbf{hist}}_j - \textbf{hist}_j| < \frac{\text{MoV}}{m!}$ for all j.

Suppose $|\hat{\varphi}_j - \varphi_j| < 2^{-s} + \delta$ holds for all dimension j. Let $h(x) = 2^t \sin^2(\pi x)$. We have $h(\hat{\varphi}_j) = \widehat{\textbf{hist}}_j$ and $h(\varphi_j) = \textbf{hist}_j$. Note that $\frac{dh(x)}{dx} = 2^t \pi \sin(2\pi x) \leq 2^t \pi$ holds for arbitrary x. Therefore,

$$|\widehat{\textbf{hist}}_j - \textbf{hist}_j| \leq 2^t \pi |\hat{\varphi}_j - \varphi_j| < 2^t \pi \cdot (\delta + 2^{-s}).$$

Let $d = 2^t \pi \cdot (\delta + 2^{-s})$. By summing over all dimensions j, we have

$$\left| ||\widehat{\textbf{hist}}||_1 - ||\textbf{hist}||_1 \right| \leq \sum_{j=1}^{m!} |\widehat{\textbf{hist}}_j - \textbf{hist}_j| < (m!)d.$$

Given the bound of difference between $\widehat{\textbf{hist}}$ and **hist**, we show that since $\overline{\textbf{hist}}$ is the modified $\widehat{\textbf{hist}}$, the difference between $\overline{\textbf{hist}}$ and **hist** is also bounded. Now we consider the difference between $\overline{\textbf{hist}}_j$ and $\textbf{hist}_j$ for any dimension j.

For the upper bound, we have $\widehat{\textbf{hist}}_j < \textbf{hist}_j + d$ and $(||\textbf{hist}||_1 - ||\widehat{\textbf{hist}}||_1) < (m!)d$. Therefore,

$$\begin{aligned}\overline{\textbf{hist}}_j &= \widehat{\textbf{hist}}_j + (||\textbf{hist}||_1 - ||\widehat{\textbf{hist}}||_1)/(m!) \\ &< \textbf{hist}_j + 2d.\end{aligned}$$

Similarly, for the lower bound, we have $\widehat{\textbf{hist}}_j > \textbf{hist}_j - d$. Therefore, $\overline{\textbf{hist}}_j > \textbf{hist}_j - 2d$.

Therefore, for all dimension j, $|\overline{\textbf{hist}}_j - \textbf{hist}_j| < 2d$. The condition that $\delta < \frac{\text{MoV}}{2^{t+1}\pi \cdot (m!)} - 2^{-s}$ guarantees that $2d = 2^{t+1}\pi \cdot (\delta + 2^{-s}) < \frac{\text{MoV}}{m!}$. Therefore, for all dimension j, $|\textbf{hist}_j - \overline{\textbf{hist}}_j| < \frac{\text{MoV}}{m!}$, which finishes our proof. □

Proposition 2. *For any histogram $\overline{\textbf{hist}}$ such that $||\textbf{hist}||_1 = \left|\left|\overline{\textbf{hist}}\right|\right|_1$ and $\left|\left|\textbf{hist} - \overline{\textbf{hist}}\right|\right|_1 < 2\text{MoV}$, $r(\textbf{hist}) = r(\overline{\textbf{hist}})$.*

Proof. We consider the voting rule r that allows voters to vote fractionally. That is, each voter has a total weight of 1 and can assign the weight arbitrarily to every ranking. Accordingly, MoV is the smallest amount of weight of votes to change the winner.

Suppose the statement is not true, and there exists a $\overline{\textbf{hist}}$ such that $\left|\left|\textbf{hist} - \overline{\textbf{hist}}\right|\right|_1 < 2\text{MoV}$ and $r(\textbf{hist}) \neq r(\overline{\textbf{hist}})$. Let J_1 be the set of dimension j such that $\overline{\textbf{hist}}_j > \textbf{hist}_j$, and J_2 be the set of j such that $\overline{\textbf{hist}}_j < \textbf{hist}_j$. Since $||\textbf{hist}||_1 = \left|\left|\overline{\textbf{hist}}\right|\right|_1 = n$, there exists a way of transforming $\overline{\textbf{hist}}$ to **hist** by accumulating all the excess weights of the dimensions in J_1 and assigning it to the dimensions in J_2. The changes in the weight

$$\sum_{j \in J_1} |\overline{\textbf{hist}}_j - \textbf{hist}_j| = \sum_{j \in J_2} |\overline{\textbf{hist}}_j - \textbf{hist}_j| \geq \text{MoV}$$

by the definition of MoV. However, this contradicts the assumption that

$$\left|\left|\textbf{hist} - \overline{\textbf{hist}}\right|\right|_1 = \sum_{j \in (J_1 \cup J_2)} |\overline{\textbf{hist}}_j - \textbf{hist}_j| < 2\text{MoV}.$$

Therefore, for any histogram $\overline{\mathbf{hist}}$ such that $\left|\left|\mathbf{hist} - \overline{\mathbf{hist}}\right|\right|_1 < 2\text{MoV}$, $r(\mathbf{hist}) = r(\overline{\mathbf{hist}})$. □

Then, the $\Pr[\text{correct}]$ property of Lemma 2 follows by combining Proposition 1 and Proposition 2. □

According to Lemma 2, Algorithm 1's $\Pr[\text{correct}] \geq 1 - \frac{m!}{2(\delta \cdot 2^s - 1)}$. In order to achieve $\Pr[\text{correct}]$ of $1-\varepsilon$, constraint $\delta \geq 2^{-s}(\frac{m!}{2\varepsilon} + 1)$ needs to be satisfied. Combining the constraint and the feasible region in Lemma 2, we know that $\delta \in \left[2^{-s}(\frac{m!}{2\varepsilon} + 1),\ \frac{\text{MoV}}{2^{t+1}\pi \cdot (m!)} - 2^{-s}\right)$, which must not be empty, *i.e.*

$$2^{-s}\left(\frac{m!}{2\varepsilon} + 1\right) < \frac{\text{MoV}}{2^{t+1}\pi \cdot (m!)} - 2^{-s}. \tag{2}$$

It's not hard to verify that $s = \sigma(\varepsilon)$ satisfies the constraint in (2). Then, Theorem 2 follows by setting $\delta = 2^{-s}(\frac{m!}{2\varepsilon} + 1)$ and $s = \sigma(\varepsilon)$ for Lemma 2. □

5 COMPARE QUANTUM AND CLASSICAL VOTING

This section compares quantum-accelerated voting with (the best performance of) classical voting algorithms. The classical algorithm is designed according to the idea of sampling (either with or without replacement). At the high level, it uses the randomly sampled votes to estimate the winner. We analyze the runtime and space requirements for classical sampling algorithms and compare them with those of our quantum-accelerated voting algorithm.

When does quantum (may) accelerate voting? Firstly, we provide an intuitive explanation of when quantum would accelerate voting the most. We first think about the cases where classical algorithms (*e.g.*, randomly sampling a subset of votes and using the subset to predict the winner) do not need to be improved or cannot be improved. When the margin of victory $\text{MoV} = \Theta(n)$, classical algorithms are already very fast according to the Chernoff bound, which says the classical algorithms' error rate can be exponentially small in terms of runtime [Bhattacharyya and Dey, 2021]. Another case is when MoV is very small (*e.g.*, $\text{MoV} = \Theta(1)$) where classical algorithms' performance is close to the optimal. In this case, any algorithm has to look into each vote to decide the winner. Since the complexity of counting every vote is $\Theta(n)$, there is not a lot of space for the classical algorithms to be improved.

Between these two extremes is the case where quantum accelerates voting most significantly, for example, when the margin of victory $\text{MoV} = \Theta(n^c)$, where $c \in (0, 1)$ is a constant. In this case, the classical voting would be as slow as $\Omega(\frac{n^2}{\text{MoV}^2})$ for a fixed error rate ϵ. On the other hand, the runtime of the quantum-accelerated voting, $\Theta(\frac{n}{\text{MoV}})$, is quadratically faster. This comparison is also shown in our experiment of $m = 2$ and $\text{MoV} = 1024$ for plurality (the middle column in Figure 2), where the number of voters $n \approx 10^6$ and $\text{MoV} = \sqrt{n}$, *i.e.* the winner gets $\sim 0.2\%$ more votes than the loser.

Theorem 3 establishes a theoretical "complexity lower bound" of any classical voting algorithms (based on sampling with replacement) for many common voting rules. Consequently, sampling-based algorithm with replacement is at least quadratically slower than quantum-accelerated voting. Here, an algorithm being " sampling-based" means the sampling method is the only method for the algorithm to get information about the voting profile P. We say one voting rule reduces to majority voting for two candidates if the voting rule always has the same winner as majority (using whatever tie-breaking method) when $m = 2$. Most commonly used voting rules reduce to majority for two candidates (*e.g.,* any positional scoring rules, STV, Copeland, and maximin, just to name a few).

Theorem 3. *Given any fixed $m \geq 2$, for any $\varepsilon \in (0, 0.5]$, any fast voting algorithm based on sampling with replacement for the voting rule such that reduces to majority voting for two candidates requires at least $\Omega\left(\frac{n^2 \cdot \left(\frac{1}{2} - \varepsilon\right)^2}{\text{MoV}^2}\right)$ (expected) runtime and at least $\Omega\left(\log\left(\frac{n^2 \cdot \left(\frac{1}{2} - \varepsilon\right)^2}{\text{MoV}^2}\right)\right)$ (expected) space to achieve $\Pr[\text{correct}] \geq 1 - \varepsilon$ in the worst case.*

When ε is a constant, the bound in Theorem 3 becomes $\Omega\left(\frac{n^2}{\text{MoV}^2}\right)$, which is tight according to Bhattacharyya and Dey [2021]. We note Theorem 3 is more general than the bounds in Bar-Yossef et al. [2001] (require a lower bound for ε) and Canetti et al. [1995] (only holds for small-scale samplings).

Appendix B.2 shows that sampling without replacement has the same asymptotic manner as sampling with replacement when the number of samples $T = o(\sqrt{n})$, which is the setting of fast majority voting in many application scenarios.

Proof of Theorem 3. **Step 1.** A lower bound of runtime and space requirement for any fast majority voting algorithm for two candidates (Lemma 3).

Lemma 3. *For any $\varepsilon \in (0, 0.5]$, any fast (2-candidate) majority voting algorithm based on sampling with replacement requires at least $\Omega\left(\frac{n^2 \cdot \left(\frac{1}{2} - \varepsilon\right)^2}{\text{MoV}^2}\right)$ (expected) runtime and at least $\Omega\left(\log\left(\frac{n^2 \cdot \left(\frac{1}{2} - \varepsilon\right)^2}{\text{MoV}^2}\right)\right)$ (expected) space to achieve $\Pr[\text{correct}] \geq 1 - \varepsilon$.*

The proof of Lemma 3 can be found in Appendix B.1. Note that the ε part of the bound in Lemma 3 is not tight when ε

is close to 0. This is because our bound is given by the information limit, which cannot be reached when reconstructing a limited number of bits. In particular, our problem only reconstructs 1-bit information (which candidate wins out of the two candidates).

Step 2. The lower bound for the fast voting algorithm for $m > 2$ candidates cannot be smaller than the lower bound for the two-candidate case.

Suppose Theorem 3 is not true, and there exists a fast voting algorithm A for voting rule r such that for any profile P, A has a runtime of $o\left(\frac{n^2\cdot\left(\frac{1}{2}-\varepsilon\right)^2}{\mathrm{MoV}^2}\right)$ and achieves $\Pr[A(P) \text{ is correct}] \geq 1-\varepsilon$ (the reasoning for space complexity will be similar). And suppose A takes a profile P of m candidates $a, b, c_3, \cdots, c_m$ as the input and sample votes from P with replacement. We show that there exists a fast algorithm for majority voting A' with the same runtime and accuracy. Let P' be a voting profile for two candidates a and b, and let n_a and n_b be the number of votes for a and for b respectively. We construct algorithm A' as follows:

A' is almost the same as A except for the different input and sampling. A' takes a profile P' of two candidates a and b as the input. Whenever there needs a sample, A samples a vote from P'. If the voter votes for a, then A' convert it to $a \succ b \succ c_3 \succ \cdots \succ c_m$; and if it votes for b, then A' convert it to $b \succ a \succ c_3 \succ \cdots \succ c_m$. If the winner calculated is neither a nor b, A' will set a to the winner. Except for the sampling, A' running on P' is equivalent to A running on the following profile P and set the same winner as A does: there are n_a votes of $a \succ b \succ c_3 \succ \cdots \succ c_m$ and n_b votes of $b \succ a \succ c_3 \succ \cdots \succ c_m$.

If a is the winner in P', then $n_a \geq n_b$, and the margin of victory for the majority vote is $\mathrm{MoV}' = \frac{1}{2}(n_a - n_b)$. Then it is not hard to verify that when r is one of the rules mentioned in the statement, a is also the winner in P, and the margin of victory in P is $\mathrm{MoV} = \mathrm{MoV}'$. Therefore, A set the winner as A with probability at least $1-\varepsilon$ under runtime $o\left(\frac{n^2\cdot\left(\frac{1}{2}-\varepsilon\right)^2}{\mathrm{MoV}^2}\right)$. Then A', with the same operation to A, will also set the winner as A with probability at least $1-\varepsilon$ under the same runtime. Similarly, if b is the winner, A' can output the correct winner with probability at least $1-\varepsilon$ under the same runtime. Therefore, A' is a fast majority voting algorithm based on sampling with a replacement that can achieve $\Pr[\text{correct}] \geq 1-\varepsilon$ under runtime $o\left(\frac{n^2\cdot\left(\frac{1}{2}-\varepsilon\right)^2}{\mathrm{MoV}^2}\right)$, which contradicts with Lemma 3. □

6 EXPERIMENTAL RESULTS

Basic settings. We numerically compare the proposed quantum-accelerated voting (Algorithm 1) with a fast classical voting based on sampling with replacement (Algorithm 2). We set the number of samples T in Algorithm 2 to be $K \cdot 2^s$, where s and K are the parameters of Algorithm 1. By doing this, the runtime of both algorithms is $\Theta(K \cdot 2^s)$. We set the number of voters $n = 2^{20} \approx 10^6$, which is at a similar order of magnitude as the number of voters in a typical state of the United States. For example, the number of registered voters in New Hampshire is 1,009,004 $\approx 10^6$ [Independent Voter Project, 2020]. We compare Algorithm 1 and Algorithm 2 on four widely-used voting rules, which are two scoring rules (plurality and Borda), a pairwise rule (Copeland), and an elimination-based rule (STV). The formal definitions of the four rules can be found in Section 2. Note that all four rules are covered by Theorem 3. The implementation details can be found in Appendix C.1.

Algorithm 2: Fast Classical Voting Algorithm

1: **Inputs:** n voters' votes $\boldsymbol{V}_0, \cdots, \boldsymbol{V}_{n-1}$, a voting rule r, number of samples T
2: Sample T votes uniformly at random with replacement
3: Build the histogram $\widehat{\textbf{hist}}$ of the sampled votes.
4: **Output** $r(\widehat{\textbf{hist}})$ as the winner.

Detailed settings. We use random tie-breaking to break ties for all voting rules above. For example, if c_1 and c_2 are tied, each of them will win with $1/2$ probability. We set the number of candidates $m \in \{2, 4\}$ for all rules. In all experiments of this paper, we estimate the probability of outputting the correct winner ($\Pr[\text{correct}]$) by the frequency of observing the correct winner in 10^5 independent trails. We set $K \in \{1, 3, 5\}$ to avoid ties in the classical part of quantum-accelerated voting algorithm. We set $\mathrm{MoV} \in \{256, 512, 1024, 2048, 4096\}$, or equivalently, $\mathrm{MoV} \in \{n^{0.4}, n^{0.45}, n^{0.5}, n^{0.55}, n^{0.6}\}$. Since all four rules reduce to the majority rule when there are only two candidates, we consider the following profile as for $m = 2$.

$$\begin{cases} (n/2 + \mathrm{MoV}) \text{ votes for } c_1 \succ c_2 \\ (n/2 - \mathrm{MoV}) \text{ votes for } c_2 \succ c_1 \end{cases}.$$

It's easy to check that its margin of victory is MoV under all four rules above. Figure 2 compares quantum-accelerated voting with a classical fast voting algorithm (Algorithm 2). The horizontal axis of Figure 2, $\log_2(K \cdot 2^s)$, can be seen as the logarithm of the algorithms' runtime. For all curves, we set $s = 4, 5, \cdots, 16$ for the twelve points from left to right, respectively. The detailed settings and experimental results of $m = 4$ are shown in Appendix C.2.

Observations. First, with the same order of runtime, the quantum-accelerated voting algorithm has better $\Pr[\text{correct}]$ than classical fast voting no matter under which setting. For example, in Figure 2, $\mathrm{MoV} = 1024$, $K = 1$, and $s = 14$, the quantum-accelerated voting algorithm outputs the correct winner almost for certain. However, the classical algorithm only has $\sim 60\%$ probability of outputting

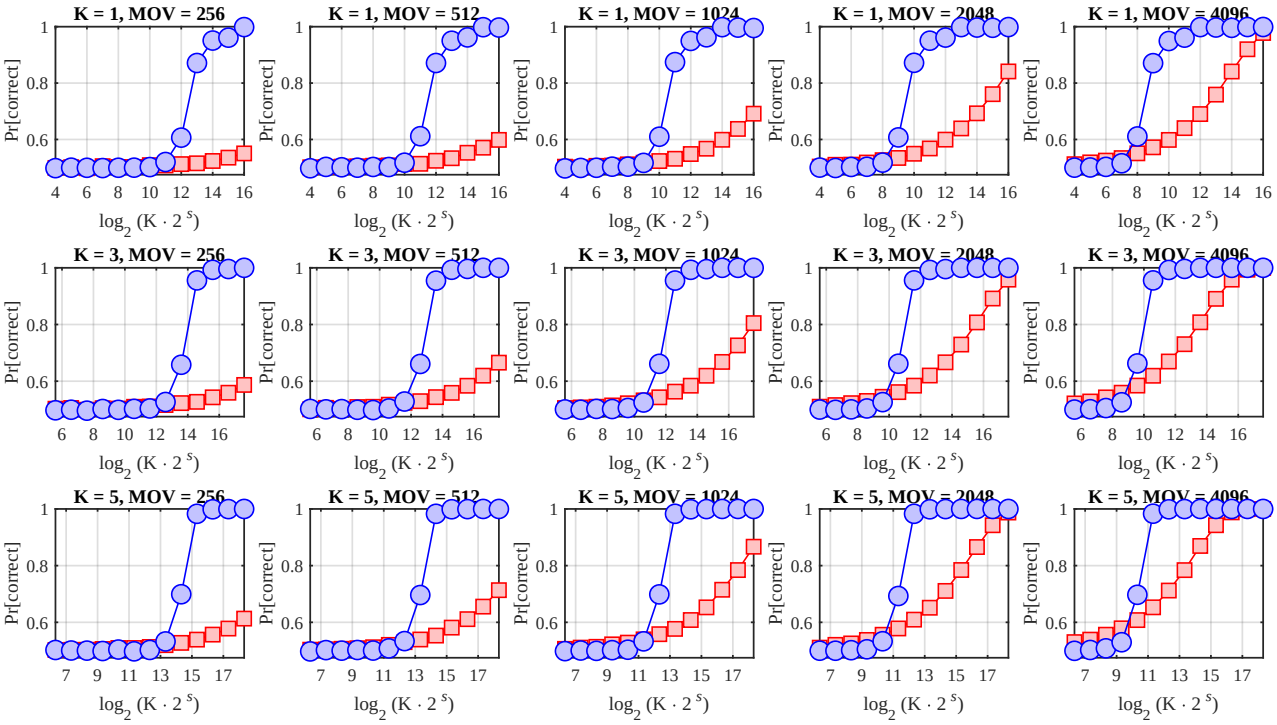

Figure 2: Compare quantum-accelerated voting (blue circles) with classical fast voting (red squares) for plurality, Borda, Copeland, and STV when $m = 2$. The horizontal axis can be seen as the logarithm of the algorithms' runtime.

the correct winner. Second, quantum-accelerated voting requires much less runtime to achieve the same $\Pr$ [correct]. For example, in Figure 2, MoV $= 1024$, and $K = 1$, to achieve $\sim 90\%$ $\Pr$[correct], the quantum algorithm requires 2^{10} runtime. In comparison, the classical algorithm with 2^{16} runtime can only achieve $\sim 70\%$ $\Pr$[correct]. Both observations match our theoretical result: the proposed algorithm is quadratically faster than any classical algorithm.

7 FURTHER ACCELERATING QUANTUM VOTING

Can quantum-accelerated voting algorithm be further accelerated? This section discusses some heuristics.

Pre-sampling. One way to improve the average performance of the quantum-accelerated voting algorithm is to pre-sample a small subset of the votes. For example, in the majority vote for binary candidates, if the pre-sample votes indicate an almost irreversible win of a candidate, then we directly announce the winner and skip the quantum computing. By carefully setting the pre-sampling size and the skipping thresholds, we may improve the average run-time while keeping high $\Pr$ [correct].

Sampling + quantum. Another natural idea is to apply the quantum-accelerated voting algorithm on a sampled subset of the votes. Sampling decreases the number of votes, so the quantum circuit consumes fewer bits and operators, which reduces the time and space cost. However, such improvement sacrifices $\Pr$ [correct], as both sampling and quantum computing has a probability to make a mistake. It is still unclear if such a sampling-quantum algorithm would be faster to achieve the same level of $\Pr$ [correct].

8 CONCLUSIONS AND FUTURE WORKS

In this paper, we took the first step in using quantum computation to accelerate voting. Our proposed quantum-accelerated voting algorithm can quadratically accelerate various widely-used voting rules and may potentially improve the efficiency of voting in large-scale and/or high-frequency decision-making scenarios.

An extension of this paper is to further accelerate the proposed algorithm by combining existing acceleration techniques in classical fast voting algorithms. Since voting is widely used in artificial intelligence, it would also be interesting to apply the proposed methods to accelerate the algorithms in other fields (*e.g.,* search engine, crowdsourcing, database management, and blockchain governance).

As real quantum computers may come across quantum errors caused by quantum interference and environmental effects, another interesting extension is to test the robustness of the proposed algorithm. For example, testing its performance on real quantum computers and/or running experiments with quantum errors taken into account.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. Lirong Xia is supported by NSF #1453542 and a gift fund from Google.

References

Akshay Ajagekar and Fengqi You. Quantum computing assisted deep learning for fault detection and diagnosis in industrial process systems. *Computers & Chemical Engineering*, 143:107119, 2020.

Akshay Ajagekar and Fengqi You. Quantum computing based hybrid deep learning for fault diagnosis in electrical power systems. *Applied Energy*, 303:117628, 2021.

Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Sampling algorithms: Lower bounds and applications. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, page 266–275, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133499.

John Bartholdi, Craig A Tovey, and Michael A Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and welfare*, 6:157–165, 1989.

Francesco Belardinelli and Umberto Grandi. Social choice methods for database aggregation. In *17th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2019)*, volume 297, pages 50–67, 2019.

Marcello Benedetti, John Realpe-Gómez, Rupak Biswas, and Alejandro Perdomo-Ortiz. Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning. *Physical Review A*, 94(2):022308, 2016.

Arnab Bhattacharyya and Palash Dey. Predicting winner and estimating margin of victory in elections using sampling. *Artificial Intelligence*, 296:103476, 2021.

Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, 2016.

Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum counting. In *Automata, Languages and Programming: 25th International Colloquium, ICALP'98 Aalborg, Denmark, July 13–17, 1998 Proceedings 25*, pages 820–831. Springer, 1998.

Markus Brill. Interactive democracy. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1183–1187, 2018.

Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53(1):17–25, 1995.

Vincent Conitzer, Matthew Rognlie, and Lirong Xia. Preference functions that score rankings and maximum likelihood estimation. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622, 2001.

Bailey Flanigan, Paul Gölz, Anupam Gupta, and Ariel D Procaccia. Neutralizing self-selection bias in sampling for sortition. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 6528–6539, 2020.

Bailey Flanigan, Paul Gölz, Anupam Gupta, Brett Hennig, and Ariel D Procaccia. Fair algorithms for selecting citizens' assemblies. *Nature*, 596(7873):548–552, 2021.

Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Architectures for a quantum random access memory. *Physical Review A*, 78(5):052310, 2008a.

Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008b.

Davide Grossi. Social choice around the block: On the computational social choice of blockchain. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1788–1793, 2022.

Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

Tad Hogg and Dmitriy Portnov. Quantum optimization. *Information Sciences*, 128(3-4):181–197, 2000.

Christopher M Homan and Lane A Hemaspaandra. Guarantees for the success frequency of an algorithm for finding dodgson-election winners. *Journal of Heuristics*, 15:403–423, 2009.

2020 Independent Voter Project. New hampshire voter statistics, 2020.

Emil T Khabiboulline, Juspreet Singh Sandhu, Marco Ugo Gambetta, Mikhail D Lukin, and Johannes Borregaard. Efficient quantum voting with information-theoretic security, 2021.

Shiguo Lian and Yan Zhang. *Handbook of research on secure multimedia distribution*. IGI Global, Hershey, PA, 2009. ISBN 1605662623.

Ao Liu, Yun Lu, Lirong Xia, and Vassilis Zikas. How private are commonly-used voting rules? In *Conference on Uncertainty in Artificial Intelligence*, pages 629–638. PMLR, 2020.

Pia Mancini. Why it is time to redesign our political system. *European View*, 14(1):69–75, 2015.

Andrew Mao, Ariel Procaccia, and Yiling Chen. Better human computation through principled voting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, pages 1142–1148, 2013.

John C McCabe-Dansted, Geoffrey Pritchard, and Arkadii Slinko. Approximability of dodgson's rule. *Social Choice and Welfare*, 31(2):311–330, 2008.

Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

Daniel K Park, Francesco Petruccione, and June-Koo Kevin Rhee. Circuit-based quantum random access memory for classical data. *Scientific reports*, 9(1):3949, 2019.

Joan Alfina Vaccaro, Joseph Spring, and Anthony Chefles. Quantum protocols for anonymous voting and surveying. *Physical Review A*, 75(1):012333, 2007.

Toby Walsh and Lirong Xia. Lot-based voting rules. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 603–610, 2012.

Jun Wang, Sujoy Sikdar, Tyler Shepherd, Zhibing Zhao, Chunheng Jiang, and Lirong Xia. Practical algorithms for multi-stage voting rules with parallel universes tiebreaking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2189–2196, 2019.

Lirong Xia and Weiqiang Zheng. Beyond the worst case: Semi-random complexity analysis of winner determination. In *Web and Internet Economics: 18th International Conference, WINE 2022, Troy, NY, USA, December 12–15, 2022*, pages 330–347. Springer, 2022.

Peng Xue and Xin Zhang. A simple quantum voting scheme with multi-qubit entanglement. *Scientific reports*, 7(1): 1–4, 2017.