Accelerating Voting by Quantum Computation

Abstract

Studying the computational complexity of determining winners under voting rules and designing fast algorithms are classical and fundamental questions in computational social choice. In this paper, we accelerate voting by leveraging quantum computing. We propose a quantum voting algorithm that can be applied to any anonymous voting rule. We further show that our algorithm can be quadratically faster than any classical sampling algorithm under a wide range of common voting rules, including plurality, Borda, Copeland, and STV. Precisely, our quantum voting algorithm achieves an accuracy of at least $1-\varepsilon$ with runtime $\Theta\left(\frac{n\cdot\log(1/\varepsilon)}{\text{MoV}}\right)$, where n is the number of votes and MoV is *margin of victory*, the smallest number of voters to change the winner. On the other hand, any classical voting algorithm based on sampling a subset of voting achieves the same accuracy with runtime $\Theta\left(\frac{n^2\cdot\log(1/\varepsilon)}{\text{MoV}^2}\right)$ [Bhattacharyya and Dey, 2021]. Our theoretical results are supported by experiments under the plurality and Borda rule.

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 engine [Dwork et al., 2001], crowdsourcing [Mao et al., 2013], database management [Belardinelli and Grandi, 2019], and blockchain governance [Grossi, 2021], 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, perhaps at the (small) cost of accuracy. In fact, the study of computational complexity and algorithmic aspects of voting rules has been a key topic of the field 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. Bhattacharyya and Dey [2021] showed that the runtime of the 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 voting algorithm where a small probability of "errors" are allowed, which outperforms the classical sampling voting algorithms. Our contributions are three-fold. First, we propose the quantum voting algorithm (Algorithm 1). Our algorithm leverages classical 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 sampling algorithm for many com-

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.

	Runtime	Space Requirement
Quantum (Theorem. 1)	$\Theta\left(\frac{n\cdot\log(1/\varepsilon)}{\text{MoV}}\right)$	$\Theta\left(\log(\frac{n\cdot\log(1/\varepsilon)}{\text{MoV}})\right)$
Classical [Bhattacharyya and Dey, 2021]	$\Theta\left(\frac{n^2\cdot\log(1/\varepsilon)}{\text{MoV}^2}\right)$	$\Theta\left(\log(\frac{n^2\cdot\log(1/\varepsilon)}{\text{MoV}^2})\right)$

Table 1: Summary for the theoretical results, where n is the number of votes, and MoV means the margin of victory, the smallest number of voters needed to change the winner. All results in the table assume algorithms output the correct winner with no less than $1-\varepsilon$ probability.

mon voting rules including plurality, positional scoring rule, Copeland, and STV (see Table 1). Third, we experimentally verified our theoretical results on plurality and Borda rule (Section 6). In Section 7, we provide heuristics that may further improve the performance of quantum voting.

Our quantum algorithm accelerates voting most significantly in the case where the margin of victory is sub-linear to n, i.e. 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. And the ratio $\frac{n}{\text{MoV}} = \Theta(n^{1-c})$ implies that the algorithm is sub-linear. Also see Section 5 for a detailed explanation of when quantum computation accelerates.

Related works and discussions. To the best of our knowledge, none of the literature has used 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] proposed an "all-in-one" quantum voting protocol, which focuses on achieving anonymity without losing security guarantees. However, all of the above approaches require $\Omega(n)$ quantum communication cost, which means their methods take $\Omega(n)$ time.

There is a large literature on efficient 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 some semi-random models [Xia and Zheng, 2022]. Most of these work focus on provide efficient algorithms NP-hard winner determination problem, and few consider fast, randomized voting.

2 PRELIMINARIES

Voting. In a voting scenario, $n > 1$ voters cast their votes on $m > 1$ candidates. A vote $\boldsymbol{V}_i$ is a full-ranking (linear order) over candidates that represents a voter's preference towards the candidates. Since there are $m!$ types of full-rankings for m candidates, a vote can be represented as a $m!$-dimensional unit vector. Here, 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 agents' rankings. A voting rule r is a mapping from the profile P to the winner among m candidates.

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 histogram **hist** is a $m!$ dimension vector that records the number of each type of ranking in the profile. We use $r(\textbf{hist})$ to denote the winner under anonymous voting rule r and a profile with histogram **hist**. A voting rule satisfies canceling out if the winner does not change after adding one copy of each ranking to the profile. Many common voting rules satisfy anonymity and canceling-out, such as plurality, Borda, and STV.

Margin of victory (MoV) describes the smallest number k such that k voters can change the winner by voting differently.

For the quantum computation, we introduce the technique of *quantum counting* [Brassard et al., 1998] which we applied in our quantum voting algorithm. An introduction to quantum computing and implementation can be found in Appendix A.1. [1]

Quantum Counting Algorithm [Brassard et al., 1998]. Quantum counting algorithm is a quantum algorithm that 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

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

counting circuit uses t quantum bits to encode the function and s quantum bits to calculate and record the output. More specifically, suppose there are exactly n_1 solutions of f. The quantum counting circuit with $t+s$ quantum bits outputs a binary decimal $\hat{\varphi} = 0.b_1 b_2 \cdots b_s$ that is an estimation of $\varphi = \arcsin\left(\sqrt{n_1 \cdot 2^{-t}}\right)/\pi$. For example, binary decimal 0.011 represents $(2^{-2} + 2^{-3}) = 3/8$. We present a useful error bound for quantum counting.

Lemma 1 (Error bound for quantum counting, Inequality (5.34) in [Nielsen and Chuang, 2010])**.** *For any $\delta > 2^{-s}$,*

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

The number of quantum bits s affects both the precision and the runtime of the quantum counting algorithm. A large s leads to a more precise estimation of the result while being time-costly, as the runtime of the quantum counting algorithm is $\Theta(2^s)$. The detailed implementation of the quantum counting circuit and the reasoning behind why it accelerates voting can be found in Appendix A.2.

Applying Quantum Counting to Voting. We apply the quantum counting algorithm to count the histogram of a profile. For a type j, we can count the number of j-th type of votes by setting the following binary function:

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

The number of x such that $f_j(x) = 1$ is exactly 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. We assume that the information of f_j functions is stored in the quantum RAM and can be efficiently encoded into quantum circuits [Giovannetti et al., 2008b,a, Park et al., 2019].

3 QUANTUM VOTING ALGORITHM

Formal definition of quantum voting. We formally define quantum voting in Algorithm 1. In classical voting, votes are usually sent to an "aggregator", who is responsible for aggregating the votes and announcing the winner. Our quantum voting follows a similar procedure, where the "aggregator" is the quantum counting algorithm.

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 the winner. Then, the "aggregator" announces the candidate who wins in the most rounds as the winner of the voting. A tie-breaking rule is applied when there are multiple candidates with the most winning rounds. Next, we will introduce the functionality of each step of Algorithm 1 in detail.

Algorithm 1: Quantum 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 $r(\widehat{\mathbf{hist}})$ as the winner of the k-th iteration
end for
Announce the candidate that wins in the largest number of iterations

Construct binary functions. For each type j of preferences, we construct a binary function f_j where all j-th type votes in the profile are solutions. Formally, $f_j : \{0, 1, \cdots, 2^t - 1\} \to \{0, 1\}$ is defined as

$$f_j(x) = \begin{cases} 1 & \text{if voter } x\text{'s vote is } j\text{-th type} \\ 0 & \text{otherwise} \end{cases},$$

where $t = 2^{\lceil \log n \rceil}$ is the number of quamtum bits to encode n, and voters an numbered from 0 to $(n-1)$. For $x = n, \cdots, 2^t - 1$, we just set $f_j(x) = 0$. Then, the number of x such that $f_j(x) = 1$ is exactly the number of votes of j-th type, i.e. $\mathbf{hist}_j$.

Count the histogram. For each type j, a quantum counting circuit is constructed and counts the solution of f_j. The output of the counting circuit is the estimation $0.b_1 b_2 \cdots b_s$ of $\varphi = \arcsin\left(\sqrt{\mathbf{hist}_j \cdot 2^{-t}}\right)/\pi$. Therefore, we set $\widehat{\mathbf{hist}}_j$ as $2^t \cdot \sin^2(\pi \cdot 0.b_1 \cdots b_s)$. By enumerating all types j, we get the estimation $\widehat{\mathbf{hist}}$ of the histogram **hist**.

Decide the winner. The quantum counting procedure runs for K rounds. In each round, the winner is computed on the estimated histogram $\widehat{\mathbf{hist}}$. The overall winner of the algorithm is the candidate that wins in the most rounds.

4 THEORETICAL ANALYSIS OF FAST QUANTUM VOTING

In Section 3, we proposed our quantum voting algorithm (Algorithm 1). In this section, we provide theoretical guarantees about the accuracy (probability of outputting

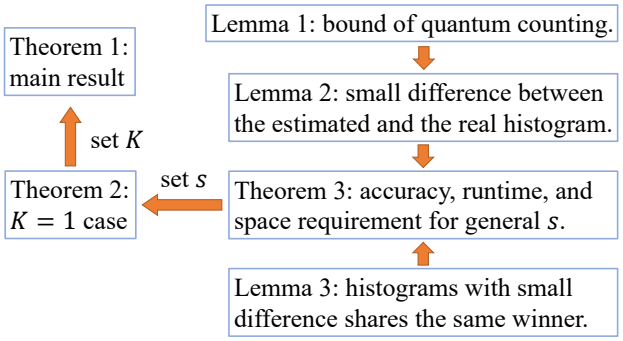


Figure 1: Logical chain of theorems and technical lemmas. For example, the arrow from Theorem 2 to Theorem 1 represents that Theorem 1 is proved by applying Theorem 2.

the correct winner), runtime, and space requirements of our algorithm.

Our bound for the quantum voting algorithm depends on three factors: the number of votes n, the margin of victory MoV, and the error probability ε. We regard the number of candidates m as fixed in the analysis. Figure 1 shows how the theorems are logically founded.

Theorem 1 (Theoretical guarantee of quantum voting)**.** *For arbitrary constant $\varepsilon \in (0,1)$, quantum voting (Algorithm 1) has the following three properties when parameters $K = 24\ln(1/\varepsilon)$ and $s = 2 + \lceil t + \log(2(m!)+2) + \log(\pi(m!)) - \log(MoV)\rceil$.*
1. It outputs the correct voting outcome with at least $1-\varepsilon$ probability.
2. Its runtime is $\Theta\left(\frac{n\log(1/\varepsilon)}{MoV}\right)$.
3. Its space requirement is $\Theta\left(\log(\frac{n\log(1/\varepsilon)}{MoV})\right)$.

Proof. Theorem 1 holds by applying Theorem 2 with $\varepsilon = \frac{1}{4}$. By setting s as in the statement, the algorithm outputs the correct result in each round with a probability of at least $p = \frac{3}{4}$. Then by Chernoff bound, the probability that the correct result is chosen for at least $K/2$ rounds (which guarantees to be the most rounds) is at least

$$1-\varepsilon \geq 1-\exp\left(-(1-\frac{1}{2p})^2\cdot K\cdot p/2\right),$$

which is equivalent to

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

□

Theorem 2 (Theoretical guarantee of quantum voting when $K = 1$)**.** *For arbitrary constant $\varepsilon \in (0,1)$, quantum voting (Algorithm 1) has the following three properties when $K = 1$ and $s = 2 + \lceil t + \log(\frac{m!}{2\varepsilon}+2) + \log(\pi(m!)) - \log(MoV)\rceil$.*
1. It outputs the correct voting outcome with at least $1-\varepsilon$ probability.
2. Its runtime is $\Theta\left(\frac{n}{\varepsilon\cdot MoV}\right)$.
3. Its space requirement is $\Theta\left(\log(\frac{n}{\varepsilon\cdot MoV})\right)$.

Proof. The proof of Theorem 2 follows by setting a proper parameter s for Theorem 3. From Theorem 3, for a given s, the accuracy of Algorithm 1 is at least $1-\frac{m!}{2(\delta\cdot 2^s-1)}$. In order to achieve the accuracy of $1-\varepsilon$, constraint $\delta \geq 2^{-s}(\frac{m!}{2\varepsilon}+1)$ needs to be satisfied. Then, s needs to be large enough to guarantee that the feasible region of δ is not empty, i.e.

$$2^{-s}(\frac{m!}{2\varepsilon}+1) < \frac{\mathrm{MoV}}{2^{t+1}\pi(m!)} - 2^{-s}.$$

It's not hard to verify that s in the statement of the theorem satisfies this constraint. Then, by setting $\delta = 2^{-s}(\frac{m!}{2\varepsilon}+1)$, we get that the accuracy is at least $1-\varepsilon$. □

Theorem 3. *For any $s \geq 2$, quantum voting (Algorithm 1) with $K = 1$ has the following three properties.*
1. For all $\delta \in (2^{-s}, \frac{MoV}{2^{t+1}\pi(m!)} - 2^{-s})$, it outputs the correct winner with at least $1-\frac{m!}{2(\delta\cdot 2^s-1)}$ probability.
2. Its runtime is $\Theta(2^s)$
3. Its space requirement is $\Theta(\log(n)+s)$.

Remark 1. *Property 1 has a guarantee of accuracy only when the feasible region of δ is non-empty, which sets a constraint of s.*

The proof of the accuracy is obtained by applying two lemmas. Lemma 2 guarantees that for all dimension j, the difference of $\mathbf{hist}_j$ and the estimation $\widehat{\mathbf{hist}}_j$ is bounded by $\mathrm{MoV}/(m!)$ with at least $1-\frac{m!}{2(\delta\cdot 2^s-1)}$ probability. Lemma 3 guarantees that any histogram in the neighborhood of **hist** shares the same winner with **hist**. Therefore, with at least $1-\frac{m!}{2(\delta\cdot 2^s-1)}$ probability, $\widehat{\mathbf{hist}}$ leads to the correct winner. The runtime and the space requirement come from the quantum counting algorithm. Recall that a quantum counting algorithm has a runtime of $\Theta(2^s)$. [2] And since it requires $t+s$ quantum bits where $t = \lceil \log n \rceil$, the space requirement is $\Theta(\log(n)+s)$.

In the following lemma, let $\overline{\mathbf{hist}}_j$ be the histogram by adding the same fraction of each ranking to $\widehat{\mathbf{hist}}$ so that the sum of the histogram is n: for all dimension j, $\overline{\mathbf{hist}}_j = \widehat{\mathbf{hist}}_j + (||\mathbf{hist}||_1 - ||\widehat{\mathbf{hist}}||_1)/(m!)$. (Recall that $||\mathbf{hist}||_1 = n$ is known.) By the assumption of canceling-out, we know that $r(\overline{\mathbf{hist}}) = r(\widehat{\mathbf{hist}})$. Therefore, it is suffice to show that the difference between $\mathbf{hist}_j$ and $\overline{\mathbf{hist}}_j$ is bounded.

[2] As the anonymous rule r computes the winner based on a $m!$ dimension histogram, its runtime will only depend on the number of candidates m, which is $\Theta(1)$ in our analysis.

Lemma 2. *For all $\delta \in (2^{-s}, \frac{MoV}{2^{t+1}\pi(m!)} - 2^{-s})$, the probability that $|\overline{\boldsymbol{hist}}_j - \boldsymbol{hist}_j| < \frac{MoV}{m!}$ for all dimension j is at least $1 - \frac{m!}{2(\delta \cdot 2^s - 1)}$.*

Proof. In this proof, we fix an arbitrary pair of s and δ satisfying the condition. By applying the sum of probability on Lemma 1 for all dimension j, we know that the probability that $|\hat{\varphi}_j - \varphi_j| < 2^{-s} + \delta$ for every dimension j 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 $\frac{dh(x)}{dx} = 2^t \pi \sin(2\pi x) \leq 2^t \pi$. Therefore,

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

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

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

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

Lemma 3. *For any histogram $\widehat{\boldsymbol{hist}}$ such that $||\boldsymbol{hist}||_1 = \left|\left|\widehat{\boldsymbol{hist}}\right|\right|_1$ and $\left|\left|\boldsymbol{hist} - \widehat{\boldsymbol{hist}}\right|\right|_1 < 2MoV$, $r(\boldsymbol{hist}) = r(\widehat{\boldsymbol{hist}})$.*

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 $\widehat{\textbf{hist}}$ such that $\left|\left|\textbf{hist} - \widehat{\textbf{hist}}\right|\right|_1 < 2\text{MoV}$ and $r(\textbf{hist}) \neq r(\widehat{\textbf{hist}})$. Let J_1 be the set of dimension j such that $\widehat{\textbf{hist}}_j > \textbf{hist}_j$, and J_2 be the set of j such that $\widehat{\textbf{hist}}_j < \textbf{hist}_j$. Since $||\textbf{hist}||_1 = \left|\left|\widehat{\textbf{hist}}\right|\right|_1 = n$, there exists a way of transferring $\widehat{\textbf{hist}}$ 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

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

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

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

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

5 COMPARE QUANTUM AND CLASSICAL VOTING

In this section, we compare quantum voting (Algorithm 1) with classical fast 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 histogram of) randomly sampled votes to estimate the winner. We analyze the runtime and space requirement for classical sampling algorithms and compare them with those of our quantum voting algorithm.

When does quantum voting (may) accelerate? Firstly, we provide an intuitive explanation of when quantum voting would accelerate 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 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.*, MoV $= \Theta(1)$) where classical algorithms' performance is close to the optimal. In this case, any algorithms have 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 voting accelerates most significantly, for example, when the margin of victory MoV $= \Theta(n^c)$, where $c \in (0, 1)$ is a constant. In this case, the classical voting would be as slow as $\Omega(n)$ when $c \leq \frac{1}{2}$, and the acceleration of $\Theta(\frac{n}{\text{MoV}})$ from

quantum voting is significant. On the other hand, the runtime of the quantum voting $\Theta(\frac{n\log(1/\varepsilon)}{\text{MoV}})$ is still sub-linear and satisfies the requirement of "fast voting". For example, in the experiment of $m = 2$ and $\text{MoV} = 10$ in our experimental verification (See the right column in Figure 2), the number of voters $n \approx 10^6$ and $\text{MoV} = \sqrt{n}$. In this example, the winner only got $\sim 0.2\%$ more votes than the loser.

Our comparison between classical and quantum voting algorithms also lies in the case of $\text{MoV} = \Theta(n^c)$.

Runtime. The runtime of a sampling algorithm is the number of sample it needs to achieve the given level of correctness. Bhattacharyya and Dey [2021] proposes a generic sample bound of sampling algorithms that hold for a wide range of voting rules.

Theorem 4 (Bhattacharyya and Dey [2021])**.** *Given an election where the MoV is at least αn, where α can be a function of n, the sample needed to determine the winner of the election with probability at least $1-\varepsilon$ is $\Theta\left(\frac{\log(1/\varepsilon)}{\alpha^2}\right)$ for plurality, approval, scoring rules, maximin, Copeland, Bucklin, plurality with runoff, and STV voting rules.*

The lower bound in Theorem 4 derives from the lower bound of samples to distinguish two distributions [Canetti et al., 1995], and the upper bound is proved by Chernoff bound. Given $\text{MoV} = \Theta(n^c)$ for some $c \in (0,1)$ we set the parameter $\alpha = \frac{\text{MoV}}{n} = \Theta(n^{c-1})$ and get the following bound of runtime characterized by n and MoV.

Corollary 1. *For all $c \in (0,1)$ and $\text{MoV} = \Theta(n^c)$, the runtime of any sampling algorithm with accuracy at least $1-\varepsilon$ is $\Theta\left(\frac{n^2\log(1/\varepsilon)}{\text{MoV}^2}\right)$ for plurality, approval, scoring rules, maximin, Copeland, Bucklin, plurality with runoff, and STV voting rules.*

Space Requirement. Consider a sampling algorithm with T samples. An algorithm needs to store an integer T_j for each j which denotes the number type j preferences in the votes. For either average-case analysis or worst-case analysis, storing all the T_j requires $\Theta(\log T)$ bits. Therefore, given the runtime bound in Corollary 1, we know the space requirement of any sampling algorithm with accuracy $1-\varepsilon$ is $\Theta\left(\log\left(\frac{n^2\log(1/\varepsilon)}{\text{MoV}^2}\right)\right)$.

Recall that our quantum algorithm (Algorithm 1) has a runtime of $\Theta\left(\frac{n\log(1/\varepsilon)}{\text{MoV}}\right)$ and space requirement $\Theta\left(\log\left(\frac{n\log(1/\varepsilon)}{\text{MoV}}\right)\right)$. Comparing the complexities, our quantum algorithm is quadratically faster than any classical sampling algorithm.

6 EXPERIMENTAL RESULTS

Basic settings. We numerically compare the proposed quantum voting (Algorithm 1) with a classical voting (Algorithm 2), whose general idea is sampling without replacement. 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 each 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 quantum voting and classical voting on two widely-used voting rules, plurality and Borda, which are formally defined as follows.

Algorithm 2: Classical Sampling 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 without replacement
3: Build the histogram $\widehat{\textbf{hist}}$ of the sampled votes.
4: Set $r(\widehat{\textbf{hist}})$ as the winner.

Plurality. The plurality rule counts the top-ranked candidate of each vote. The candidate ranked top in the most votes is announced as the winner. For plurality, we set the number of candidates $m = 2$ (Figure 2) and $m = 4$ (Figure 3).

Borda. Borda rule computes a Borda score for each candidate. A candidate ranked i-th in a vote gains a score of $m - i - 1$ from that vote. For example, the Borda scores for a 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. The candidate with the largest Borda score is announced as the winner. For Borda, we also set the number of candidates $m = 2$ (Figure 7 in Appendix B) and $m = 4$ (Figure 8 in Appendix B).

Implementation details. For any margin of victory MoV, we set the profile for plurality rule as

$$\begin{cases} \frac{n+(m!-2)\text{MoV}}{m!} \text{ voters vote for } c_1 \succ \cdots \succ c_m \\ \frac{n-2\text{MoV}}{m!} \text{ voters vote for each other type of rankings} \end{cases}.$$

For Borda, we let $d = \frac{4\text{MoV}}{(m-2)!\cdot m}$ and set the profile as

$$\begin{cases} \frac{n+(m-1)d}{m!} \text{ voters vote for each type of votes such that } c_1 \text{ is top-ranked} \\ \frac{n-d}{m!} \text{ voters vote for each other kind of rankings} \end{cases}.$$

It's easy to check that the margin of victory for both profiles is MoV. In all experiments of this paper, we first draw 10^5 independent trails for the sampled profile and calculate the winner for each trail. Then, we estimate the probability of outputting the correct winner ($\Pr[\text{correct}]$) by the frequency of observing the correct winner in the trails. The horizontal axis, $\log_2(K \cdot 2^s)$ can be seen as the logarithm

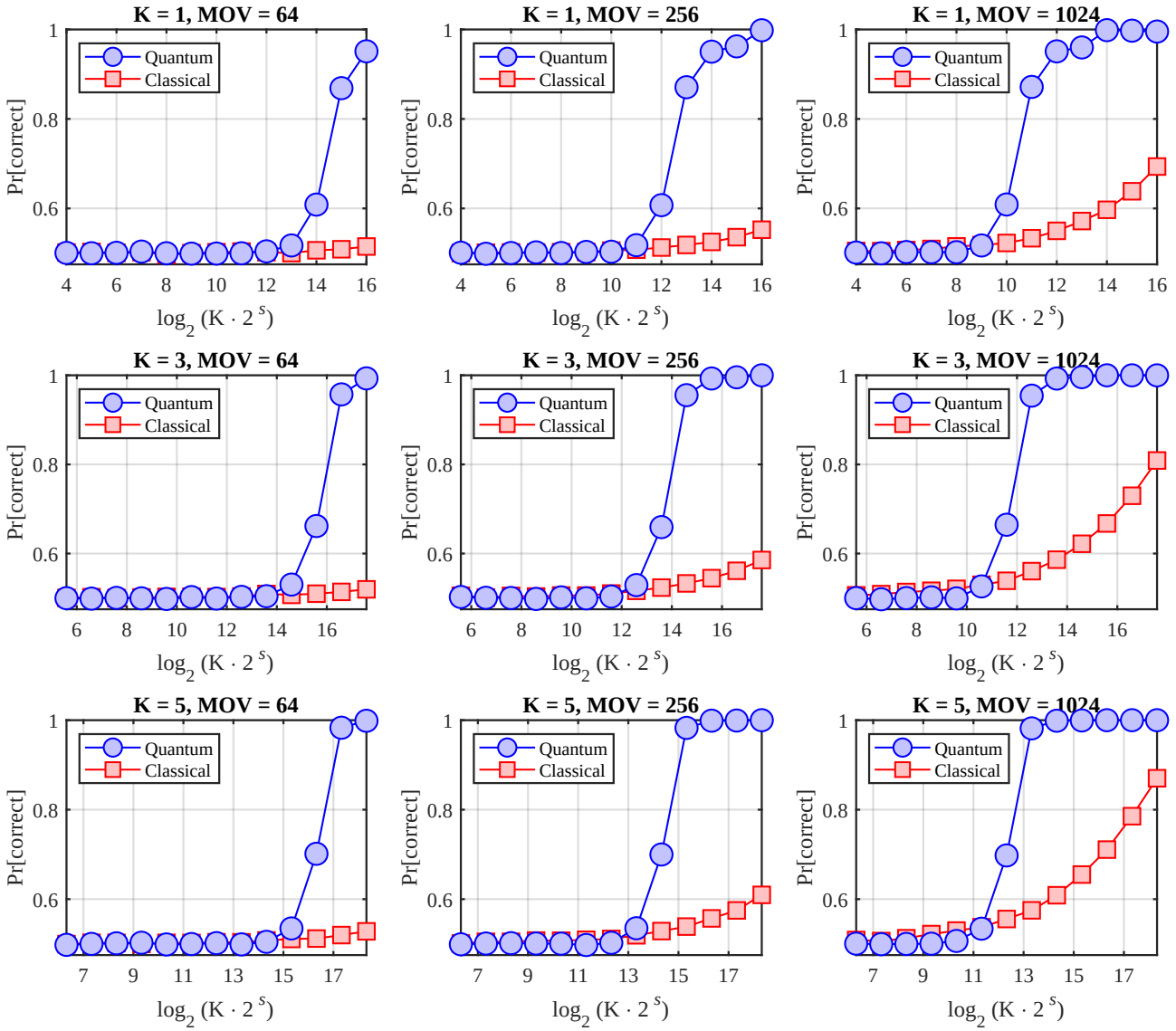


Figure 2: Compare quantum voting (blue circles) with classic voting (red squares) for plurality rule when $m = 2$. The horizontal axis can be seen as the logarithm of the algorithms' runtime.

of the algorithms' runtime in all figures. For all curves, we set $s = 4, \cdots, 16$ for the twelve points from left to right respectively. We set $K = 1, 3$, or 5 to avoid ties in the quantum algorithm. For plurality, we set MoV $= 64, 256$, or 1024, or equivalently, MoV $= n^{0.3}, n^{0.4}$ or $n^{0.5}$. For Borda, we set MoV $= 1024, 4096$, or 16384, or equivalently, MoV $= n^{0.5}, n^{0.6}$, or $n^{0.7}$. All experiments of this paper are implemented through MATLAB 2022b and run on a Windows 11 desktop with AMD Ryzen 9 5900X CPU and 32GB RAM.

Observations. The first observation is: the quantum voting algorithm has better accuracy than classical voting no matter which setting when the runtime is fixed. For example, MoV $= 1024$, $K = 1$, $m = 2$, and $s = 14$, the quantum voting algorithm outputs the correct winner almost for certain. However, the classical algorithm only has $\sim 60\%$ probability to output the correct winner. The second observation is: for MoV $= 1024$, $K = 1$, $m = 2$, to achieve $\sim 90\%$ for $\Pr[\text{correct}]$, the quantum algorithm requires 2^{10} runtime. In comparison, the classical algorithm with 2^{16} runtime can only achieve $\sim 70\%$ for $\Pr[\text{correct}]$. This matches our theoretical result: the proposed quantum voting algorithm is quadratically faster than any classical voting algorithm.

7 HEURISTICS TO FURTHER ACCELERATE QUANTUM VOTING.

In this section, we would like to provide further insights into quantum voting algorithms. Can quantum voting be further accelerated? We provide several possible heuristics.

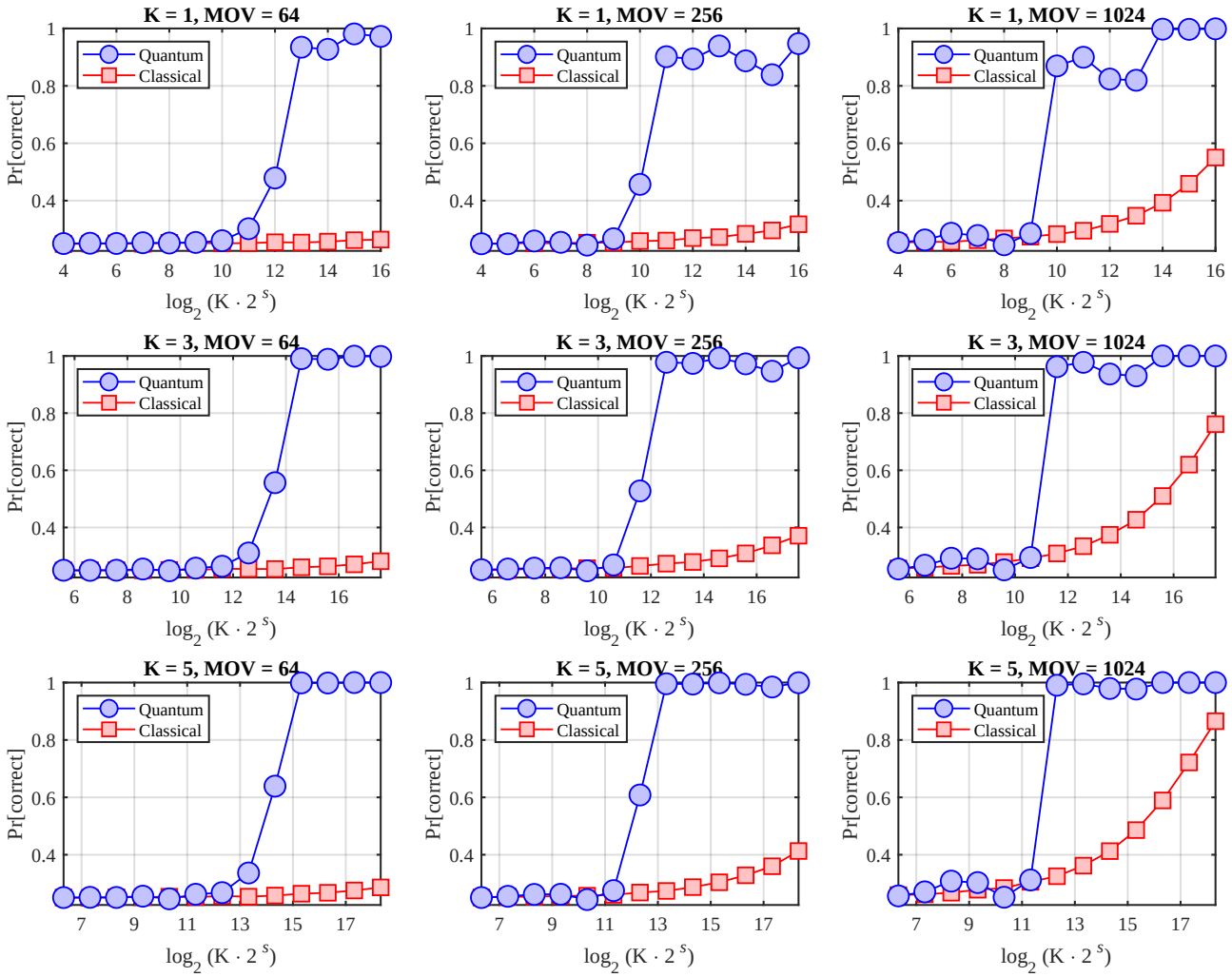


Figure 3: Compare quantum voting (blue circles) with classic voting (red squares) for plurality rule when $m = 4$. The horizontal axis can be seen as the logarithm of the algorithms' runtime.

Pre-sampling. One way to improve the average performance of the quantum 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 accuracy.

Sampling + Quantum. Another natural idea is to apply the quantum 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 accuracy, 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 accuracy.

8 CONCLUSIONS AND FUTURE WORKS

In conclusion, we took the first step in using quantum computation to accelerate voting. We found that a variety of common voting rules can be accelerated quadratically using quantum computation, including plurality, Borda, STV, Copeland, and so on. Our proposed quantum computation has the potential to 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 quantum voting algorithm by combining existing acceleration techniques of classical fast voting algorithms. It would also be interesting to extend the theoretical guarantee to a wider range of voting rules, such as generalized scoring rules Xia [2013], Liu et al. [2020], which contain most of the widely-used voting rules in real-world elections, and preference functions whose output is aggregated preference among all the candidates.

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. ISSN 0098-1354.

Akshay Ajagekar and Fengqi You. Quantum computing based hybrid deep learning for fault diagnosis in electrical power systems. *Applied Energy*, 303:117628, 2021. ISSN 0306-2619.

John Bartholdi, III, Craig Tovey, and Michael 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. *Electronic Proceedings in Theoretical Computer Science*, 297:50–67, jul 2019. doi: 10.4204/eptcs.297.4. URL https://doi.org/10.4204%2Feptcs.297.4.

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. *Phys. Rev. A*, 94:022308, Aug 2016.

André Berthiaume and Gilles Brassard. Oracle quantum computing. *Journal of modern optics*, 41(12):2521–2535, 1994.

Arnab Bhattacharyya and Palash Dey. Predicting winner and estimating margin of victory in elections using sampling. *Artificial Intelligence*, 296:103476, 2021. ISSN 0004-3702.

Felix Brandt, Vincent Conitzer, Ulle Endriss, Jerome Lang, and Ariel D. Procaccia, editors. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum counting. In Kim G. Larsen, Sven Skyum, and Glynn Winskel, editors, *Automata, Languages and Programming*, pages 820–831, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-68681-1.

Markus Brill. Interactive Democracy. In *Proceedings of AAMAS*, 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. ISSN 0020-0190.

Vincent Conitzer, Matthew Rognlie, and Lirong Xia. Preference functions that score rankings and maximum likelihood estimation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 109–115, Pasadena, CA, USA, 2009.

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

Bailey Flanigan, Paul Gölz, Anupam Gupta, and Ariel D Procaccia. Neutralizing self-selection bias in sampling for sortition. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6528–6539. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/48237d9f2dea8c74c2a72126cf63d933-Paper.pdf.

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. *Phys. Rev. A*, 78:052310, Nov 2008a. doi: 10.1103/PhysRevA.78.052310. URL https://link.aps.org/doi/10.1103/PhysRevA.78.052310.

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 AAMAS*, 2021.

Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917855.

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.

Elham Kashefi, Adrian Kent, Vlatko Vedral, and Konrad Banaszek. Comparison of quantum oracles. *Physical Review A*, 65(5):050304, 2002.

Alastair Kay. Tutorial on the quantikz package, 2018.

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(69–75), 2015.

Andrew Mao, Ariel D. Procaccia, and Yiling Chen. Better human computation through principled voting. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Bellevue, WA, USA, 2013.

John C. McCabe-Dansted, Geoffrey Pritchard, and Arkadii Slinko. Approximability of Dodgson's rule. *Social Choice and Welfare*, 31:311—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.

Wim Van Dam. Quantum oracle interrogation: Getting all information for almost half the price. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 362–367. IEEE, 1998.

Toby Walsh and Lirong Xia. Lot-based voting rules. In *Proceedings of the Eleventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 603–610, Valencia, Spain, 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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):2189–2196, 2019.

Lirong Xia. Generalized scoring rules: a framework that reconciles borda and condorcet. *ACM SIGecom Exchanges*, 12(1):42–48, 2013.

Lirong Xia and Weiqiang Zheng. Beyond the worst case: Semi-random complexity analysis of winner determination. In *Web and Internet Economics*, pages 330–347, Cham, 2022. Springer International Publishing.

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

A IMPLEMENTATION OF QUANTUM VOTING ALGORITHM.

In this section, we aim to introduce the implementation of quantum voting algorithm in a more technical perspective. We will first introduce the basics in quantum computing. Then we will specify the implementation of circuits of quantum counting in Algorithm 1, and why they accelerates the voting process.

A.1 QUANTUM BASICS.

Basic quantum computation. Quantum bit (or *qubit* in short) is the counterpart of classical *bit*, which takes a deterministic binary from $\{0, 1\}$. Qubit, on the other hand, is represented by a linear combination of $\{|0\rangle, |1\rangle\}$, which are counterparts to $\{0, 1\}$, respectively. That is, every qubit $|\psi\rangle$ is written as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are complex numbers and are usually called amplitudes. If we measure the qubit, there is $|\alpha|^2$ probability to get 0 and $|\beta|^2$ probability to get 1. Naturally, we always have $|\alpha|^2 + |\beta|^2 = 1$ because the probabilities should sum to 1. Qubits sometimes are written as vectors to simplify notations. Formally,

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \triangleq \alpha|0\rangle + \beta|1\rangle.$$

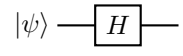
$t > 1$ qubits are presented as a 2^t-dimensional vector, where the j-th component of the vector (denoted as α_j) represents the amplitude of $|j_1 \cdots j_t\rangle$ (or $|j\rangle$), where $j_1 \cdots j_t$ is the binary representation of j. Similar to the 1-qubit case, the probability of observing $j_1, \cdots, j_t$ from those t qubit equals to $|\alpha_j|^2$.

A quantum operation (quantum gate) Q on t qubits is denoted by a $2^t \times 2^t$ unitary matrix, which means the matrix's inverse is its Hermitian conjugate. Applying a quantum operation Q on quantum state $|\psi\rangle$ is denoted by

$$Q|\psi\rangle \triangleq \boldsymbol{Q}_{(2^t\times 2^t)}\ \vec{\psi}_{(2^t)},$$

where the the quantum operator $\boldsymbol{Q}_{(2^t\times 2^t)}$ is a $2^t \times 2^t$ unitary matrix and the quantum state $\vec{\psi}_{(2^t)}$ is a 2^t dimensional column vector.

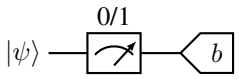
Quantum circuit of some useful quantum operators.[3] Quantum circuits run from the left-hand side to the right-hand side. For example, the following circuit means applying Hadamard gate H on a quantum state $|\psi\rangle$.



where $\boldsymbol{H} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$.

[3]All quantum circuits of this paper are drawn using the Quantikz package [Kay, 2018] for LaTeX.

The quantum circuit notion

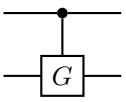


denotes measuring quantum state $|\psi\rangle$ with $0/1$ base (b denotes the result of measurement). Naturally, the complexity of quantum measurement and Hadamard gate are both $\Theta(1)$.

Quantum oracle [Berthiaume and Brassard, 1994, Van Dam, 1998, Kashefi et al., 2002] is a widely-used operator to encode binary functions or binary information. Given t qubits and a binary function $f : \{0, \cdots, 2^t - 1\} \mapsto \{0, 1\}$, quantum oracle (based on function $f(\cdot)$) applies a phase shift of $-1 = e^{\pi i}$ if $f(x) = 1$ and does nothing otherwise. We can query oracle many times and regard the number of queries as the cost [Grover, 1996]. Formally,

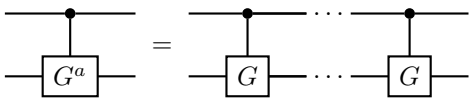
$$\begin{cases} O_f|x\rangle = |x\rangle & \text{if } f(x) = 1\\ O_f|x\rangle = -|x\rangle & \text{otherwise}\end{cases}.$$

Suppose we have a quantum gate G on t qubits. The following operation is called *controlled-G*.



$$= \begin{bmatrix}\boldsymbol{I}_{(2^t\times 2^t)} & \boldsymbol{0}_{(2^t\times 2^t)}\\ \boldsymbol{0}_{(2^t\times 2^t)} & \boldsymbol{G}_{(2^t\times 2^t)}\end{bmatrix},$$

where $\boldsymbol{I}$ denotes the identity matrix, and $\boldsymbol{0}$ denotes the zeros matrix. To simplify notations, we also write



(repeat a times).

A.2 IMPLEMENTATION OF QUANTUM COUNTING CIRCUIT.

Figure 4 shows the quantum counting circuit, which is a combination of Grover search algorithm [Grover, 1996] and quantum reverse Fourier transformation (the $QFT^\dagger$ operator) Followings we focus on introducing Grover algorithm and why it accelerates the computation.

Grover operator. Grover algorithm is an efficient search algorithm. Given a binary function $f : \{0, 1, \cdots, 2^t - 1\} \to \{0, 1\}$, Grover algorithms returns an x with $f(x) = 1$ with high probability. The Grover operation in Algorithm 1 is constructed by the quantum circuit in Figure 5, where $t = \lceil \log n \rceil$ denotes the minimum number of quantum bits to encode n. The quantum operator QPS is called quantum phase shifting, which provides a phase shift of -1 on every state except $|0\rangle$. Mathematically,

$$|0\rangle \xrightarrow{QPS} |0\rangle \qquad \text{and}$$
$$|x\rangle \xrightarrow{QPS} -|x\rangle \ \text{ for any } x \in 1, \cdots, 2^t - 1.$$

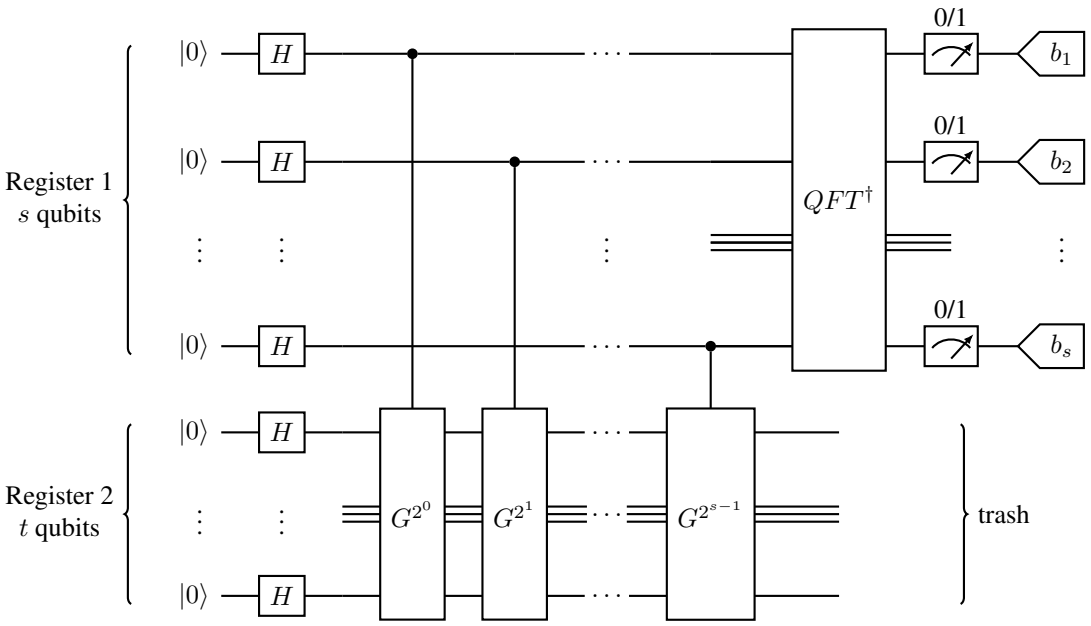


Figure 4: The circuit for quantum counting algorithm.

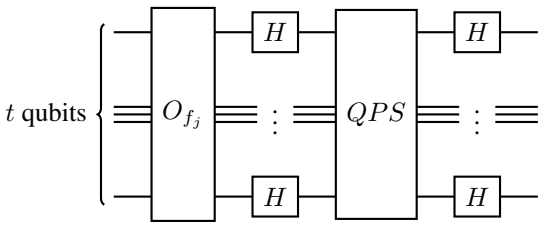


Figure 5: The circuit for Grover operator.

Here, $|x\rangle$ represents the x-th base state of the t qubits. The high-level idea of Grover operator's functionality is shown in Figure 6, where $|\psi\rangle$ is the input of Grover operators in quantum counting, and $\{|\alpha\rangle, |\beta\rangle\}$ is a pair of orthogonal bases. The formal definition of $|\psi\rangle$, $|\alpha\rangle$, and $|\beta\rangle$ can be found in Appendix A.3. Under the $|\alpha\rangle$ $|\beta\rangle$ base, the quantum oracle O_{f_j} reflects $|\psi\rangle$ over $|\alpha\rangle$, while the rest parts of G reflects $O_{f_j}|\psi\rangle$ over $|\psi\rangle$. The angle between the output state $G|\psi\rangle$ and initial state $|\psi\rangle$

$$\theta = 2\arcsin\left(\sqrt{\mathbf{hist}_j \cdot 2^{-t}}\right),$$

which includes the information about $\mathbf{hist}_j$. Since function $\arcsin(\sqrt{x})$ grows quadratically faster than linear functions when x is small, we expect that an estimation about $\arcsin(\sqrt{x})$ could be quadratically more accurate than directly estimate x.

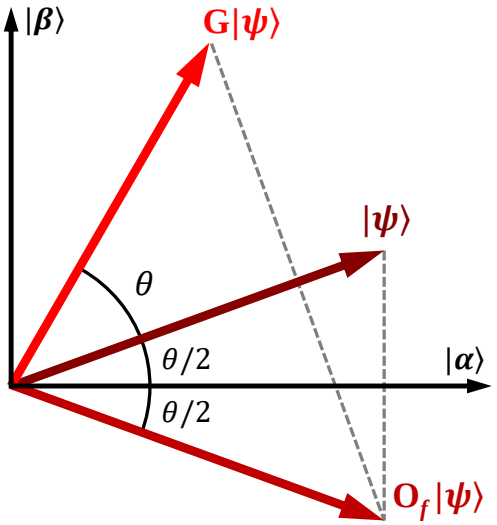


Figure 6: An illustration of Grover operator's functionality (Figure 6.3 in Nielsen and Chuang [2010]).

A.3 FUNCTIONALITY FOR GROVER ALGORITHM

According to (6.4) in Nielsen and Chuang [2010], Hadamard gate changes t qubits of $|0\rangle$ to an equal superposition state (equal probability of observing any outcome under quantum measurements).

$$|\psi\rangle = \frac{1}{2^{t/2}} \cdot \sum_{x=0}^{2^t-1} |x\rangle.$$

Letting $f : \{0, \cdots, 2^t - 1\} \mapsto \{0, 1\}$ be the binary function to construct the quantum oracle, and $\mathring{n}_1$ be the number of x such that $f(x) = 1$. The orthogonal bases $|\alpha\rangle$ and $|\beta\rangle$ are

defined as,

$$|\alpha\rangle \triangleq \frac{1}{\sqrt{2^t - \mathring{n}_1}} \cdot \sum_{x:f(x)=0} |x\rangle \qquad \text{and}$$
$$|\beta\rangle \triangleq \frac{1}{\sqrt{\mathring{n}_1}} \cdot \sum_{x:f(x)=1} |x\rangle.$$

Under the $|\alpha\rangle\ |\beta\rangle$ base, the equal superposition state

$$|\psi\rangle = \sqrt{\frac{2^t - \mathring{n}_1}{2^t}}\ |\alpha\rangle + \sqrt{\frac{\mathring{n}_1}{2^t}}\ |\beta\rangle.$$

Since

$$\theta = 2\arcsin\left(\sqrt{\mathring{n}_1 \cdot 2^{-t}}\right),$$

we have

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)\ |\alpha\rangle + \sin\left(\frac{\theta}{2}\right)\ |\beta\rangle,$$
$$O_f|\psi\rangle = \cos\left(\frac{\theta}{2}\right)\ |\alpha\rangle + \sin\left(-\frac{\theta}{2}\right)\ |\beta\rangle, \text{ and}$$
$$G|\psi\rangle = \cos\left(\frac{3\theta}{2}\right)\ |\alpha\rangle + \sin\left(\frac{3\theta}{2}\right)\ |\beta\rangle.$$

B ADDITIONAL EXPERIMENTAL RESULTS

Figure 7 and Figure 8 plot the comparison between quantum voting and classical voting for $m = 2$ and $m = 4$ respectively when the voting rule is Borda. Similar behavior as the plurality can be observed for Borda. We note that the probability of the quantum algorithm outputting the correct winner does not necessarily increase with the runtime. This is because a finer quantization process may decrease the probability of getting the best estimation.

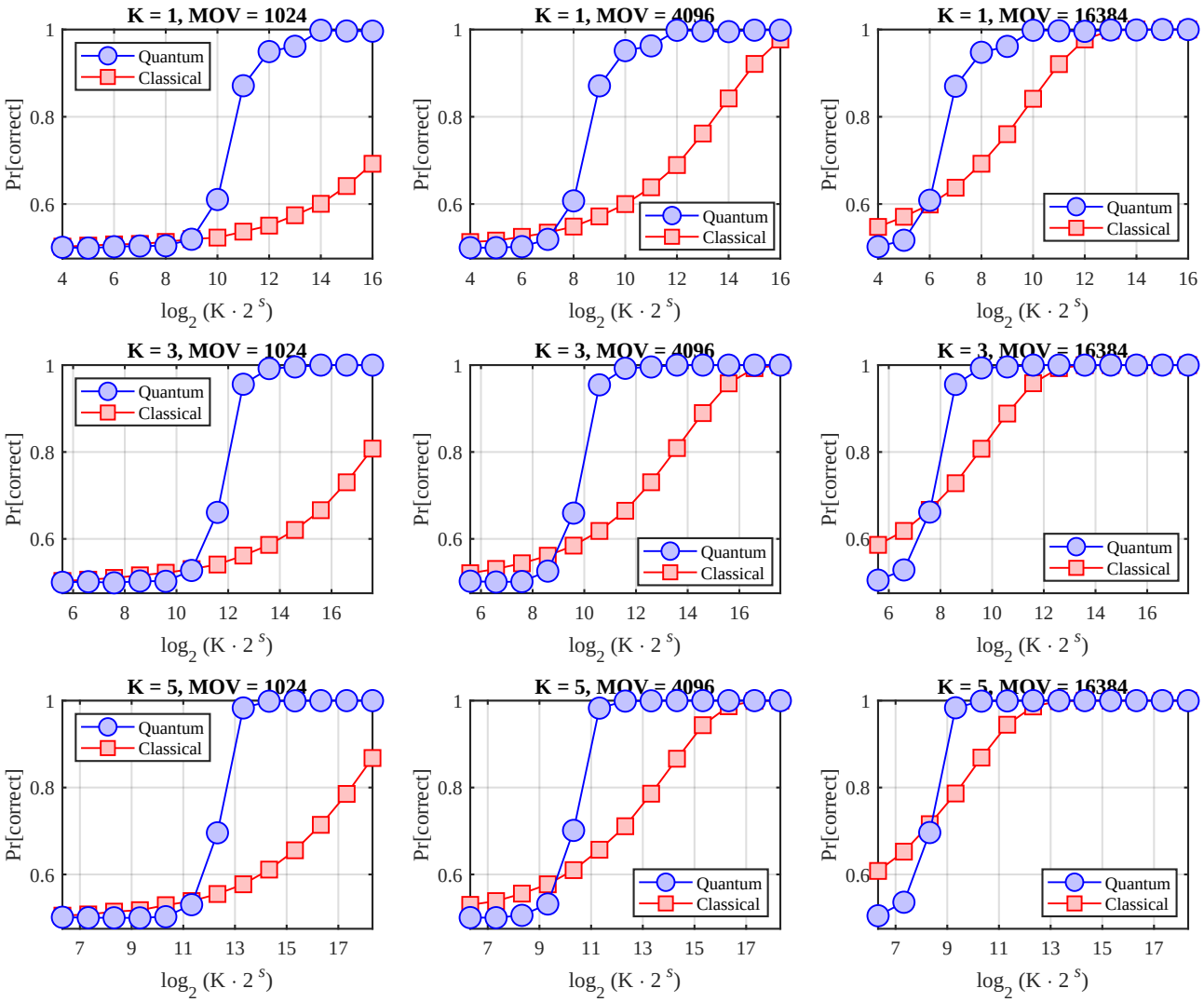


Figure 7: Compare quantum voting (blue circles) with classic voting (red squares) for Borda rule when $m = 2$. The horizontal axis can be seen as the logarithm of the algorithms' runtime.

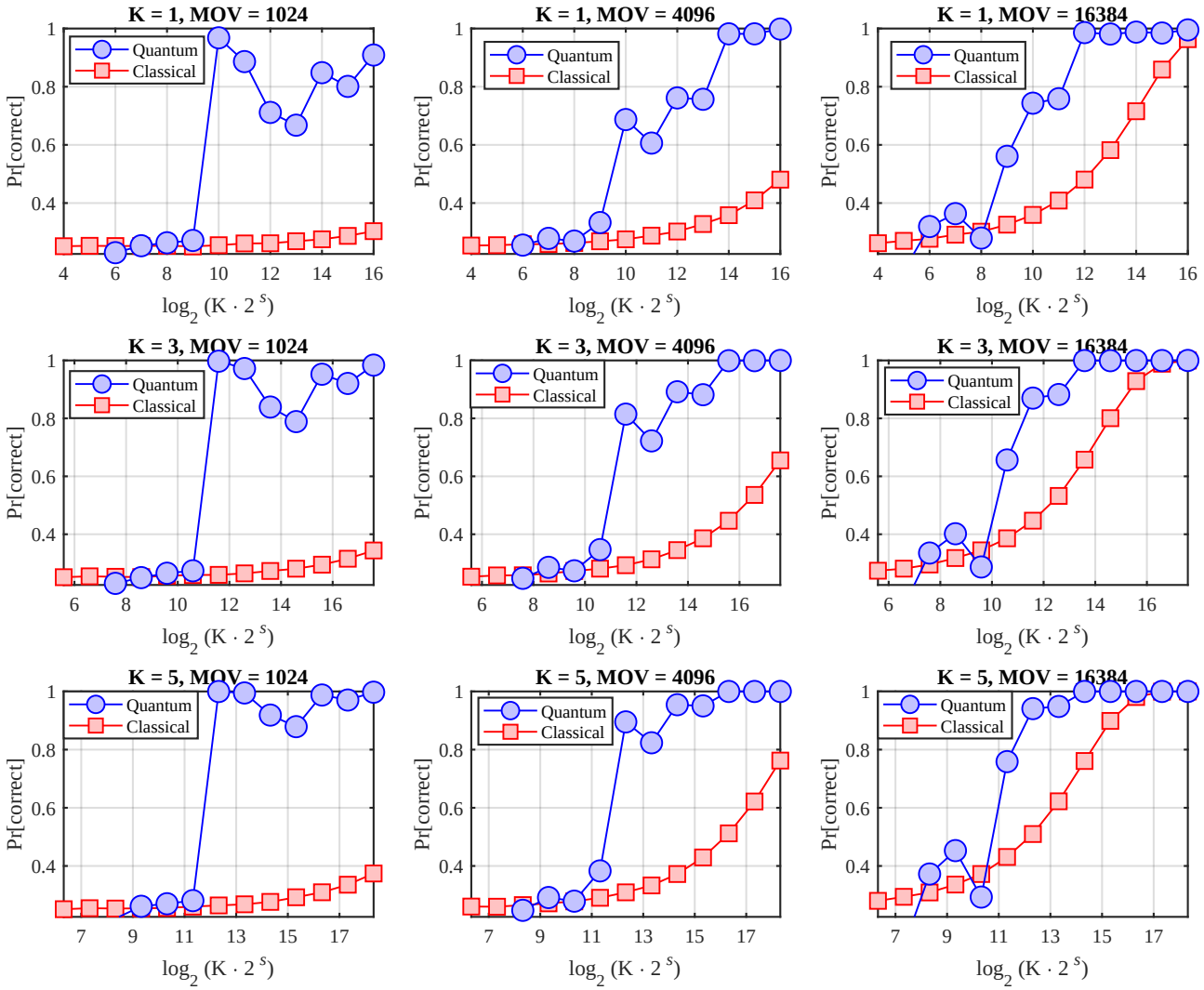


Figure 8: Compare quantum voting (blue circles) with classic voting (red squares) for Borda rule when $m = 4$. The horizontal axis can be seen as the logarithm of the algorithms' runtime.