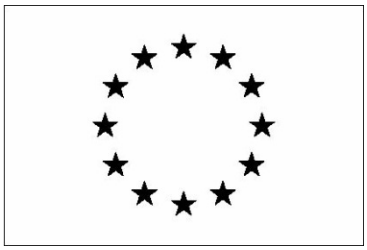

Grant Agreement number: **248095**

Project acronym: **Q-ESSENCE**

Project title: **Quantum Interfaces, Sensors, and Communication based on Entanglement**

Funding Scheme: **Collaborative Project (Large-Scale Integrating Project)**

DELIVERABLE REPORT

Deliverable no.:	**D3.5.2**
Deliverable name:	**Report on quantum computations being classically simulable and hybrid quantum-classical communications**
Workpackage no.	**WP3.5**
Lead beneficiary	**UCAM**
Nature	**R = Report**
Dissemination level	**PU = Public**
Delivery date from Annex I (proj month)	**30/04/2013**
Actual delivery date	**30/04/2013**

The report on classically (non-)simulable quantum computations can be found in three works:

[RH1-11] A generalization of the standard oracle model has been studied, in which the oracle acts on the target with a permutation which is selected according to internal random coins. It has been shown new exponential quantum speedups turn up, in comparison to classical algorithms in this oracle model. Even stronger results are obtained: several problems which are impossible to solve classically but can be solved by a quantum algorithm using a single query. The other basic result is that such infinity-vs.-one separations between classical and quantum query complexities can be constructed from any separation between classical and quantum query complexities (in the unbounded-error regime). Conditions are given which determine when oracle problems, either in the standard model, or in any of the considered generalizations, cannot be solved with success probability better than random guessing would achieve. In the oracle model with internal randomness, for which the goal is to gain any non-zero advantage over guessing, is shown that (roughly speaking) that (k) quantum queries are equivalent in power to (2k) classical queries. Thus results of Meyer and Pommersheim, and Montanaro, Nishimura and Raymond have been extended.

[MJM1-11] Several families of total Boolean functions are presented, which have exact quantum query complexity, which is a constant multiple (between 1/2 and 2/3) of their classical query complexity, and it is shown that optimal quantum algorithms for these functions cannot be obtained by simply computing parities of pairs of bits. A characterization is given for the model of non-adaptive exact quantum query complexity in terms of coding theory. A complete characterization of the query complexity of symmetric Boolean functions in this context is given.

[ME3-12] This third work complements the previous two, in that a class of circuits is presented that can be efficiently simulated. Indeed, it turns out that quantum circuits where the initial state and all the following quantum operations can be represented by positive Wigner functions can be classically efficiently simulated. This is true both for continuous-variable as well as discrete variable systems in odd prime dimensions, two cases which will be treated on entirely the same footing. Noting the fact that Clifford and Gaussian operations preserve the positivity of the Wigner function, our result generalizes the Gottesman-Knill theorem. The derived algorithm provides a way of sampling from the output distribution of a computation or a simulation, including the efficient sampling from an approximate output distribution in case of sampling imperfections for initial states, gates, or measurements. In this sense, this work highlights the role of the positive Wigner function as separating classically efficiently simulable systems from those that are potentially universal for quantum computing and simulation, and it emphasizes the role of negativity of the Wigner function as a computational resource.

The second part of the deliverable has been achieved by the end of the project (models of hybrid quantum-classical computations).

[JN1-13] Clifford gates are a winsome class of quantum operations combining mathematical elegance with physical significance. The Gottesman-Knill theorem asserts that Clifford computations can be classically efficiently simulated but this is true only in a suitably restricted setting. In the work Clifford computations were considered with a variety of additional ingredients: (a) strong vs. weak simulation, (b) inputs being computational basis states vs. general product states, (c) adaptive vs. non-adaptive choices of gates for circuits

involving intermediate measurements, (d) single line outputs vs. multi-line outputs. The classical simulation complexity of all combinations of these ingredients were considered and shown that many are not classically efficiently simulable (subject to common complexity assumptions such as P not equal to NP). The results reveal a surprising proximity of classical to quantum computing power viz. a class of classically simulable quantum circuits which yields universal quantum computation if extended by a purely classical additional ingredient that does not extend the class of quantum processes occurring. An expository discussion and proof of the relationship between Clifford computations and the classical complexity class parity-L originally obtained by Aaronson and Gottesman has also been provided.

[RH1-11] D. Rosenbaum and A. Harrow, Uselessness for an Oracle Model with Internal Randomness arXiv:1111.1462.
[MJM1-11] A. Montanaro, R. Jozsa and G. Mitchison, On exact quantum query complexity, arXiv:1111.0475.
[JN1-13] Richard Jozsa and Maarten Van den Nest, Classical simulation complexity of extended Clifford circuits (arXiv:1305.6190).

Uselessness for an Oracle Model with Internal Randomness

David Rosenbaum and Aram Harrow
University of Washington,
Department of Computer Science and Engineering
Email: djr@cs.washington.edu, aram@cs.washington.edu

November 8, 2011

Abstract

We consider a generalization of the standard oracle model in which the oracle acts on the target with a permutation which is selected according to internal random coins. We show new exponential quantum speedups which may be obtained over classical algorithms in this oracle model. Even stronger, we describe several problems which are impossible to solve classically but can be solved by a quantum algorithm using a single query; we show that such infinity-vs-one separations between classical and quantum query complexities can be constructed from any separation between classical and quantum query complexities (in the unbounded-error regime).

We also give conditions to determine when oracle problems—either in the standard model, or in any of the generalizations we consider—cannot be solved with success probability better than random guessing would achieve. In the oracle model with internal randomness where the goal is to gain any nonzero advantage over guessing, we prove (roughly speaking) that k quantum queries are equivalent in power to $2k$ classical queries, thus extending results of Meyer and Pommersheim, and Montanaro, Nishimura and Raymond.

1 Introduction

Oracles are an important conceptual framework for understanding quantum speedups. They may represent subroutines whose code we cannot usefully examine, or an unknown physical system whose properties we would like to estimate. When used by a quantum computer, the most general form of an oracle is a possibly noisy quantum operation that can be applied to an n-qubit input. However, oracles this general have no obvious classical analogue, which makes it difficult to compare the ability of classical and quantum computers to efficiently interrogate oracles. This was the original motivation of the *standard oracle model*, in which f is a function from $[N] = \{1, \ldots, N\}$ to $\{0, 1\}$, and the oracle $\mathcal{O}_f$ acts for a classical computer by mapping x, y to $x, y \oplus f(x)$, and for a quantum computer as a unitary that maps $|x, y\rangle$ to $|x, y \oplus f(x)\rangle$. One way to justify the standard oracle model is that if there is a (not necessarily reversible) classical circuit computing f, then $\mathcal{O}_f$ can be simulated by computing f, XORing the answer onto the target, and uncomputing f. This model is depicted in figure 1.

In this paper, we consider other forms of oracles that are more general than the standard oracle model, but nevertheless permit comparison between classical and quantum query complexities. Meyer and Pommersheim [1] generalized the standard model by letting A be a deterministic classical algorithm which takes the control x of the oracle and computes a value $A(x)$. The oracle then acts by applying a permutation $\pi_{A(x)}$ to

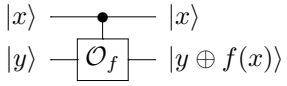

Figure 1: An oracle in the standard model

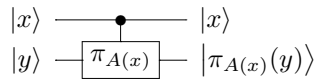

Figure 2: An oracle in the internal randomness model

the target. We will further generalize the model by replacing A with a randomized classical algorithm. The random coins used by A are internal to the oracle and cannot be accessed externally. We call this concept an *oracle with internal randomness*. A general oracle with internal randomness is shown in figure 2. Note that even if there are no controls as in figure 3, such an oracle can still be interesting since it may apply different permutations depending on its internal coin flips.

Oracles with internal randomness correspond naturally to the situation in which a (quantum or classical) computer seeks to determine properties of a device which acts in a noisy or otherwise non-deterministic manner. One simple example is an oracle that "misfires", i.e. when queried, the oracle does nothing with probability p and responds according to the standard oracle model with probability $1 - p$. This model was considered in [2], which found, somewhat surprisingly, that the square-root advantage of Grover search disappears (i.e. there is an $\Omega(N)$ quantum query lower bound for computing the OR function) for any constant $p > 0$.

The rest of our paper is divided into two parts. First, we explore various examples of oracles with internal randomness that demonstrate the power of the model. We will see that in some cases, this can even result in problems solvable with one quantum query that are completely unsolvable using classical queries. The basic idea behind these oracles is first demonstrated in Section 2, where we contrast the consequences of the various oracle models by considering the problem of determining whether two sets are equal or disjoint. This problem is further generalized in Section 4 to several group-theoretic problems: determining membership in the same coset, equivalence of orbit cosets, and distinguishing permutations with certain cycle structures.

In the second part, we consider the question of when oracle problems can be solved with any nontrivial advantage; i.e. a probability of success better than could be obtained by simply guessing the answer according to the prior distribution. For an example of when such advantage is *not* possible, consider the parity function on N bits. If these bits are drawn from the uniform distribution, then any classical algorithm making $\leq N-1$ queries will not be able to guess the parity with any nontrivial advantage. In Section 5, we consider the problem of when some number of queries are *useless* for solving an oracle problem. Informally, our main result is roughly that that k quantum queries are useless (in a way we formalize later) if and only if $2k$ classical queries are useless. However, a subtlety arises in our theorem when oracles have internal randomness, in that the $2k$ classical queries need to be considered as k pairs, each of which uses a separate sample from the internal randomness of the oracle.

Special cases of our result had been previously obtained by Farhi, Goldstone, Gutman and Sipser [3], who proved the result for the case of the parity function; by Meyer and Pommersheim [4], who proved that if $2k$ classical queries are useless for the permutative oracle model then so are k quantum queries, and by Montanaro, Nishimura and Raymond [5], who proved the equivalence in the standard oracle model when f has binary output, using techniques that do not readily generalize to non-binary f. Our proof is arguably simpler and more operational. We introduce an analogue of gate teleportation [6] for oracles by showing that oracles can be (a) encoded into states analogous to Choi-Jamiołkowski states, and (b) retrieved from those states with an exponentially small, but heralded, success probability. We expect that this characterization will be useful for future study of query complexity in the regime where any nonzero advantage is sought. We conclude by leveraging our encoding to obtain a generalized method of constructing of problems with infinity-vs-one query complexity separations from any problem which has a separation between the classical and quantum query complexities and discuss how this can be extended to the bounded-error model (though this comes at the price of increasing the number of quantum queries).

2 The loops problem

Let π_1 and π_2 be permutations on $[N]$ which are given as black boxes. We think of each $\pi_i[N/2] = L_i$ as the exponentially large loop which consists of the edges $\pi_i(k) \leftrightarrow \pi_i(k+1)$ for each $k \in [N/2-1]$ and $\pi_i(N) \leftrightarrow \pi_i(1)$. We are promised that either

- L_1 and L_2 are disjoint or
- L_1 and L_2 contain the same nodes

The loops problem is to decide which of these is the case. This is a variant of the collision problem[7, 8]. The key difference is our requirement that π_1, π_2 (thought of as maps from $[N/2]$ to $[N]$) be injective, which will later make it possible to encode them in non-standard oracles.

In the rest of this section, we explore encodings of the loops problem using different oracle models, with an eye towards establishing stronger quantum-classical query complexity separations than can be achieved with the standard model. We will see that several models permit fast quantum solutions, but not all of them admit fair comparisons with classical algorithms.

2.1 Formulation using unitary oracles

The most natural way to formulate this problem precisely is to provide each π_i as a black box unitary oracle U_i. Unlike oracles in the standard model which take a control x and compute $f(x)$ which is then XORed into the target as shown in figure 1, unitary oracles have no controls but may perform an arbitrary unitary operation on their input. The classical analogue is a black box that applies a permutation to a target register.Although this problem is hard classically in this formulation, there is a simple quantum algorithm (see Algorithm 1) which makes this model uninteresting. The probability that this algorithm returns the correct answer can be made arbitrarily close to 1 by increasing the accuracy of the phase estimation used to perform the inversion.

Algorithm 1 A quantum algorithm for the unitary black box formulation

1: Compute $U_1 |1\rangle = |\pi_1(1)\rangle$
2: Using phase estimation [9], construct $\hat{U}_2^\dagger$ which is a good approximation of $U_2^\dagger$
3: Measure $\hat{U}_2^\dagger |\pi_1(1)\rangle$ in the computational basis
4: **if** the outcome k is in $[N/2]$ **then**
5: Return "$L_1 = L_2$"
6: **else**
7: Return "$L_1 \neq L_2$"
8: **end if**

2.2 Formulation using randomized and quantum oracles

Another approach is to encode π_1 and π_2 into probability distributions. In this formulation, we make a distinction between the classical and quantum versions of the oracles. The classical oracle P_i for π_i outputs a value from L_i which is chosen uniformly at random; the quantum version of the oracle performs the analogous operation of preparing the state $|L_i\rangle$.

To justify this formulation, we think of oracles as being generated by circuits whose internal structure is in principle accessible, but in practice we do not know any way to access this structure besides evaluating it. If such a circuit has random elements, then in some cases, we can modify it to produce a coherent superposition of outputs instead of a random mixture. The paper [10] calls this task *q-sampling* and describes several general scenarios in which this is possible. Another general model is when a probability distribution is specified by a data structure such as a Bayesian network. If this data structure is known, then it is again possible to construct a quantum circuit that q-samples from the appropriate distribution.

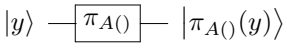

Figure 3: An oracle in the internal randomness model with no controls

However, in general, q-sampling requires substantial additional assumptions about how the oracle is constructed. Thus either the classical algorithms can *also* exploit this additional structure, or the quantum algorithm has an unfair advantage.

2.3 Formulation using indistinguishable oracles

Consider again the formulation of the loops problem in Section 2.1. The main feature of the formulation which allows a quantum algorithm to succeed using phase estimation to invert one of the unitary oracles is that the labels 1 and 2 of π_1 and π_2 are *known* to the algorithm designer. To see this, we can think of the algorithm designer as having a box filled with different devices which perform unitary operations and measurements; he is also given two black box devices which implement π_1 and π_2 and have labels marking them as "π_1" and "π_2". To solve the problem, the algorithm designer prepares a quantum state and then takes a device out of the box and applies it to the state; after using it once, the device is placed back inside the box. By repeating this process with different devices, the algorithm designer can invert one of the unitary oracles and solve the problem. Suppose that we consider the same problem except that the labels π_1 and π_2 on the devices have been erased so that the two devices now look *exactly* the same. In this case, the algorithm designer can no longer perform phase estimation since after he uses one of the unitary oracles once and puts it back into the box, he will not be able to find the same oracle again without having a 50% chance of using the wrong oracle by mistake. We now focus on finding an oracle model in which we can implement this scenario. To do this, it is necessary to use an oracle with internal randomness.

2.4 Formulation of the loops problem using oracles with internal randomness

The idea is to "erase" the labels of π_1 and π_2 by encoding both of them in a single oracle which acts as shown in Algorithm 2. Note that this oracle has no controls and is of the form shown in figure 3.

Algorithm 2 The oracle for the loops problem

1: Flip an internal fair coin
2: **if** the outcome is heads **then**
3: Apply π_1 to the target
4: **else**
5: Apply π_2 to the target
6: **end if**

Using the fair coin to select which permutation to apply is analogous to erasing the labels on the devices for π_1 and π_2 and putting each of them back into the box after each use in the scenario discussed above. This problem is hard in a classical setting as we will show. Our quantum algorithm for the loops problem is based on the swap test [11] which we now introduce.

Let $|\psi\rangle$ and $|\phi\rangle$ be two states which we wish to compare. The swap test is performed by preparing a qubit $|c\rangle$ in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and applying a swap controlled by $|c\rangle$ to the states $|\psi\rangle$ and $|\phi\rangle$. A Hadamard is then applied to the control qubit and a measurement is preformed in the computational basis. A simple calculation shows that the probability of measuring 0 (symmetric) is

$$\Pr(0) = \frac{1 + |\langle\psi|\phi\rangle|^2}{2} \tag{1}$$

If $|\psi\rangle = |\phi\rangle$, then the swap test will always return 0. However, if $\langle\psi|\phi\rangle = 0$ then 0 will be observed with probability 1/2. Thus, the swap test allows these two cases to be distinguished with one-sided error. In the sequel we refer to outcome 0 as the "symmetric" outcome and 1 as the "antisymmetric" outcome.

We now show how this problem may be solved by a quantum algorithm (see Algorithm 3) based on the swap test [11].

Algorithm 3 The quantum algorithm for the loops problem

1: Prepare the pair of states $|\alpha\rangle$ and $|\beta\rangle$ by applying the oracle $\mathcal{O}$ to $\sqrt{\frac{2}{N}}\sum_{x=1}^{N/2}|x\rangle$
2: Apply the swap test to $|\alpha\rangle$ and $|\beta\rangle$
3: **if** the swap test returns "symmetric" **then**
4: Return "$L_1 = L_2$"
5: **else**
6: Return "$L_1 \neq L_2$"
7: **end if**

Theorem 1. *There is a quantum algorithm which solves the loops problem using $O(1)$ queries and $O(\log N$ elementary operations with success probability at least $2/3$.*

The proof of the next theorem is straightforward but long so we defer it to Appendix B.

Theorem 2. *Randomized algorithms require $\Omega(\sqrt{N})$ queries to solve the loops problem with success probability $\geq 2/3$.*

3 Infinity-vs-one separations

In this section, we discuss problems which can be solved using a single quantum query but cannot be solved classically even with an unlimited number of queries. To achieve such infinity-vs-one separations it is necessary (but not sufficient) for the oracle to have internal randomness since otherwise one could simulate the quantum algorithm classically with exponential overhead. The key point is that internal randomness effectively causes a different oracle to be used for each query so such a simulation is not possible in this case.

3.1 Distinguishing involutions with no fixed points from cycles

We describe the problem of distinguishing involutions from cycles. Compared to the oracles discussed previously, this problem has several distinct properties:

- Classically, the problem *cannot be solved* even with an unlimited number of queries to the oracle.
- The problem can be solved by a quantum algorithm using only one query.
- In other words, the problem admits an infinity-vs-one separation between the classical and quantum query complexities.

Define $INV = \{\pi \in S_N | \, \pi^2 = \iota$ and $\pi x \neq x$ for all $x \in [N]\}$; this is the set of involutions in S_N with no fixed points. Let $CYC = \{\pi \in S_N | \, \pi$ is a cycle of length $N\}$. For any nonempty subset S of S_N, define $\mathcal{O}_S$ to be the oracle with a control $x \in [N]$ and a target $y \in [N]$ which acts according to Algorithm 4.

Theorem 3. *Classical algorithms cannot solve the problem of distinguishing cycles from involutions with no fixed points with unbounded error using any number of queries.*

Proof. In this problem, an oracle $\mathcal{O}_S$ is given which is either $\mathcal{O}_{INV}$ or $\mathcal{O}_{CYC}$; the problem is to determine which of these is the case. Consider querying the oracle when the control is x. Then $\pi(x)$ is a uniformly random value in $[N] \setminus \{x\}$ for both cases so this problem cannot be solved by a classical algorithm. □

Algorithm 4 The oracle for the problem of distinguishing involutions with no fixed points from cycles

1: Select $\pi \in S$ uniformly at random
2: Compute $\pi(x)$ where x is the value of the control
3: Add $\pi(x)$ to the target y modulo N

However, the problem can be solved by a quantum algorithm using a single query to the oracle as shown in Algorithm 5.

Algorithm 5 The quantum algorithm for distinguishing involutions with no fixed points from cycles

1: Prepare the state $\frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x\rangle$
2: Apply $\mathcal{O}_S$ to obtain the state $\frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x, \pi(x)\rangle$
3: Perform the measurement $\{M_{xy}\}_{1 \leq x < y \leq N}$ where $M_{xy} = |xy\rangle \langle xy| + |yx\rangle \langle yx|$
4: Let M_{xy} be the measurement outcome and let $|\psi\rangle$ be the residual state.
5: Measure $|\psi\rangle$ in the $\frac{|xy\rangle \pm |yx\rangle}{\sqrt{2}}$ basis
6: **if** the outcome is $\frac{|xy\rangle + |yx\rangle}{\sqrt{2}}$ **then**
7: Return "INV"
8: **else**
9: Return "CYC"
10: **end if**

The measurement $\{M_{xy}\}$ has measurement outcomes that range over all $x < y$. We omit the subspace spanned by the states $|x, x\rangle$, since we are promised that π has no fixed points, and so our states will have no overlap with this subspace. To implement the measurement, we use Algorithm 6.

Algorithm 6 Implementing the measurement $\{M_{xy}\}$

1: Let the input be $|w, z\rangle$
2: Compute $|w, z\rangle \otimes |\min\{w, z\}, \max\{w, z\}\rangle$
3: Measure the last two registers; the measurement outcome is $\{x = \min\{w, z\}, y = \max\{w, z\}\}$

Algorithm 5 may be simplified by applying the swap directly to the state $\frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x, \pi(x)\rangle$ as shown in Algorithm 7.

We now show that the above algorithm effectively counts the number of transpositions in an arbitrary permutation which is sufficient to distinguish involutions from cycles.

Theorem 4. *Quantum algorithms can solve the problem of distinguishing cycles from involutions with no fixed points using a single query with one-sided error* $1/2$.

Proof. Consider a general state ρ^{AB} on two identical systems A and B. Then applying the swap test to this system (where the swap exchanges A and B) outputs 0 (symmetric) with probability

$$\Pr(0) = \frac{1 + \operatorname{tr} \rho^{AB} F}{2} \tag{2}$$

where F is a swap operation. Applying this formula to the state $\frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x, \pi(x)\rangle$, the probability of observing 0 is

Algorithm 7 The simplified quantum algorithm for distinguishing involutions with no fixed points from cycles

1: Prepare the state $\frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x\rangle$
2: Apply $\mathcal{O}_S$ to obtain the state $\frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x, \pi(x)\rangle$
3: Apply the swap test to $\frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x, \pi(x)\rangle$
4: **if** the swap test outputs "symmetric" **then**
5: Return "INV"
6: **else**
7: Return "CYC"
8: **end if**

$$\Pr(0) = \frac{1 + (1/N) \sum_{xy} \langle \pi(y)|x\rangle \langle y|\pi(x)\rangle}{2} \tag{3}$$

$$= \frac{1 + (1/N) \left|\{(x,y)|\, \pi(x) = y \text{ and } \pi(y) = x\}\right|}{2} \tag{4}$$

This probability is 1/2 if $\pi \in CYC$ and is 1 if $\pi \in INV$. □

Hence, there is an infinity-vs-one separation in the unbounded-error classical and quantum query complexities for this problem.

This analysis also shows that if the oracle $\mathcal{O}_S$ in Algorithm 7 is replaced by an oracle which always applies the same permutation π (which is now an arbitrary element of S_N), then an estimate may be obtained for $\Pr(0)$ by repeatedly applying the swap test to the state $\frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x, \pi(x)\rangle$. This in turn will allow an estimate to be obtained for number of pairs (x, y) such that $\pi(x) = y$ and $\pi(y) = x$. Note that each fixed point of π is counted once in this value while each transposition $(x\ y)$ is counted once for the pair (x, y) and once for the pair (y, x). To obtain an estimate for the number of transpositions, the number of fixed points of π can be estimated by repeatedly performing the measurement $\{|x, x\rangle \langle x, x|\}$ on the state $\frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x, \pi(x)\rangle$. This may then be combined with the estimate for $\left|\{(x,y)|\, \pi(x) = y \text{ and } \pi(y) = x\}\right|$ to obtain an estimate for the number of transpositions in π.

3.2 An infinity-vs-$\Theta(n)$ separation for a modification of Simon's problem

We now show how to modify Simon's problem [12] to obtain an infinity-vs-$\Theta(n)$ separation between the classical and quantum query complexities. Recall for Simon's problem, we are given oracle access to a function $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ and $f(x) = f(y)$ if and only if $x = y + a$ for some fixed element $a \in \mathbb{Z}_2^n$ and our task is to determine a. Classically, exponentially many queries are required; however, quantumly at each step we learn a vector which is orthogonal to a so that the expected number of queries required is $\Theta(n)$. The crucial point here is that this algorithm will return a vector orthogonal to a for any f which is constant and distinct on the cosets $\{x, x + a\}$, so if f changes between calls to the oracle and a does not, then the quantum algorithm will not be affected.

Our randomized oracle is defined as follows. Fix some unknown $a \in \mathbb{Z}_2^n$. Then construct an oracle $\mathcal{O}_a : |x\rangle |y\rangle \mapsto |x\rangle |y + f(x)\rangle$ where $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ is selected uniformly at random at each call subject to the constraint that $f(x) = f(y)$ if and only if $x = y + a$. The problem is then to determine a.

Classically, this cannot be done since each query to the oracle results in a random number; however, the quantum algorithm still requires only $\Theta(n)$ queries.

3.3 An infinity-vs-one separation for the hidden linear structure problem

Beaudrap, Cleve and Watrous [15] introduced the hidden linear structure problem where we are given a blackbox which performs the mapping $|x\rangle|y\rangle \mapsto |x\rangle|\pi(y+sx)\rangle$ where $\pi \in S_q$ and $s \in GF(q)$ for $q = 2^n$. The problem is to find s. By extending quantum Fourier transforms to $GF(q)$, Beaudrap, Cleve and Watrous [15] show that this problem can be solved exactly using a single quantum query but classical algorithms require $\Omega(\sqrt{q})$ queries to determine s. They are able to achieve such a query complexity separation by using a non-standard (but still deterministic) oracle model. In the 10 years since their paper, it is still an open question whether such separations are possible in the standard oracle model.

We propose the following randomized variant of their oracle problem. Fix some (unknown) $s \in GF(q)$. Then define the oracle by $\mathcal{O}_s : |x\rangle|y\rangle \mapsto |x\rangle|\pi(y+sx)\rangle$ where π is elected uniformly at random for each query. The goal is still to determine s. Since the quantum algorithm only uses one query it is unaffected by this change; however, classically the output of the oracle is completely random at each query so we obtain an infinity-vs-one separation.

3.4 Generalized construction from problems where one classical query is useless but one quantum query is not useless

The previous three separations are examples of a more general phenomenon in which randomness can be used to amplify a modest quantum-vs-classical query separation into an unbounded one.

Let $\{C_i\}$ be disjoint classes of functions such that a single classical query to the deterministic classical oracle $\mathcal{O}_f : |x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$ is useless (ie, yields no information) for determining which C_i contains f but that this can be determined by a single quantum query with bounded (or one-sided) error. Let $\mathcal{O}_i$ be the oracle which selects a $f \in C_i$ using the prior distribution on f and acts by bitwise XOR according to f. Suppose we are given an oracle $\mathcal{O}_i$ and the problem is to determine i. It can be shown (see Section 6) that this is impossible classically even with an unlimited number of queries but is possible with bounded error using only one quantum query.

This amplification works whenever a single quantum query can solve the problem and a single classical query provides no information at all about a problem. However, as we have seen with Simon's problem, it is sufficient for a single quantum query to make progress towards solving the problem, e.g. learning a random vector orthogonal to a. Note that this construction also works if the original oracle has internal randomness which is used to select a permutation. In Section 6, we will show how to generalize this technique to amplify any separation between classical and quantum query complexities into an infinity-vs-one separation (though this comes at the price of increased probability of error).

3.5 The problem of distinguishing involutions with no fixed points from cycles cannot be generalized to k-wise independent permutations

Consider the oracles $\mathcal{O}_{INV}$ and $\mathcal{O}_{CYC}$. As mentioned before, querying either oracle for a single value results in a random output so the permutations in the sets INV and CYC are essentially[1] 1-wise independent permutations. However, the permutations in these sets are not 2-wise independent since a cycle cannot contain a transposition and in the case of an involution with no fixed points $\pi(\pi(x))$ is determined once $\pi(x)$ is known. An interesting question is then if the algorithm can be generalized to the case of two disjoint sets A and B of k-wise independent permutations for $k \geq 2$. For $\pi \in S$, let $\mathcal{O}_\pi$ denote the version of $\mathcal{O}_S$ which has been derandomized by forcing the permutation selected to always be π. Since $k \geq 2$, two classical queries to $\mathcal{O}_\pi$ will not yield any information about whether π was selected from A or B. Meyer and Pommersheim [4] showed that if $2k$ classical queries to a deterministic oracle yield no information (in which case they are said to be useless) then k quantum queries are also useless. This shows that a single quantum query to $\mathcal{O}_\pi$ is useless so repeated queries to the oracle $\mathcal{O}_S$ each followed immediately by a POVM measurement cannot

[1] This is a slight abuse of terminology since for the permutations to truly be 1-wise independent it would be necessary for every value x to be mapped by a random permutation to a uniformly random value in $[N]$ whereas in this case it is mapped to a uniformly random value in $[N] \setminus \{x\}$.

solve the generalized problem. The question now becomes whether it is possible to solve this problem by making a POVM measurement after making multiple queries to $\mathcal{O}_S$. We now generalize the result of Meyer and Pommersheim to the case of oracles with internal randomness. In particular, this shows that no quantum algorithm can distinguish $\mathcal{O}_A$ from $\mathcal{O}_B$ so that the generalized problem cannot be solved by any algorithm: either classical or quantum.

4 Generalizations

We now explore generalizations of the loops problem and investigate their quantum speedups.

4.1 Coset equivalence testing

Let π_1, π_2 be chosen as in the loops problem. Now consider the permutations σ on $[N]$ such that $\pi_1 = \pi_2\sigma$. In the case where $\pi_1[N/2] = \pi_2[N/2]$, it must be that $\sigma \in S_{[N/2]} \oplus S_{(N/2,N]}$. However, if $\pi_1[N/2] \cap \pi_2[N/2] = \emptyset$ then $\sigma[N/2] = (N/2, N]$ and $\sigma(N/2, N] = [N/2]$ so that $\sigma \in \phi(S_{[N/2]} \oplus S_{(N/2,N]})$ where ϕ is any permutation that swaps the elements of $[N/2]$ and $(N/2, N]$. The loops problem is then to determine the coset $S_{[N/2]} \oplus S_{(N/2,N]}$ or $\phi(S_{[N/2]} \oplus S_{(N/2,N]})$ to which $\sigma = \pi_2^{-1}\pi_1$ belongs. Equivalently, the loops problem is to decide if $\sigma \in S_{[N/2]} \oplus S_{(N/2,N]}$ which is the case precisely when π_1 and π_2 are in the same coset. This motivates the following coset equivalence problem. Let G be a group where $H \leq G$. Let $g_1, g_2 \in G$ in analogy to the permutations π_1 and π_2 in the loops problem. The problem is to determine if g_1 and g_2 are in the same coset of G/H. More precisely, the formulation is as follows:

- G and H are given as blackbox groups and $|H\rangle$ can be prepared efficiently
- Inverses cannot be calculated
- An oracle $\mathcal{O}$ is provided (see Algorithm 8) which randomly multiplies the target $g \in G$ by g_1 or g_2
- The goal is to decide if $g_1H = g_2H$

The requirement that $|H\rangle$ can be initialized efficiently is not too restrictive; for instance, if $\langle K\rangle = H \leq S_n$ then the state $|H\rangle$ can be created in time polynomial in $|K|$ and n [13].

Algorithm 8 The oracle for coset equivalence testing

```
Flip an internal fair coin
if the outcome is heads then
    Multiply the target by g_1 on the left
else
    Multiply the target by g_2 on the left
end if
```

This problem may then be solved in the same way as the loops problem as shown in Algorithm 9.

Algorithm 9 The quantum algorithm for the coset equivalence problem

```
Prepare the pair of states |α⟩ and |β⟩ by applying the oracle O to |H⟩
Apply the swap test to |α⟩ and |β⟩
if the swap test returns "symmetric" then
    Return "g_1H = g_2H"
else
    Return "g_1H ≠ g_2H"
end if
```

Note that the restriction that inverses cannot be calculated in G is rather artificial; however, this issue can be resolved by generalizing the problem to orbit cosets.

4.2 Orbit coset equivalence testing

Section 4.1 showed how to test if two elements of a group G described by an oracle are in the same coset. However, the loops problem is not a special case of coset equivalence testing since $[N]$ is not a group. The coset equivalence testing problem can be thought of as a group G acting on an isomorphic group G' in the natural way. We now generalize this to the case of a group acting on a set. Let G be a group which acts faithfully on a set X and has a subgroup H. Let $g_1, g_2 \in G$ be specified by an oracle which acts randomly on the target (an element of X) using g_1 or g_2. The precise formulation is as follows:

- G and Hx are given as part of the input and the description of Hx is powerful enough that $|Hx\rangle$ can be prepared efficiently
- An oracle $\mathcal{O}$ is provided (see Algorithm 10) which randomly acts on the target $y \in X$ by g_1 or g_2
- The goal is to decide if $g_1 Hx = g_2 Hx$

We can think of the action of element of H as a permutation of X; thus, we may regard H as a subgroup of S_X. Let $\langle K \rangle = H$. The state $|Hx\rangle$ can then be prepared in time polynomial in $|K|$ and $|X|$ [13].

Algorithm 10 The oracle for orbit coset equivalence testing

```
Flip an internal fair coin
if the outcome is heads then
    Act on the target by g_1
else
    Act on the target by g_2
end if
```

Algorithm 11 solves this problem in the same way as before.

Algorithm 11 The quantum algorithm for the orbit coset equivalence problem

```
Prepare the pair of states |α⟩ and |β⟩ by applying the oracle O to |Hx⟩
Apply the swap test to |α⟩ and |β⟩
if the swap test returns "symmetric" then
    Return "g_1 Hx = g_2 Hx"
else
    Return "g_1 Hx ≠ g_2 Hx"
end if
```

Note that both the loops problem and the coset equivalence problem may be thought of as special cases of this problem by choosing the correct group action. Thus, the separation between classical and quantum query complexities for the loops problem also applies to the orbit coset equivalence problem.

Corollary 5. *There exists an orbit coset equivalence testing problem such that there is an $\Omega(\sqrt{N})$ vs $O(1)$ separation between the classical and quantum bounded-error query complexities*

5 Uselessness for oracles with internal randomness

We now turn to the general problem of when some number of queries are *useless* for solving an oracle problem. Equivalently we can ask when it is possible to answer an oracle problem with any positive advantage over guessing. The minimum number of queries required to achieve nonzero advantage has also been called the *unbounded error* query complexity (see e.g. [5]) by analogy with **PP**, the complexity class in which the correct answer needs to be output only with probability $> 1/2$.

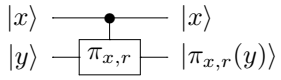

Figure 4: An oracle in the internal randomness model

To define oracle problems, we use a slightly more compact notation than in previous sections. An oracle π is defined by a collection of permutations $\pi_{x,r} \in S_M$, where $x \in [N]$ is input by the algorithm, and r is the internal randomness which is distributed according to R. Overloading notation, we say that if the oracle is queried k times, then $\boldsymbol{r} = (r_1, \ldots, r_k)$ is distributed according to R^k, which may not necessarily be an i.i.d. distribution.

To describe the problem we want to solve, we follow the notation of Meyer and Pommersheim[1] while adding internal randomness to the oracle. We are promised that our oracle belongs to a set C, which in general may be a strict subset of all functions from $[N] \times \mathrm{supp}(R)$ to $[M]$. The set C is partitioned into sets $\{C_j\}$, and our goal is to determine which C_j contains π. By an abuse of notation, we say that C is our oracle problem. Queries are made to an oracle $\mathcal{O}_\pi$ which acts as shown in figure 4. We will assume that π is distributed according to a known distribution μ.

Before stating our own results, we describe the main result of [4]. In their model there is no internal randomness, so the action of the oracle is simply $\pi_x \in S_M$ for each $x \in [N]$. If $\boldsymbol{x} = (x_1, \ldots, x_k)$ and $\boldsymbol{y} = (y_1, \ldots, y_k)$, then define $\pi_{\boldsymbol{x}}(\boldsymbol{y}) = (\pi_{x_1}(y_1), \ldots, \pi_{x_k}(y_k))$. Their result may then be stated as follows.

Definition 6 (Classical uselessness[4])**.** *k classical queries are useless for the oracle problem C and distribution μ if for all $\boldsymbol{x} \in [N]^k$, $\boldsymbol{y} \in [M]^k$ and $\boldsymbol{z} \in [M]^k$,*

$$\Pr(\pi \in C_j \mid \pi_{\boldsymbol{x}}(\boldsymbol{y}) = \boldsymbol{z}) = \Pr(\pi \in C_j) \tag{5}$$

for all j, where the probabilities over π are taken with respect to μ.

Definition 7 (Quantum uselessness[4])**.** *k quantum queries are useless for the oracle problem C and distribution μ if for any k-query quantum algorithm run on any initial state and any POVM measurement $\{M_s\}$ which is made on the output of the algorithm*

$$\Pr(\pi \in C_j \mid s) = \Pr(\pi \in C_j) \tag{6}$$

for all j and s, where the probabilities over π are taken with respect to μ.

The main result of [4] is the following theorem.

Theorem 8 (Classical uselessness implies quantum uselessness[4])**.** *For any deterministic oracle problem C, μ, if $2k$ classical queries are useless then k quantum queries are useless.*

We will give an alternate proof of this theorem, establish a converse, and generalize it to oracles with internal randomness.

5.1 Definitions of Classical Uselessness

In order to characterize uselessness for oracles with internal randomness, we first need to extend the definitions to this case. As above, we define $\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = (\pi_{x_1,r_1}(y_1), \ldots \pi_{x_k,r_k}(y_k))$. One natural definition of uselessness in this setting is that a classical algorithm ignorant of the oracle's internal randomness should not be able to gain any nontrivial advantage in learning which C_j contains π.

Definition 9 (Weak classical uselessness)**.** *If C, μ is an oracle problem, then k classical queries are weakly useless if for all $\boldsymbol{x} \in [N]^k$ and all $\boldsymbol{y}, \boldsymbol{z} \in [M]^k$,*

$$\Pr(\pi \in C_j \mid \pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{z}) = \Pr(\pi \in C_j) \tag{7}$$

for all j, where the probabilities over π are with respect to μ and over $\boldsymbol{r}$ are with respect to R^k.

It is easy to see that if $2k$ classical queries are weakly useless then k quantum queries need not be useless since Algorithm 7 is a counterexample. A much stronger definition of uselessness would be to allow the classical algorithm to see, or equivalently to choose, the internal random bits used by the oracle.

Definition 10 (Strong classical uselessness)**.** *If C, μ is an oracle problem, then k classical queries are strongly useless if for all $\boldsymbol{x} \in [N]^k$, $\boldsymbol{y}, \boldsymbol{z} \in [M]^k$ and all possible values $\boldsymbol{r} \in \mathrm{supp}(R^k)$,*

$$\Pr(\pi \in C_j \mid \pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{z}) = \Pr(\pi \in C_j) \tag{8}$$

for all j, where π is distributed according to μ.

We will see later that strong classical uselessness for $2k$ queries is sufficiently powerful to imply quantum uselessness for k queries. However, it is in fact too strong, so the definition must be weakened as follows.

Definition 11 (Pairwise classical uselessness)**.** *If C, μ is an oracle problem, then $2k$ classical queries are pairwise useless if for all $\boldsymbol{x}, \boldsymbol{x}' \in [N]^k$ and $\boldsymbol{y}, \boldsymbol{y}', \boldsymbol{z}, \boldsymbol{z}' \in [M]^k$*

$$\Pr(\pi \in C_j \mid \pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{z}, \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}') = \boldsymbol{z}') = \Pr(\pi \in C_j) \tag{9}$$

for all j where π is distributed according to μ and $\boldsymbol{r}$ is distributed according to R^k.

This definition ensures that each pair of query values (x_i, x_i') shares the same random seed r_i. We will see later that this corresponds precisely (in the unbounded error setting) to the power of quantum queries, because the density matrix resulting from a quantum query depends on only one random seed, while the different row and column indices can interrogate two different choices of x, y.

We observe that strong classical uselessness implies both weak classical uselessness and pairwise classical uselessness. However, weak classical uselessness and pairwise classical uselessness are not comparable: there exist problems which satisfy weak classical uselessness but not pairwise classical uselessness and vice versa. Section 3.1 gives an example where two classical queries are weakly useless but not pairwise useless. For an example of a problem where two classical queries are not weakly useless but are pairwise useless, let C be the set of all balanced binary functions and let f is chosen uniformly at random from C. Consider the task of determining the function implemented by the oracle which acts for the i^{th} query by $|x\rangle \mapsto |x \oplus f(r_i)\rangle$ where r_i is the i^{th} random seed; let r_1 be uniformly distributed in $\{0, 1\}$ and let $r_i = 0$ for $i \geq 2$. Clearly, two classical queries with the random seeds r_1 and r_2 determine f. However, two classical queries which share the random seed r_1 yield no useful information.

We now consider the dependence of uselessness on the distribution over the oracles. Consider an oracle problem C, μ. Each of the definitions of uselessness (weak, strong, pairwise or quantum) states that the condition $\Pr(\pi \in C_j \mid Q) = \Pr(\pi \in C_j)$ holds for all j and Q (where the values for Q depend on the definition we consider). Since we assume that $\mathrm{supp}(\mu) = C$, we may rewrite this as $\Pr(Q \mid \pi \in C_j) = \Pr(Q)$. This is equivalent to the assertion that for any Q, $\Pr(Q \mid \pi \in C_i) = \Pr(Q \mid \pi \in C_j)$ for all i and j. Now, $\Pr(Q \mid \pi \in C_j)$ depends only on R^k and the conditional distribution $\mu(\pi \mid \pi \in C_j)$. Thus, uselessness does not depend on the distribution over the classes $\mu(\pi \in C_j)$.

We will show by example that uselessness can depend on the conditional distribution of the oracle within each class. Let C be the set of all standard oracles for a function $f : [N] \to \{0, 1\}$. Let us partition C into C_0 and C_1 and b is the parity of the functions in C_b. Suppose that μ is the uniform distribution over C. Clearly, $N - 1$ classical queries are useless. By Corollary 13, $\left\lfloor \frac{N-1}{2} \right\rfloor$ quantum queries are useless. Suppose that we design μ such that $\mu(C_0) = \mu(C_1)$ but $f(1)$ is the parity of f with probability greater than $1/2$. Then clearly one query (classical or quantum) is not useless.

Our main result in this section is the following equivalence:

Theorem 12. *For any oracle problem C, μ, k quantum queries are useless if and only if $2k$ classical queries are pairwise useless.*

For deterministic oracles, weak, pairwise and strong classical uselessness are all the same. In this case, Theorem 12 can be simplified to the following strengthening of Theorem 8.

Corollary 13. *For any deterministic oracle problem C, μ, k quantum queries are useless if and only if $2k$ classical queries are useless.*

As mentioned in the introduction, corollary 13 was proved for the parity function by [3], and for the standard oracle model with binary functions (i.e. $M = 2$) by [5]. Since their proof technique used the polynomial method [14], it does not readily generalize even to non-binary functions.

5.2 Encoding oracles in states

In this section we will prove Theorem 12. Our strategy will be to show that in the unbounded-error setting, the optimal algorithms for query complexity make a series of fixed queries and then measure the resulting states. The key ingredient is to show that oracles can be encoded in states in a way that is perfectly efficient in terms of queries (i.e. one oracle call creates one state, and one state simulates one oracle call), albeit at a cost of producing algorithms the output "I don't know" most of the time.

We define these encodings first for deterministic oracles.

Definition 14. *Let $\mathcal{O}_\pi$ be a deterministic permutation oracle that maps $|x, y\rangle \in \mathbb{C}^N \otimes \mathbb{C}^M$ to $|x, \pi_x(y)\rangle$. Then define the* encoding *of π to be*

$$|\psi_\pi\rangle := \frac{1}{\sqrt{NM}} \sum_{\substack{x\in[N]\\ y\in[M]}} |x\rangle^X |y\rangle^Y |\pi_x(y)\rangle^Z \tag{10}$$

Here X, Y, Z label different registers for notational convenience.

Clearly one use of $\mathcal{O}_\pi$ allows the creation of one copy of $|\psi_\pi\rangle$; simply prepare the state $\frac{1}{\sqrt{NM}} \sum_{x,y} |x\rangle^X |y\rangle^Y |y\rangle^Z$ and apply $\mathcal{O}_\pi$ to registers XZ. We will see shortly that one copy of $|\psi_\pi\rangle$ can in turn simulate one use of $\mathcal{O}_\pi$, albeit with a very high, but heralded, failure probability. Before proving this result, we show how Definition 14 generalizes to oracles with internal randomness.

Definition 15. *Let $\mathcal{O}_\pi$ be an oracle whose action is defined by*

$$\mathcal{O}_\pi(|x\rangle\langle x'| \otimes |y\rangle\langle y'|) = \mathop{\mathbb{E}}_{r\sim R} |x\rangle\langle x'| \otimes |\pi_{x,r}(y)\rangle\langle \pi_{x',r}(y')|$$

For each r, define the deterministic oracle $\mathcal{O}_{\pi,r}$ by

$$\mathcal{O}_{\pi,r} |x, y\rangle = |x, \pi_{x,r}(y)\rangle$$

and define the encoding for fixed r to be

$$|\psi_{\pi,r}\rangle := \frac{1}{\sqrt{NM}} \sum_{\substack{x\in[N]\\ y\in[M]}} |x\rangle^X |y\rangle^Y |\pi_{x,r}(y)\rangle^Z . \tag{11}$$

Now we define encodings of oracles with randomness.

Definition 16. *If $\mathcal{O}_\pi$ is an oracle with internal randomness, then define the* encoding *of $\mathcal{O}_\pi$ to be*

$$\rho_\pi := \mathop{\mathbb{E}}_r \psi_{\pi,r}. \tag{12}$$

In this last definition, we use the convention that $\psi := |\psi\rangle\langle\psi|$.

The utility of considering encodings comes from the following operational equivalence.

Theorem 17. *1. One use of $\mathcal{O}_\pi$ can create one copy of ρ_π.*

2. It is possible to consume one copy of ρ_π and simulate $\mathcal{O}_\pi$ with success probability $1/NM^2$. The simulation outputs a classical flag indicating success or failure.

In both cases, the run time required is linear in the number of qubits, i.e. $O(\log NM)$.

We point out that in the simulation, failure destroys not only the encoding, but also the state input to the oracle. Nevertheless, this simulation is enough to distinguish the case when k queries are useless from the case when they are not.

Additionally, Theorem 17 is stated implicitly in terms of a distribution R. In the case of k correlated queries, we have the following variant:

Theorem 18. *1. k uses of $\mathcal{O}_\pi$ (correlated according to R^k) can create*

$$\rho_\pi^k := \mathop{\mathbb{E}}_{\boldsymbol{r}\sim R^k}[\psi_{\pi,r_1} \otimes \cdots \otimes \psi_{\pi,r_k}].$$

2. It is possible to consume ρ_π^k and simulate k uses of $\mathcal{O}_\pi$ (correlated according to R^k) with success probability $1/N^k M^{2k}$, again with a flag indicating success or failure.

As an corollary, for correlated internal randomness in the unbounded-error scenario, we can permit algorithms to make the k oracle calls in any order.

We will prove Theorem 18 only, since it subsumes Theorem 17.

Proof. To create ρ_π^k, we simply apply $\mathcal{O}_\pi$ k times to $\left(\frac{1}{\sqrt{NM}}\sum_{x,y} |x\rangle^X |y\rangle^Y |y\rangle^Z\right)^{\otimes k}$.

For the second reduction, suppose we are given a copy of ρ_π^k and would like to apply $\mathcal{O}_\pi$ to simulate the i^{th} query of some algorithm. If we condition on $\boldsymbol{r}$, then ρ_π^k becomes the state $\psi_{\pi,r_1} \otimes \cdots \otimes \psi_{\pi,r_k}$. We will use the i^{th} component of this state to simulate our query.

Suppose we want to simulate the action of $\mathcal{O}_{\pi,r_i}$ on the state $|x'\rangle^{X'} |y'\rangle^{Y'}$. Define

$$A = \sum_{x\in[N]} |x\rangle^X \langle x,x|^{XX'} \otimes \frac{1}{\sqrt{M}} \sum_{y\in[M]} \langle y,y|^{YY'}$$

Since $AA^\dagger = \sum_x |x\rangle\langle x| = I_N$, it follows that $A^\dagger A \leq I_{N^2M^2}$ and $\{A, \sqrt{I - A^\dagger A}\}$ comprise valid Kraus operators for a quantum operation. Our simulation will apply this operation, with outcome A labeled success, and $\sqrt{I - A^\dagger A}$ labeled failure.

Upon outcome $\sqrt{I - A^\dagger A}$, the algorithm declares failure. If this occurs at any step of a multi-query algorithm, then the algorithm should guess j according to the *a priori* distribution μ. Thus, for the purposes of determining whether the algorithm outperforms the best guessing strategy, it then suffices to consider only the cases when outcome A occurs.

Upon outcome A, $|\psi_{\pi,r_i}\rangle^{XYZ} |x'\rangle^{X'} |y'\rangle^{Y'}$ is mapped to the (unnormalized) state $\frac{1}{\sqrt{NM^2}} |x'\rangle^X |\pi_{x,r_i}(y')\rangle^Z$. Since the normalization is independent of the input, this means that A occurs with probability $1/NM^2$ regardless of the input state. Conditioned on this outcome, the resulting map is precisely the action of $\mathcal{O}_{\pi,r_i}$.

We say that the overall algorithm succeeds when each of the k queries succeeds. Since each query independently succeeds with probability $1/NM^2$, the overall algorithm succeeds with probability $1/N^kM^{2k}$. □

Armed with our notion of encoding, it becomes straightforward to characterize when k quantum queries are useless.

Corollary 19. *Define $\sigma_j := \mathbb{E}_{\pi \in C_j} \rho_\pi^k$. Then k quantum queries are useless if and only if all the σ_j are the same.*

Proof. By Theorem 18, any k-query algorithm can WLOG create ρ_π^k, resulting in the state σ_j if π is drawn randomly from C_j. The algorithm then proceeds to determine which σ_j it holds, using no further oracle queries. If all the σ_j are equal, then it can learn nothing about j. Conversely, if some σ_j is different from the others, then there is a measurement which will be able to guess j with positive advantage. □

To conclude the proof of Theorem 12, observe that the quantity on the LHS of (9) is precisely

$$\operatorname{tr}\left(|\boldsymbol{x}\rangle\langle\boldsymbol{x'}| \otimes |\boldsymbol{y}\rangle\langle\boldsymbol{y'}| \otimes |\boldsymbol{z}\rangle\langle\boldsymbol{z'}|\right) \sum_{\pi \in C_j} \mu(\pi)\rho_\pi^k \tag{13}$$

$$= \operatorname{tr}\left(|\boldsymbol{x}\rangle\langle\boldsymbol{x'}| \otimes |\boldsymbol{y}\rangle\langle\boldsymbol{y'}| \otimes |\boldsymbol{z}\rangle\langle\boldsymbol{z'}|\right)\sigma_j \tag{14}$$

which will be independent of j for all $\boldsymbol{x}, \boldsymbol{x'}, \boldsymbol{y}, \boldsymbol{y'}, \boldsymbol{z}, \boldsymbol{z'}$ if and only if all of the σ_j are identical.

Combined with Corollary 19, this completes the proof of Theorem 12.

Theorem 20. *Suppose that for some oracle problem k classical queries are weakly useless but k quantum queries are not useless. Then there exists an oracle problem in which this separation holds where the oracle acts by bitwise XOR.*

Proof. Consider an oracle problem C, μ for which k classical queries are weakly useless but k quantum queries are not useless. The oracle acts by $\mathcal{O} : |x\rangle|y\rangle \mapsto |x\rangle|\pi_{x,r_i}(y)\rangle$ on the i^{th} call. We can define a new oracle $\mathcal{O}' : |x\rangle|y\rangle|z\rangle \mapsto |x\rangle|y\rangle|z \oplus \pi_{x,r_i}(y)\rangle$. Our new oracle $\mathcal{O}'$ can be used to prepare the encoding for $\mathcal{O}$ so k queries to $\mathcal{O}'$ can simulate any quantum algorithm which uses k queries to $\mathcal{O}$. Classically, $\mathcal{O}'$ can be simulated using $\mathcal{O}$ so we conclude that k classical queries to $\mathcal{O}'$ are weakly useless. We have proven the following: □

6 Amplifying query complexity separations

We now leverage our results to obtain a general method of amplifying any separation between classical and quantum query complexities. Let C, μ be an oracle problem where C is partitioned into $\{C_i\}$ and r_j is the j^{th} random seed. For each $\pi \in C$, we have an oracle $\mathcal{O}_\pi$. Suppose that for this problem there is a separation between the classical and quantum unbounded query complexities. Then for some k, k classical queries are weakly useless but k quantum queries are not useless. Let us define the oracle $\mathcal{O}_i : |x_1\rangle\cdots|x_k\rangle|y_1\rangle\cdots|y_k\rangle \mapsto |x_1\rangle\cdots|x_k\rangle|\pi_{x_1,r_1}(y_1)\rangle\cdots|\pi_{x_k,r_k}(y_k)\rangle$ where π is selected from C_i according to μ (this is done independently for each query), $\boldsymbol{r}$ is distributed according to R^k and a fresh random seed $\boldsymbol{r}$ is used for every query to $\mathcal{O}_i$. Consider the problem of determining i where the oracle $\mathcal{O}_i$ is given with probability $\mu(\pi \in C_i)$.

Theorem 21. *Any number of classical queries to the oracle $\mathcal{O}_i$ is weakly useless for determining i; thus no classical algorithm can determine i with unbounded error no matter how many queries are made.*

Proof. Clearly, a single query to $\mathcal{O}_i$ is equivalent to k queries to the original oracle which are weakly useless by assumption. We conclude that a single classical query to the new oracle is weakly useless. We now show that ℓ classical queries are weakly useless for any $\ell \geq 1$. Let $\boldsymbol{x}_j \in [N]^k$, $\boldsymbol{y}_j, \boldsymbol{z}_j \in [M]^k$ and let each $\boldsymbol{r}_j$ be sampled independently from R^k where $1 \leq j \leq \ell$. We must prove that

$$\Pr(i \mid \pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = \boldsymbol{z}_j, j = 1, \ldots, \ell) = \Pr(i) \tag{15}$$

where each π^j is sampled independently from C_i according to μ. This condition is equivalent to

$$\Pr(\pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = z_j, j = 1, \ldots, \ell \mid i) = \Pr(\pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = \boldsymbol{z}_j, j = 1, \ldots, \ell) \tag{16}$$

Note that by construction, $\Pr(\pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = z_j, j = 1,\ldots,\ell \mid i) = \prod_j \Pr(\pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = \boldsymbol{z}_j \mid i)$. By our assumption that k classical queries to the original oracle are weakly useless, we have that $\Pr(i \mid \pi_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = \boldsymbol{z}_j) = \Pr(i)$ or equivalently $\Pr(\pi_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = \boldsymbol{z}_j \mid i) = \Pr(\pi_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = \boldsymbol{z}_j)$. Therefore,

$$\Pr(\pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = \boldsymbol{z}_j, j = 1,\ldots,\ell) = \sum_i \Pr(\pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = z_j, j = 1,\ldots,\ell \mid i) \Pr(i) \tag{17}$$

$$= \sum_i \left(\prod_j \Pr(\pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = z_j \mid i) \right) \Pr(i) \tag{18}$$

$$= \sum_i \left(\prod_j \Pr(\pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = z_j) \right) \Pr(i) \tag{19}$$

$$= \prod_j \Pr(\pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = z_j) \tag{20}$$

$$= \prod_j \Pr(\pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = z_j \mid i) \tag{21}$$

$$= \Pr(\pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = z_j, j = 1,\ldots,\ell \mid i) \tag{22}$$

which is the desired result. □

We conclude that no matter how many classical queries are made to $\mathcal{O}_i$, no information is obtained about i. On the other hand, we have the following result:

Theorem 22. *A single quantum query to $\mathcal{O}_i$ is* not *useless for determining i; equivalently, the unbounded-error quantum query complexity is* 1.

Proof. One can use a single quantum query to $\mathcal{O}_i$ to construct the state ρ^k_π as described in Theorem 18. Applying Theorem 18, this state may be used to guess i with higher probability than random guessing since k quantum queries are not useless. □

Thus, we have constructed an infinity-vs-one separation in unbounded-error classical and quantum query complexities from an arbitrary initial separation. We now show how to obtain a infinity-vs-one separation in the bounded-error regime from an arbitrary separation between unbounded-error classical and quantum query complexities. Consider the oracle $\mathcal{O}_i$ as defined above. By Theorem 22, there exists a single-query quantum algorithm A, a POVM $\{M_s\}$ and an i' such that for some s, $\Pr(i = i' \mid s) > \Pr(i = i')$. Equivalently,

$$\Pr(s \mid i = i') > \Pr(s) \tag{23}$$

$$\Pr(s \mid i = i')(1 - \Pr(i = i')) > \Pr(s \mid i \neq i') \Pr(i \neq i') \tag{24}$$

$$\Pr(s \mid i = i') > \Pr(s \mid i \neq i') \tag{25}$$

$$\Pr(s \mid i = i') = \Pr(s \mid i \neq i') + \epsilon \tag{26}$$

for some $\epsilon > 0$. Consider the problem of deciding if $i = i'$ by querying $\mathcal{O}_i$. By running A some large number of times T and using majority voting and Chernoff bounds, we may decide if $i = i'$ with bounded error. Although T may be quite large, the gap is large since it is a separation between an infinite number of classical queries and a finite number of classical queries.

Corollary 23. *The bounded-error quantum query complexity of deciding if $i = i'$ using $\mathcal{O}_i$ is finite.*

By Theorem 21, $\Pr(i \mid \pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = z_j, j = 1,\ldots) = \Pr(i)$ for all $\ell \geq 1$ and $\boldsymbol{x}_j \in [N]^k$, $\boldsymbol{y}_j, \boldsymbol{z}_j \in [M]^k$. Thus, $\Pr(i = i' \mid \pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = z_j, j = 1,\ldots) = \Pr(i = i')$ and $\Pr(i \neq i' \mid \pi^j_{\boldsymbol{x}_j,\boldsymbol{r}_j}(\boldsymbol{y}_j) = z_j, j = 1,\ldots) = \Pr(i \neq i')$ so ℓ queries are weakly useless for deciding if $i = i'$.

Corollary 24. *Any number of classical queries to the oracle $\mathcal{O}_i$ is weakly useless for deciding if $i = i'$; thus no classical algorithm can decide if $i = i'$ with unbounded error no matter how many queries are made.*

We can construct a new oracle $\mathcal{O}'_i$ which simulates T queries to $\mathcal{O}_i$ using an independent random seed for each query. From this we obtain the following.

Corollary 25. *The bounded-error quantum query complexity of deciding if $i = i'$ using $\mathcal{O}_i$ is* 1.

Corollary 26. *Any number of classical queries to the oracle $\mathcal{O}'_i$ is weakly useless for deciding if $i = i'$; thus no classical algorithm can decide if $i = i'$ with unbounded error no matter how many queries are made.*

Thus, we have constructed an infinity-vs-one separation between the bounded-error quantum query complexity and the unbounded-error classical query complexity from an arbitrary initial separation. This comes at the price of large inputs for the constructed oracle.

Acknowledgments

DJR was funded by NSF grant CCF-0916400 and by the DoD through an NDSEG fellowship. AWH was funded by NSF grants CCF-0916400, CCF-0829937, PHY-0803478, DARPA QuEST contract FA9550-09-1-0044 and the IARPA MUSIQC and QCS contracts. We thank an anonymous reviewer for suggesting the infinity-vs-one separation for the problem in [15].

A Alternate proofs of uselessness

In this appendix, we present alternate proofs of various uselessness theorems. These proofs do not rely on the idea of encoding oracles into states, but instead give direct arguments. First we prove that pairwise classical uselessness implies quantum uselessness.

Proof. The proof is an extension of the technique used by Meyer and Pommersheim. Suppose that $2k$ classical queries are pairwise useless. Consider an oracle π which acts by $\mathcal{O}^i_\pi : |x, y, z\rangle \mapsto |x, \pi_{x,r_i}, z\rangle$ for the i^{th} query. Note that, as before, the r_i variables may obey an arbitrary joint distribution so different queries are not necessarily independent. Consider an arbitrary k-query quantum algorithm with initial state ρ_0 and POVM $\{M_s\}$. For the i^{th} query, the algorithm queries the oracle and then applies an arbitrary unitary transformation U_i. This yields the final state

$$\rho_\pi = U_k \mathcal{O}^k_\pi \dots U_1 \mathcal{O}^1_\pi \rho_0 {\mathcal{O}^1_\pi}^\dagger U_1^\dagger \dots {\mathcal{O}^k_\pi}^\dagger U_k^\dagger \tag{27}$$

Let us fix the random seed used for the i^{th} query as r_i. The final state is then

$$\rho_{\pi,\boldsymbol{r}} = U_k P_{r_k} \dots U_1 P_{r_1} \rho_0 P^\dagger_{r_1} U_1^\dagger \dots P^\dagger_{r_k} U_k^\dagger \tag{28}$$

where P_{r_i} denotes the permutative action $|x, y, z\rangle \mapsto |x, \pi_{x,r_i}(y), z\rangle$ of the oracle when the random seed is fixed to r_i. Let A be a matrix, $L = (x, y, z)$ and $L' = (x', y', z')$. Then

$$\left(P_{r_i} A P^\dagger_{r_i}\right)_{L,L'} = \left\langle x, \pi^{-1}_{x,r_i}(y), z \right| A \left| x', \pi^{-1}_{x',r_i}(y'), z' \right\rangle \tag{29}$$

$$= A_{\pi_{\cdot,r_i}(L), \pi_{\cdot,r_i}(L')} \tag{30}$$

where $\pi_{\cdot,r_i}(L) = (x, \pi^{-1}_{x,r_i}(y), z)$. Then the state after the $i + 1^{\text{th}}$ query (for the fixed values $\boldsymbol{r}$ of the seeds) is

$$\rho_{i+1,\boldsymbol{r}} = U_{i+1} P_{r_{i+1}} \rho_{i,\boldsymbol{r}} P_{r_{i+1}}^\dagger U_{i+1}^\dagger \tag{31}$$

so that the matrix elements are

$$(\rho_{i+1,\boldsymbol{r}})_{L,L'} = \sum_{K,K'} (U_{i+1})_{L,K} (\rho_{i,\boldsymbol{r}})_{\pi_{\cdot,r_{i+1}}(K),\pi_{\cdot,r_{i+1}}(K')} (U_{i+1}^\dagger)_{K',L'} \tag{32}$$

This value is a function of L, L', $\pi_{\cdot,r_{i+1}}(K)$ and $\pi_{\cdot,r_{i+1}}(K')$. Therefore, the final state $\rho_{\pi,\boldsymbol{r}} = \rho_{k,\boldsymbol{r}}$ may be written as

$$\rho_{\pi,\boldsymbol{r}} = \sum_I Q_I(\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}), \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}')) \tag{33}$$

where $I = (L_1, \ldots, L_k, L'_1 \ldots, L'_k)$. Let $\mathbb{E}_{\pi|\pi\in C_j}$ denote the expectation over π according to the distribution $\Pr(\pi \mid \pi \in C_j)$. Then for any j,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\pi|\pi\in C_j} \rho_{\pi,\boldsymbol{r}} &= \sum_I \mathop{\mathbb{E}}_{\pi|\pi\in C_j} Q_I(\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}), \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}')) && (34)\\
&= \sum_I \sum_{\boldsymbol{w},\boldsymbol{w}'} \mathop{\mathbb{E}}_{\pi|\pi\in C_j} Q_I(\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}), \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}'))[\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{w}, \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}') = \boldsymbol{w}']) && (35)\\
&\quad \text{where } \boldsymbol{w} = (w_1, \ldots, w_k) \text{ and } \boldsymbol{w}' = (w'_1, \ldots, w'_k)\\
&= \sum_I \sum_{\boldsymbol{w},\boldsymbol{w}'} Q_I(\boldsymbol{w}, \boldsymbol{w}') \mathop{\mathbb{E}}_{\pi|\pi\in C_j} [\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{w}, \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}') = \boldsymbol{w}']) && (36)\\
&= \sum_I \sum_{\boldsymbol{w},\boldsymbol{w}'} Q_I(\boldsymbol{w}, \boldsymbol{w}') \Pr(\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{w}, \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}') = \boldsymbol{w}' \mid \pi \in C_j) && (37)\\
& && (38)
\end{aligned}$$

Taking the expectation over the random seeds $\boldsymbol{r}$,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\pi|\pi\in C_j} \mathop{\mathbb{E}}_{\boldsymbol{r}} \rho_{\pi,\boldsymbol{r}} &= \sum_I \sum_{\boldsymbol{w},\boldsymbol{w}'} Q_I(\boldsymbol{w}, \boldsymbol{w}') \mathop{\mathbb{E}}_{\boldsymbol{r}} \Pr(\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{w}, \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}') = \boldsymbol{w}' \mid \pi \in C_j) && (39)\\
&= \sum_I \sum_{\boldsymbol{w},\boldsymbol{w}'} Q_I(\boldsymbol{w}, \boldsymbol{w}') \Pr(\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{w}, \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}') = \boldsymbol{w}' \mid \pi \in C_j) && (40)\\
&= \sum_I \sum_{\boldsymbol{w},\boldsymbol{w}'} Q_I(\boldsymbol{w}, \boldsymbol{w}') \Pr(\pi \in C_j \mid \pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{w}, \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}') = \boldsymbol{w}') && (41)\\
&\quad \cdot \frac{\Pr(\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{w}, \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}') = \boldsymbol{w}')}{\Pr(\pi \in C_j)} && (42)\\
&= \sum_I \sum_{\boldsymbol{w},\boldsymbol{w}'} Q_I(\boldsymbol{w}, \boldsymbol{w}') \Pr(\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{w}, \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}') = \boldsymbol{w}') && (43)\\
&\quad \text{by pairwise classical uselessness}\\
&= \mathop{\mathbb{E}}_{\pi} \mathop{\mathbb{E}}_{r} \sum_I \sum_{\boldsymbol{w},\boldsymbol{w}'} Q_I(\boldsymbol{w}, \boldsymbol{w}')[\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}) = \boldsymbol{w}, \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}') = \boldsymbol{w}'] && (44)\\
&= \mathop{\mathbb{E}}_{\pi} \mathop{\mathbb{E}}_{r} \sum_I Q_I(\pi_{\boldsymbol{x},\boldsymbol{r}}(\boldsymbol{y}), \pi_{\boldsymbol{x}',\boldsymbol{r}}(\boldsymbol{y}')) && (45)\\
&= \mathop{\mathbb{E}}_{\pi} \mathop{\mathbb{E}}_{\boldsymbol{r}} \rho_{\pi,\boldsymbol{r}} && (46)\\
& && (47)
\end{aligned}$$

Defining $\rho_\pi = \mathbb{E}_{\boldsymbol{r}} \rho_{\pi,\boldsymbol{r}}$, this may be written as

$$\mathop{\mathbb{E}}_{\pi|\pi\in C_j} \rho_\pi = \mathop{\mathbb{E}}_{\pi} \rho_\pi \tag{48}$$

Note that for a random $\pi \in C$, the state after running the algorithm is $\mathbb{E}_\pi \rho_\pi$ and for a random $\pi \in C_j$ the state is $\mathbb{E}_{\pi|\pi\in C_j} \rho_\pi$. Now, consider the probability that $\pi \in C_j$ given the measurement outcome s. We have

$$\Pr(\pi \in C_j \mid s) = \frac{\Pr(s \mid \pi \in C_j) \Pr(\pi \in C_j)}{\Pr(s)} \tag{49}$$

$$= \frac{\operatorname{tr} M_s \mathbb{E}_{\pi|\pi\in C_j} \rho_f}{\operatorname{tr} M_s \mathbb{E}_\pi \rho_\pi} \Pr(\pi \in C_j) \tag{50}$$

$$= \Pr(\pi \in C_j) \tag{51}$$

as claimed. □

Next, we prove that quantum uselessness implies classical uselessness, but in the special case of standard oracles that act via XOR but with internal randomness. Specifically, consider an oracle which acts by $\mathcal{O}_f^i : |x, y, z\rangle \mapsto |x, y \oplus f(x, r_i), z\rangle$ for the i^{th} query. As before, we allow the r_i variables to be drawn from an arbitrary joint distribution.

Proof. Suppose that k quantum queries are useless. This means that for any POVM $\{M_s\}$ and quantum algorithm run on any initial state ρ_0, $\Pr(f \in C_j \mid s) = \Pr(f \in C_j)$ for all j. Since $\Pr(f \in C_j \mid s) = \frac{\Pr(s|f\in C_j)\Pr(f\in C_j)}{\Pr(s)}$, this implies that

$$\Pr(s \mid f \in C_j) = \Pr(s) \tag{52}$$

for all j. Let us choose the initial state

$$\rho_0 = \left(\frac{1}{N} \sum_{x,x'} |x\rangle \langle x'| \otimes |0\rangle \langle 0| \right)^{\otimes k} \tag{53}$$

and the algorithm defined by the unitary operator $\bigotimes_{i=1}^k \mathcal{O}_f^i$. The result of running the algorithm assuming a particular function f and fixed seeds $\boldsymbol{r}$ is then

$$\rho_{f,\boldsymbol{r}} = \left(\bigotimes_{i=1}^k \mathcal{O}_f^i \right) \rho_0 \left(\bigotimes_{i=1}^k \mathcal{O}_f^i \right)^\dagger \tag{54}$$

$$= \frac{1}{N^k} \bigotimes_{i=1}^k \sum_{x,x'} |x, f(x, r_i)\rangle \langle x', f(x', r_i)| \tag{55}$$

For a particular function f, the state after running the algorithm is

$$\rho_f = \mathop{\mathbb{E}}_{\boldsymbol{r}} \rho_{f,\boldsymbol{r}} \tag{56}$$

$$= \frac{1}{N^k} \mathop{\mathbb{E}}_{\boldsymbol{r}} \bigotimes_{i=1}^k \sum_{x,x'} |x, f(x, r_i)\rangle \langle x', f(x', r_i)| \tag{57}$$

Now

$$\Pr(s) = \mathop{\mathbb{E}}_{f} \Pr(s \mid f) \tag{58}$$

$$= \operatorname{tr} M_s \mathop{\mathbb{E}}_{f} \rho_f \tag{59}$$

$$= \operatorname{tr} M_s \rho_C \tag{60}$$

Similarly,

$$\Pr(s \mid f \in C_j) = \mathop{\mathbb{E}}_{f \mid f \in C_j} \Pr(s \mid f) \tag{61}$$

$$= \operatorname{tr} M_s \mathop{\mathbb{E}}_{f \mid f \in C_j} \rho_f \tag{62}$$

$$= \operatorname{tr} M_s \rho_{C_j} \tag{63}$$

Since $\Pr(s \mid f \in C_j) = \Pr(f \in C_j)$, this implies that

$$\operatorname{tr} M_s(\rho_{C_j} - \rho_C) = 0 \tag{64}$$

for *all* POVMs $\{M_s\}$ which means that

$$\rho_{C_j} = \rho_C \tag{65}$$

$$\frac{1}{N^k} \mathop{\mathbb{E}}_{\boldsymbol{r}} \mathop{\mathbb{E}}_{f \mid f \in C_j} \bigotimes_{i=1}^{k} \sum_{x,x'} |x, f(x, r_i)\rangle \langle x', f(x', r_i)| = \frac{1}{N^k} \mathop{\mathbb{E}}_{\boldsymbol{r}} \mathop{\mathbb{E}}_{f} \bigotimes_{i=1}^{k} \sum_{x,x'} |x, f(x, r_i)\rangle \langle x', f(x', r_i)| \tag{66}$$

Equating the $((x_1, y_1, \ldots, x_k, y_k), (x'_1, y'_1, \ldots, x'_k, y'_k))$ elements of these matrices, we have that

$$\mathop{\mathbb{E}}_{\boldsymbol{r}} \mathop{\mathbb{E}}_{f \mid f \in C_j} [f(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{y}, f(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{y}'] = \mathop{\mathbb{E}}_{\boldsymbol{r}} \mathop{\mathbb{E}}_{f} [f(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{y}, f(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{y}'] \tag{67}$$

$$\mathop{\mathbb{E}}_{\boldsymbol{r}} \Pr(f(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{y}, f(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{y}' \mid f \in C_j) = \mathop{\mathbb{E}}_{\boldsymbol{r}} \Pr(f(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{y}, f(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{y}') \tag{68}$$

$$\Pr(f(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{y}, f(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{y}' \mid f \in C_j) = \Pr(f(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{y}, f(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{y}') \tag{69}$$

Applying Bayes' rule, we have

$$\Pr(f \in C_j \mid f(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{y}, f(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{y}') = \frac{\Pr(f(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{y}, f(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{y}' \mid f \in C_j) \Pr(f \in C_j)}{\Pr(f(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{y}, f(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{y}')} \tag{70}$$

$$= \Pr(f \in C_j) \tag{71}$$

which is precisely the definition of pairwise classical uselessness in the case of oracles that act by XOR. □

Combining this with Theorem 12, we have the following result

Corollary 27. *For any oracle problem C, μ in the standard model with internal randomness, k quantum queries are useless if and only if $2k$ classical queries are pairwise useless*

Since pairwise classical uselessness is equivalent to classical uselessness when f is deterministic, we have the following corollary.

Corollary 28. *If k quantum queries are useless for an oracle problem C, μ in the standard model, then $2k$ classical queries are useless.*

B A classical lower bound for the loops problem

We now prove Theorem 2.

Proof. We can think of each query x to the oracle $\pi = (\pi_1, \pi_2)$ as returning the pair of values $\{\pi_1(x), \pi_2(x)\}$ as this can only reduce the number of queries required. Let oracle instances such that $L_1 = L_2$ be called yes instances and oracle instance for which $L_1 \neq L_2$ be called no instances. A definition is now required.

Definition 29. *Two queries x and y are a yes collision if*

- $\pi_1(x) = \pi_2(y)$ *or* $\pi_1(y) = \pi_2(x)$ *and*
- $x, y \in [N/2]$ *or* $x, y \in (N/2, N]$.

Two queries x and y are a no collision if

- $\pi_1(x) = \pi_2(y)$ *or* $\pi_1(y) = \pi_2(x)$ *and*
- $x \in [N/2]$ *and* $y \in (N/2, N]$ *or* $x \in (N/2, N]$ *and* $y \in [N]$.

A collision is either a yes collision or a no collision.

Consider a sequence $x_1, \ldots, x_T$ of distinct queries with $T \leq \frac{N}{4}$. Note that if a collision has occurred, then we have a yes instance for a yes collision and a no instance for a no collision. Otherwise, it cannot be determined with certainty whether an instance is a yes or no instance. Let T_1 be the number of queries x_i in $[N/2]$ and T_2 be the number of queries in $(N/2, N]$ so $T = T_1 + T_2$. Suppose there have not been any collisions. Consider the possible yes instances that are consistent with the values queried so far. The first step is to choose the rest of the values in $\pi_1[N/2] = \pi_2[N/2]$. Since there have not been any collisions yet, the T queries resulted in $2T$ distinct values (T each for π_1 and π_2). Because the images of π_1 and π_2 on $[N/2]$ are the same, there are $N/2 - 2T_1$ images in $\pi_1[N/2] = \pi_2[N/2]$ to choose from $N - 2T$ possible values. There are $(N/2 - T_1)!$ ways to permute the $N/2 - T_1$ images of π_1 on $[N/2]$ which are not specified by the T queries. Similarly, there are $(N/2 - T_2)!$ ways to permute the $N/2 - T_2$ images of π_1 on $(N/2, N]$ which are not specified by the T queries. Similar statements hold for π_2. Thus, the number of consistent yes instances is

$$K_{yes} = \binom{N - 2T}{N/2 - 2T_1} (N/2 - T_1)!^2 (N/2 - T_2)!^2 \tag{72}$$

We now consider the number of no instances. There are $N/2 - T$ images in $\pi_1[N/2] = \pi_2[N/2]$ to choose from $N - 2T$ possible values. The number of possible permutations is the same as before. Hence, the number of consistent no instances is

$$K_{no} = \binom{N - 2T}{N/2 - T} (N/2 - T_1)!^2 (N/2 - T_2)!^2 \tag{73}$$

Note that $\binom{N-2T}{N/2-T} \geq \binom{N-2T}{N/2-2T_1}$ so $K_{no} \geq K_{yes}$. This implies that when there are not any collisions it is always best to guess that the oracle is a no instance. Assume without loss of generality that $T_2 \geq T_1$. Then the probability of guessing incorrectly is at least the probability of a yes instance which is

$$\Pr(Y \mid x_1, \ldots, x_T) \geq \frac{K_{yes}}{K_{no} + K_{yes}} \tag{74}$$

$$\geq \frac{K_{yes}}{2K_{no}} \tag{75}$$

$$= \frac{(N/2 - T)!^2}{2(N/2 - 2T_1)!(N/2 - 2T_2)!} \tag{76}$$

$$= \frac{(N/2 - T) \cdots (N/2 - 2T_2 + 1)}{2(N/2 - 2T_1) \cdots (N/2 - T + 1)} \tag{77}$$

$$\geq \frac{1}{2} \left(\frac{N/2 - 2T_2 + 1}{N/2 - 2T_1} \right)^{T_2 - T_1} \tag{78}$$

$$= \frac{1}{2} \left(\frac{N/2 - T - D}{N/2 - T + D} \right)^{D} \text{ where } D = T_2 - T_1 \tag{79}$$

$$= \frac{1}{2} \left(1 - \frac{2D}{N/2 - T + D} \right)^{D} \tag{80}$$

$$\geq \frac{1}{2} \left(1 - \frac{2D^2}{N/2 - T + D} \right) \tag{81}$$

$$\geq \frac{1}{2} \left(1 - \frac{2T^2}{N/2 - T} \right) \tag{82}$$

$$\tag{83}$$

Suppose $T = o(\sqrt{N})$. In the limit, $\frac{1}{2} \left(1 - \frac{2T^2}{N/2 - T} \right) \sim \frac{1}{2}$ so asymptotically, the probability of an error given that no collision occurred is at least

$$\Pr(yes \mid x_1, \ldots, x_T) \geq \frac{1}{2} - O\left(\frac{T^2}{N} \right) \tag{84}$$

All that remains is to show that the probability of a collision within the first T queries is small for $T \leq N/4$ and $T = o(\sqrt{N})$. Suppose the first $t - 1$ queries $x_1, \ldots, x_{t-1}$ do not contain any collisions. Then by the union bound, the probability of a collision at the t^{th} query satisfies

$$\Pr(C_t \mid x_1, \ldots, x_{t-1}) \leq \frac{2t}{N - t} \tag{85}$$

Applying the union bound again, the probability of a collision in the first T queries is at most

$$\Pr(C) \leq \frac{2T^2}{N - T} \tag{86}$$

Therefore, asymptotically the probability of an error satisfies

$$\Pr(E) \geq \frac{1}{2}(1 - O(T^2/N)) \tag{87}$$

and is therefore unbounded. We conclude that no randomized algorithm can solve the loops problem with bounded error using less than $\Omega(\sqrt{N})$ queries. □

References

[1] D. A. Meyer and J. Pommersheim. Single Query Learning from Abelian and Non-Abelian Hamming Distance Oracles, 2009. arXiv:0912.0583.

[2] Oded Regev and Liron Schiff. Impossibility of a quantum speed-up with a faulty oracle. In *Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I*, pages 773–781, 2008.

[3] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Limit on the speed of quantum computation in determining parity. *Phys. Rev. Lett.*, 81(24):5442–5444, 1998. arXiv:quant-ph/9802045.

[4] D. A. Meyer and J. Pommersheim. On the Uselessness of Quantum Queries. *Theor. Comp. Sci.*, In Press:–, 2011. arXiv:1004.1434.

[5] A. Montanaro, H. Nishimura, and R. Raymond. Unbounded Error Quantum Query Complexity, 2007. arXiv:0712.1446.

[6] D. Gottesman and I.L. Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402:390–393, 1999. arXiv:quant-ph/9908010.

[7] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum algorithm for the collision problem. *ACM SIGACT News*, 28:14–19, 1997. arXiv:quant-ph/9705002.

[8] Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *J. ACM*, 51(4):595–605, 2004. arXiv:quant-ph/0112086.

[9] L. Sheridan, D. Maslov, and M. Mosca. Approximating fractional time quantum evolution. *J. Phys. A: Math. Theor.*, 42:185302, 2009. arXiv:0810.3843.

[10] Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *STOC '03: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, pages 20–29, 2003. arXiv:quant-ph/0301023.

[11] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. Quantum fingerprinting. *Phys. Rev. Lett.*, 87:167902, 2001. arXiv:quant-ph/0102001.

[12] Daniel R. Simon. On the power of quantum computation. *Siam Journal on Computing*, 1994.

[13] David Rosenbaum. Quantum algorithms for tree isomorphism and state symmetrization, 2010. arXiv:1011.4138.

[14] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48:778–797, July 2001. arXiv:quant-ph/9802049.

[15] J. Neil de Beaudrap, Richard Cleve, and John Watrous. Sharp quantum vs. classical query separations. *ArXiv Quantum Physics e-prints*, 2001.

On exact quantum query complexity

Ashley Montanaro*, Richard Jozsa and Graeme Mitchison

Centre for Quantum Information and Foundations, DAMTP, University of Cambridge, UK.

November 3, 2011

Abstract

We present several families of total boolean functions which have exact quantum query complexity which is a constant multiple (between 1/2 and 2/3) of their classical query complexity, and show that optimal quantum algorithms for these functions cannot be obtained by simply computing parities of pairs of bits. We also characterise the model of nonadaptive exact quantum query complexity in terms of coding theory and completely characterise the query complexity of symmetric boolean functions in this context. These results were originally inspired by numerically solving the semidefinite programs characterising quantum query complexity for small problem sizes. We include numerical results giving the optimal success probabilities achievable by quantum algorithms computing all boolean functions on up to 4 bits, and all symmetric boolean functions on up to 6 bits.

1 Introduction

Many important quantum algorithms operate in the query complexity model. In this model, the quantity of interest is the number of oracle queries to the input $x \in \{0,1\}^n$ required to compute some (possibly partial) function $f(x)$. Many functions f are now known to admit quantum speed-up in the case where the algorithm is allowed a constant probability of error, and the model of bounded-error quantum query complexity is now relatively well understood. Intriguingly, the model of exact quantum query complexity, where the quantum algorithm must succeed with certainty, seems to be more mysterious.

If f is allowed to be a partial function, it is known that there can be an exponential separation between exact quantum and exact classical query complexity [8], and even between exact quantum and bounded-error classical query complexity [4]. There are also examples of total functions where exact quantum algorithms allow a speed-up over classical algorithms. In particular, it is known that the parity of n bits can be computed exactly using only $\lceil n/2 \rceil$ quantum queries [5, 10], while any classical algorithm (exact or randomised) must make exactly n queries. However, this is the best quantum speed-up known for the exact computation of total boolean functions. Even worse, to the authors' knowledge this is essentially the *only* exact quantum speed-up known. Some years ago, van Melkebeek, Hayes and Kutin [12] gave a quantum algorithm that computes the majority function on n bits exactly using only $n + 1 - w(n)$ queries, where $w(n)$ is the number of 1s in the binary expansion of n, but the only quantum ingredient in this algorithm is computing the parity of 2 bits using 1 query. The same applies to quantum algorithms presented by Dubrovska

*am994@cam.ac.uk

and Mischenko-Slatenkova [9], and also algorithms by Vasilieva [19, 20]. In the other direction, it is known that the separation between quantum and classical exact query complexity can be at most cubic [15], whereas the best known relationship between bounded-error quantum and classical query complexity is a 6th power [3].

1.1 Our results

We show that the model of exact quantum query complexity is richer than it may have hitherto appeared. Our results can be divided into analytical and numerical parts. On the analytical side, the main results are as follows.

- We present several new families of boolean functions f for which the exact quantum query complexity of f is a constant multiple (between $1/2$ and $2/3$) of its classical query complexity, and we show that optimal quantum algorithms for these functions cannot be obtained by simply computing parities of pairs of bits. These separations are based on concatenating small separations obtained for functions on small numbers of bits; indeed, we give optimal exact quantum query algorithms for every boolean function on 3 bits.

- We give a simple and explicit quantum algorithm for determining whether a 4-bit string has Hamming weight 2, using only 2 queries. Again, this cannot be done merely by computing parities of pairs of input bits. More generally, we give an exact algorithm which distinguishes between inputs of Hamming weight $n/2$ and Hamming weight in the set $\{0, 1, n-1, n\}$, for all even n, using 2 queries. This generalises the well-known Deutsch-Jozsa problem [8] of distinguishing Hamming weight $n/2$ from Hamming weight in $\{0, n\}$.

- We characterise the model of *nonadaptive* exact quantum query complexity in terms of a coding-theoretic quantity. In this setting, all queries to the input must be made up front, in parallel. In contrast to the classical case, there exist non-trivial quantum algorithms in this model. Using our characterisation result, we completely determine the nonadaptive exact quantum query complexity of symmetric boolean functions.

These analytical results were inspired by numerical investigations in which we evaluated the best success probability achievable by quantum algorithms computing all boolean functions on up to 4 bits, and all symmetric boolean functions on up to 6 bits. This was achieved using the semidefinite programming approach of Barnum, Saks and Szegedy [2] to formulate semidefinite programs (SDPs) giving the precise optimal success probability for quantum algorithms using up to k queries, for all $k < n$. We then solved these SDPs numerically using the CVX package [11] for Matlab; the results are given in Section 6 and Appendix A. Given an SDP solution, one can then write down an *explicit* quantum query algorithm with the same parameters; we discuss how this can be done in Section 4. Our analytical results were based on inspecting these algorithms.

Some further highlights from the numerical results are as follows.

- We conjecture that the exact quantum query complexity of the two problems of determining whether an n-bit string has Hamming weight exactly k is precisely $\max\{k, n-k\}$. We also conjecture that determining whether an n-bit string has Hamming weight at least k has exact quantum query complexity $\max\{k, n-k+1\}$ for $k \geq 1$. Both conjectures hold numerically for all $n \leq 6$.

- In particular, we present strong numerical evidence that the algorithm of van Melkebeek, Hayes and Kutin [12] is not always optimal, by showing that there exists an exact quantum algorithm for computing the majority function on 5 bits using only 3 queries, while their algorithm would require 4 queries.

- We show numerically that all boolean functions on up to 4 bits, with the exception of functions equivalent to the AND function, have an exact quantum query algorithm using at most 3 queries.

Note that Reichardt and Špalek have previously solved related SDPs (known as the adversary and generalised adversary bounds) for all boolean functions on up to 4 bits [18]. These SDPs give lower bounds on bounded-error quantum query complexity but do not characterise it exactly, although the generalised adversary bound does so up to a constant factor [17].

1.2 Organisation

The remainder of this paper is organised as follows. After formalising some definitions in Section 2, in Section 3 we move on to techniques for finding separations between exact quantum query algorithms, classical algorithms, and quantum algorithms computing parities of input bits. We then discuss in Section 4 how the Barnum-Saks-Szegedy SDP can be solved for small problems to give explicit quantum query algorithms. Section 5 gives our algorithm for for determining whether a 4-bit input has Hamming weight 2. In Section 6 we give optimal exact quantum query algorithms, found by semidefinite programming, for every boolean function on 3 bits. Our results characterising nonadaptive exact quantum query complexity are in Section 7, after which we conclude with a discussion of open problems. Two appendices detail our numerical results for all 4-bit boolean functions, and all symmetric functions on up to 6 bits, and also give example CVX source code.

2 Definitions

2.1 Generalities and boolean functions

For any bit-string x, $|x|$ will denote the Hamming weight of x, and $\bar{x}$ will denote $\mathrm{NOT}(x)$. e_i will denote the bit-string with 1 in the i'th position, and 0 elsewhere. We use the convention [X] for a quantity which evaluates to 1 if the statement X is true, and 0 otherwise. We will be interested in boolean functions $f : \{0,1\}^n \to \{0,1\}$, and particularly interested in the following families of functions, all on n bits. PARITY is the function $\mathrm{PARITY}(x) = [|x| \text{ is odd}]$. MAJ is the majority function where $\mathrm{MAJ}(x) = [|x| \geq n/2]$. The EXACT_k function is defined by $\mathrm{EXACT}_k(x) = [|x| = k]$. NAE ("not-all-equal") evaluates to 0 if all the input bits are equal, otherwise evaluates to 1. Finally, the threshold function $\mathrm{Th}_k(x)$ evaluates to 1 if and only if $|x| \geq k$. All of these are examples of *symmetric* boolean functions; a boolean function f is said to be symmetric if $f(x)$ only depends on $|x|$. A non-symmetric function on 3 bits which we will consider is SEL, where $\mathrm{SEL}(x_1, x_2, x_3)$ is defined to be equal to x_2 if $x_1 = 0$, and equal to x_3 if $x_1 = 1$.

We say that two boolean functions f and g are *isomorphic* if they are equal up to negations and permutations of the input variables, and negation of the output variable. This relationship is sometimes known as NPN-equivalence.

2.2 Query complexity model

An exact classical query algorithm to compute a boolean function f is given by a decision tree, or in other words a rooted binary tree whose vertices represent variables x_i to be queried, and whose edges represent branching depending on whether $x_i = 0$ or $x_i = 1$. The number of queries used is the depth of the decision tree; the minimal depth over all decision trees is the exact classical query complexity (aka decision tree complexity) $D(f)$.

We follow what is essentially the standard quantum query complexity model (see e.g. [2, 14]). A quantum query algorithm to compute a boolean function $f : \{0,1\}^n \to \{0,1\}$ is specified by a sequence of unitary operators $U_0, \ldots, U_t$ which do not depend on the (initially unknown) input x. These unitaries are interspersed with oracle calls O_x (which do depend on the input x). The final state of an algorithm that makes t queries, before the final measurement, is given by $U_t O_x U_{t-1} O_x \ldots O_x U_0 |0\rangle$. The overall Hilbert space $\mathcal{H}$ used by the quantum query algorithm is split into three subspaces $\mathcal{H}_{\text{in}} \otimes \mathcal{H}_{\text{work}} \otimes \mathcal{H}_{\text{out}}$. For boolean functions, $\mathcal{H}_{\text{out}}$ is a single qubit, whereas the workspace $\mathcal{H}_{\text{work}}$ is of arbitrary size. The oracle O_x acts only on $\mathcal{H}_{\text{in}}$ by mapping $|i\rangle \mapsto (-1)^{x_i} |i\rangle$ for some hidden bit string x. $\mathcal{H}_{\text{in}}$ is $n+1$ dimensional and indexed by basis vectors $|0\rangle, \ldots, |n\rangle$; a query to x_0 always returns 0. The final step of the algorithm is always simply to measure the $\mathcal{H}_{\text{out}}$ register and return the outcome. We say that the algorithm computes f within error ϵ if, on input x, the algorithm returns $f(x)$ with probability at least $1 - \epsilon$ for all x. The exact quantum query complexity $Q_E(f)$ is the minimum number of queries used by any quantum algorithm which computes $f(x)$ exactly for all x.

Note that if boolean functions f and g are isomorphic, $D(f) = D(g)$ and $Q_E(f) = Q_E(g)$.

3 Separating exact quantum and classical query complexity

We observe that any fixed function demonstrating a separation between exact quantum and classical query complexity, even a small additive constant, gives rise to a constant factor multiplicative separation for an asymptotically growing family of functions as follows.

Proposition 1. *Let $f : \{0,1\}^k \to \{0,1\}$ be a boolean function on k bits such that $Q_E(f) = m_q$ and $D(f) = m_c$ for some $m_q \leq m_c$. Also let $g : \{0,1\}^n \to \{0,1\}$ be a boolean function such that $D(g) = n$. Define the family of functions $f_n : \{0,1\}^{nk} \to \{0,1\}$ as follows: divide the nk input bits into blocks $b_1, \ldots, b_n$ of k bits each, and set $f_n(x_1, \ldots, x_{nk}) = g(f(b_1), f(b_2), \ldots, f(b_n))$. Then $Q_E(f_n) \leq m_q n$ and $D(f_n) = m_c n$.*

Proof. An exact quantum query algorithm for f_n using $m_q n$ queries can be obtained simply by computing $f(b_i)$ for each block b_i individually, and then classically computing $g(f(b_1), \ldots, f(b_n))$, so $Q_E(f_n) \leq m_q n$. On the other hand, any classical algorithm for f_n must know the value of $f(b_i)$ for all i with certainty. Otherwise, this would imply an algorithm that computes g using fewer than n queries, contradicting $D(g) = n$. Hence $D(f_n) = m_c n$. □

Note that, in contrast to the classical case, it is not necessarily true that the reverse inequality $Q_E(f_n) \geq m_q n$ holds; a counterexample is provided by the PARITY function $x_1 \oplus x_2$ on 2 bits. Natural examples of functions g to which Proposition 1 can be applied are AND and OR. Thus an example of an exact query complexity separation we obtain from our results on small boolean functions is the following (see Section 5).

Theorem 2. *Let $EXACT_2^\ell$ be the boolean function on 4ℓ bits defined as follows. Split the input x into consecutive blocks $b_1, \ldots, b_\ell$ containing 4 bits each, and set $EXACT_2^\ell(x_1, \ldots, x_{4\ell}) = 1$ if each block b_i contains exactly two 1s. Then $Q_E(EXACT_2^\ell) = 2\ell$ and $D(EXACT_2^\ell) = 4\ell$.*

Note that the lower bound $Q_E(\mathrm{EXACT}_2^\ell) \geq 2\ell$ in this theorem does not follow immediately from Proposition 1, but can be shown using polynomial degree arguments [3].

3.1 Quantum algorithms based on parity queries

It is well-known that quantum computers can evaluate the parity of two input bits exactly using only one query [5]. Thus a non-trivial class of exact quantum query algorithms consists of classical decision trees, each of whose nodes is a query either to an individual bit of the input, or to the parity of two input bits. One can put a lower bound on the number of queries used by such algorithms (indeed, a more general class of algorithms) as follows.

Proposition 3. *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function, and let d be the degree of f as an n-variate polynomial over $\mathbb{F}_2$. Then any decision tree which can query the parity of any subset of the input variables at unit cost must make at least d queries to the input to compute f with certainty.*

Proof. We will show by induction that any decision tree on parities which has depth D gives rise to a degree D polynomial over $\mathbb{F}_2$. The function computed by any such tree can be written as $pT_0 + (1+p)T_1$ for some degree 1 polynomial p over $\mathbb{F}_2$ and decision trees T_0, T_1 of depth at most $D-1$. Therefore, the degree of the polynomial obtained by repeating this procedure recursively is at most D. If the tree computes f on every input, this polynomial must be equal to f, and hence be degree d. Thus $D \geq d$. □

4 Quantum query algorithms from semidefinite programming

In this section we discuss, following [2], how quantum query complexity can be evaluated as a semidefinite programming (SDP) problem, given in Definition 1 below. In this definition, $\circ$ is the Hadamard (entrywise) product of matrices. The following important characterisation will be the key to many of our results.

Theorem 4 (Barnum, Saks and Szegedy [2])**.** *There is a quantum query algorithm that uses t queries to compute a function $f : \{0,1\}^n \to \{0,1\}$ within error ϵ if and only if the SDP of Definition 1 is feasible.*

Therefore, if one minimises ϵ subject to these semidefinite constraints, one obtains the lowest possible error that can be achieved by a quantum algorithm which computes f using t queries.

4.1 A prescription for quantum algorithms

It is perhaps not immediately obvious how, given a solution to the semidefinite program of Definition 1, to produce a quantum query algorithm with the same parameters. This was implicit in [2]; we now spell out explicitly how it can be done. We will use the following standard lemma from linear algebra (see e.g. [13]).

Definition 1 (Quantum query complexity primal SDP)**.**

For a given boolean function $f : \{0,1\}^n \to \{0,1\}$ and a given integer t, find a sequence of 2^n-dimensional real symmetric matrices $(M_i^{(j)})$, where $0 \leq i \leq n$ and $0 \leq j \leq t-1$, and 2^n-dimensional real symmetric matrices Γ_0, Γ_1, satisfying the following constraints.

$$\sum_{i=0}^{n} M_i^{(0)} = E_0 \tag{1}$$

$$\sum_{i=0}^{n} M_i^{(j)} = \sum_{i=0}^{n} E_i \circ M_i^{(j-1)} \text{ (for } 1 \leq j \leq t-1) \tag{2}$$

$$\Gamma_0 + \Gamma_1 = \sum_{i=0}^{n} E_i \circ M_i^{(t-1)} \tag{3}$$

$$F_0 \circ \Gamma_0 \geq (1-\epsilon)F_0 \tag{4}$$

$$F_1 \circ \Gamma_1 \geq (1-\epsilon)F_1 \tag{5}$$

where E_i is the matrix $\langle x|E_i|y\rangle = (-1)^{x_i+y_i}$, and F_0 and F_1 are diagonal matrices defined by $\langle x|F_z|x\rangle = 1$ if and only if $f(x) = z$, and $\langle x|F_z|x\rangle = 0$ otherwise.

Lemma 5. *Let $S = (|\psi_i\rangle)$ and $T = (|\phi_j\rangle)$ be two sequences of m vectors of the same dimension. Define $\Psi = \sum_i |\psi_i\rangle\langle i|$, $\Phi = \sum_j |\phi_j\rangle\langle j|$. Then there is a unitary U such that $U|\phi_i\rangle = |\psi_i\rangle$ for all i if and only if $\Psi^\dagger\Psi = \Phi^\dagger\Phi$. If such a U exists, it can be written down as follows. Let V and W be any isometries satisfying $\Psi = V\sqrt{\Psi^\dagger\Psi}$, $\Phi = W\sqrt{\Phi^\dagger\Phi}$ (i.e. isometries occurring in polar decompositions of Ψ, Φ), and complete V and W to unitary matrices V' and W'. Then $U = V'(W')^\dagger$.*

Given a set of matrices $M_i^{(j)}$ as in Definition 1, one can derive an explicit quantum query algorithm completely mechanically, as follows. Use a workspace $\mathcal{H}_{\text{work}}$ of n qubits, and ignore the output qubit for the time being. Let the initial state be $|0\rangle|0\rangle$, and let the state of the system at time j (i.e. after j queries have been made, and just before the $(j+1)$'st query is made) be $|\psi_x^{(j)}\rangle = \sum_{i=0}^{n} |i\rangle|\psi_{x,i}^{(j)}\rangle$, where $|\psi_{x,i}^{(j)}\rangle$ is a subnormalised state in the Hilbert space $\mathcal{H}_{\text{work}}$.

We will *define* the state at time $0 \leq j \leq t-1$ in terms of the matrices $M_i^{(j)}$ occurring in a solution to the SDP by setting

$$|\psi_{x,i}^{(j)}\rangle = \sqrt{M_i^{(j)}}|x\rangle, \tag{6}$$

where $\sqrt{M_i^{(j)}}$ is the unique positive semidefinite square root of $M_i^{(j)}$. It is perhaps not immediately clear that following this prescription leads to $|\psi_x^{(j)}\rangle$ being normalised, let alone the sequence $|\psi_x^{(0)}\rangle, |\psi_x^{(1)}\rangle, \ldots, |\psi_x^{(t-1)}\rangle$ corresponding to a valid quantum query algorithm for all x; however, we will now see that this is indeed the case.

For any $1 \leq j \leq t$, define

$$|\phi_x^{(j)}\rangle = \sum_{i=0}^{n} (-1)^{x_i}|i\rangle|\psi_{x,i}^{(j-1)}\rangle,$$

and set $|\phi_x^{(0)}\rangle = |0\rangle|0\rangle$. Also define the matrices obtained by concatenating the vectors as columns,

$$\Psi^{(j)} = \sum_{x\in\{0,1\}^n} |\psi_x^{(j)}\rangle\langle x|, \ \Phi^{(j)} = \sum_{x\in\{0,1\}^n} |\phi_x^{(j)}\rangle\langle x|.$$

The vectors $|\phi_x^{(j)}\rangle$ represent the state of the system immediately *after* the j'th oracle call, and $|\phi_x^{(0)}\rangle$ is the initial state of the system. We would like to find unitaries $U_0, \ldots, U_{t-1}$ mapping $|\phi_x^{(j)}\rangle \mapsto |\psi_x^{(j)}\rangle$ for all x. By Lemma 5, such a sequence of unitaries will exist if

$$(\Psi^{(j)})^\dagger \Psi^{(j)} = (\Phi^{(j)})^\dagger \Phi^{(j)}.$$

But observe that for any $0 \leq j \leq t-1$,

$$\langle x|(\Psi^{(j)})^\dagger \Psi^{(j)}|y\rangle = \langle \psi_x^{(j)}|\psi_y^{(j)}\rangle = \sum_{i=0}^{n} \langle \psi_{x,i}^{(j)}|\psi_{y,i}^{(j)}\rangle = \sum_{i=0}^{n} \langle x|M_i^{(j)}|y\rangle,$$

so $(\Psi^{(j)})^\dagger \Psi^{(j)} = \sum_{i=0}^n M_i^{(j)}$. Similarly, for any $1 \leq j \leq t$,

$$\begin{aligned}\langle x|(\Phi^{(j)})^\dagger \Phi^{(j)}|y\rangle &= \langle \phi_x^{(j)}|\phi_y^{(j)}\rangle = \sum_{i=0}^{n} (-1)^{x_i+y_i} \langle \psi_{x,i}^{(j-1)}|\psi_{y,i}^{(j-1)}\rangle = \sum_{i=0}^{n} (-1)^{x_i+y_i} \langle x|M_i^{(j-1)}|y\rangle \\ &= \langle x| \left(\sum_{i=0}^{n} E_i \circ M_i^{(j-1)} \right) |y\rangle.\end{aligned}$$

As, by constraint (2) in the SDP, $\sum_{i=0}^n M_i^{(j)} = \sum_{i=0}^n E_i \circ M_i^{(j-1)}$ for all $1 \leq j \leq t-1$, this implies by Lemma 5 that for each $j \geq 1$ there exists a unitary U_j such that $U_j|\phi_x^{(j)}\rangle = |\psi_x^{(j)}\rangle$ for all x, and this U_j can be determined explicitly from Lemma 5. In the case $j = 0$, $(\Phi^{(j)})^\dagger \Phi^{(j)} = E_0$, and hence SDP constraint (1) implies the existence of a U_0 such that $U_0|\phi_x^{(0)}\rangle = |\psi_x^{(0)}\rangle$ for all x.

The final constraint we need to satisfy is that the algorithm outputs the correct result. Define the final state of the system on input x (just before the output qubit is measured) to be

$$|\gamma_x\rangle = |0\rangle_{\text{in}}(\sqrt{\Gamma_0}|x\rangle)_{\text{work}}|0\rangle_{\text{out}} + |0\rangle_{\text{in}}(\sqrt{\Gamma_1}|x\rangle)_{\text{work}}|1\rangle_{\text{out}}.$$

Then

$$\langle \gamma_x|\gamma_y\rangle = \langle x|\Gamma_0|y\rangle + \langle x|\Gamma_1|y\rangle = \langle x| \left(\sum_{i=0}^{n} E_i \circ M_i^{(t-1)} \right) |y\rangle$$

by SDP constraint (3), so by a similar argument there exists a U_t such that $U_t|\phi_x^{(t)}\rangle = |\gamma_x\rangle$ for all x. Measuring the output qubit gives the answer 0 with probability $\langle x|\Gamma_0|x\rangle$, which by constraint (4) is at least $1-\epsilon$ when $f(x) = 0$. Similarly, by constraint (5) we obtain the answer 1 with probability at least $1-\epsilon$ when $f(x) = 1$.

Observe that we have some freedom in our choice of states $|\psi_{x,i}^{(j)}\rangle$; while eqn. (6) gives one choice which always works, it would suffice to pick any states such that $\langle \psi_{x,i}^{(j)}|\psi_{y,i}^{(j)}\rangle = \langle x|M_i^{(j)}|y\rangle$. In particular, if the rank of $M_i^{(j)}$ is upper bounded by r for all i, j, one can choose states of dimension r throughout. This would reduce the size of the $\mathcal{H}_{\text{work}}$ register from n qubits to $\lceil \log_2 r \rceil$ qubits. Also observe that without loss of generality all states and unitaries occurring in a quantum query algorithm can be taken to be real.

5 EXACT$_2$

We now give a simple and explicit quantum algorithm for evaluating the EXACT$_2$ function on 4 bits using only 2 quantum queries. This algorithm was originally inspired by numerically solving the SDP discussed in the previous section. The algorithm does not use any workspace (or even an output register), and hence operates solely on the 5-dimensional input register indexed by basis states $\{|0\rangle, \ldots, |4\rangle\}$. Define a unitary matrix U by

$$U = \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & \omega & \omega^2 \\ 1 & 1 & 0 & \omega^2 & \omega \\ 1 & \omega & \omega^2 & 0 & 1 \\ 1 & \omega^2 & \omega & 1 & 0 \end{pmatrix},$$

where $\omega = e^{2\pi i/3}$ is a complex cube root of 1. We begin in the state

$$|\psi\rangle = \frac{1}{2}\sum_{i=1}^{4} |i\rangle$$

and then apply O_x, then U, then O_x again. Finally, we perform the measurement consisting of a projection onto the state $|\psi\rangle$ and its orthogonal complement. If the outcome is $|\psi\rangle$, we output 1, and otherwise 0.

The claim is that $V_x := O_x U O_x$ leaves $|\psi\rangle$ unchanged, up to a phase factor, when x has Hamming weight 2, and otherwise maps $|\psi\rangle$ into a subspace orthogonal to $|\psi\rangle$. To see that the claim is correct, note first that $U|\psi\rangle = |0\rangle$, since $1 + \omega + \omega^2 = 0$. But for $x = 0000$, O_x is the identity. Thus

$$V_{0000}|\psi\rangle = |0\rangle.$$

For $x = 1000$, $O_x|\psi\rangle = |\psi\rangle - |1\rangle$. So $V_x|\psi\rangle = |0\rangle - O_x U|1\rangle$. But the coefficient of $|1\rangle$ in $U|1\rangle$ is zero, so O_x leaves $U|1\rangle$ unchanged and we have

$$V_{1000}|\psi\rangle = |0\rangle - U|1\rangle.$$

Similar results hold for the other weight 1 strings x.

For $x = 1100$, $O_x|\psi\rangle = |\psi\rangle - |1\rangle - |2\rangle$, and

$$\begin{aligned} U(|\psi\rangle - |1\rangle - |2\rangle) &= |0\rangle - \frac{1}{2}\left(2|0\rangle + |1\rangle + |2\rangle + (\omega + \omega^2)(|3\rangle + |4\rangle)\right) \\ &= \frac{1}{2}(-|1\rangle - |2\rangle + |3\rangle + |4\rangle). \end{aligned}$$

Applying O_x once more we get

$$V_{1100}|\psi\rangle = |\psi\rangle.$$

We get the same result for other strings of weight 2, possibly with a phase factor. For instance

$$V_{1001}|\psi\rangle = \omega^2|\psi\rangle.$$

Given a string x of weight 3, we can flip all the bits and the oracle acts identically but with a change of sign. Thus such a string behaves like one of weight 1, and similarly a string of weight 4 behaves like one of weight 0. Thus, for x such that $|x| \neq 2$, $V_x|\psi\rangle$ lies in the span of $|0\rangle$ and $U|i\rangle$ for $i = 1, 2, 3, 4$. However $\langle\psi|0\rangle = 0$ and $\langle\psi|U|i\rangle = \frac{1}{4}(1 + \omega + \omega^2) = 0$ for $i = 1, 2, 3, 4$. So this subspace is orthogonal to $|\psi\rangle$, proving our claim.

5.1 Distinguishing weights 0 and 1 from balanced strings

We would ideally like to understand the number of queries required to solve the EXACT_k function on n input bits, for all n and k. As an intermediate goal generalising our solution for $n = 4$, one can ask for a two-query algorithm, for any even n, that distinguishes strings of weight 0 and 1 from strings of weight $n/2$. We take the previous space, with basis vectors $|i\rangle$, $i = 0, 1, \ldots, n$, and tensor it with an ancilla space. In the ancilla space we select vectors $|a_i\rangle$, $i = 1, \ldots, n$, of unit length with inner products $\langle a_i|a_j\rangle = c$, for $i \neq j$, where c is arbitrary such that $|c| \leq 1$. Such vectors can always be found. We also select some vector $|0\rangle$ in the ancilla space.

The oracle acts on the first space, so $O_x|i\rangle|j\rangle = (-1)^{x_i}|i\rangle|j\rangle$ for any i, j. Recall that e_i is the string with a 1 at position i and 0's elsewhere, and let b be the "balanced" string with 1's in the first n/2 places. The algorithm starts with the state $|\phi\rangle = \sum_{i=1}^{n}|i\rangle|0\rangle$ (we keep $|\phi\rangle$ unnormalised for simplicity). As before, $V_x := O_x U O_x$, with U to be defined shortly. The aim is to show that $V_b|\phi\rangle$ is orthogonal to the subspace spanned by $V_{e_i}|\phi\rangle$ for all i, which thus discriminates between balanced and weight 1 strings (discriminating weight 0 strings will follow automatically from this).

Now define U by its action on the states $|\tau_i\rangle := O_{e_i}|\phi\rangle = |\phi\rangle - 2|i\rangle|0\rangle$:

$$U|\tau_i\rangle = \alpha|00\rangle + \beta\left(\sum_{j=1}^{n}|j\rangle|a_{j+i-1}\rangle - |i\rangle|a_{2i-1}\rangle\right) \tag{7}$$

where the subscript of the a's is taken mod n. This will be an isometry (which can be extended to a unitary on the whole tensor product space) if

$$\alpha^2 + (n-1)\beta^2 = n, \tag{8}$$

$$\alpha^2 + (n-2)\beta^2 c = n - 4. \tag{9}$$

Note that $U|\tau_i\rangle = V_{e_i}|\phi\rangle$, since $U|\tau_i\rangle$ is invariant under O_{e_i}.

U can be defined on $O_b|\phi\rangle$ by using

$$O_b|\phi\rangle = \frac{1}{2}\left(\sum_{i=1}^{n/2}|\tau_i\rangle - \sum_{i=n/2+1}^{n}|\tau_i\rangle\right),$$

giving, up to an overall factor of β,

$$V_b|\phi\rangle = \frac{1}{2}\left(\sum_{i=1}^{n/2}O_b U|\tau_i\rangle - \sum_{i=n/2+1}^{n}O_b U|\tau_i\rangle\right) = \sum_{j=1}^{n}\sum_{k=1}^{n}\pi_{jk}|j\rangle|a_{k+j-1}\rangle + \sum_{j=1}^{n}|j\rangle|a_{2j-1}\rangle, \tag{10}$$

where

$$\pi_{jk} = -1 \text{ if } i \in L, j \in L, \qquad \pi_{jk} = 1 \text{ if } i \in L, j \in H,$$
$$\pi_{jk} = 1 \text{ if } i \in H, j \in L, \qquad \pi_{jk} = -1 \text{ if } i \in H, j \in H,$$

and $L = [1, n/2], H = [n/2 + 1, n]$. Combining (10) with (7) gives

$$\begin{aligned}\langle\phi|V_b V_{e_i}|\phi\rangle &= \sum_j\sum_k \pi_{jk}\langle a_{j+k-1}|a_{j+i-1}\rangle - \sum_j\langle a_{i+j-1}|a_{2i-1}\rangle + \sum_j\langle a_{2j-1}|a_{i+j-1}\rangle - \langle a_{2i-1}|a_{2i-1}\rangle \\ &= 0 - (c-1) + (1 + (n-1)c) - 1 \\ &= 1 + (n-2)c.\end{aligned}$$

Thus for orthogonality we require $n = -1/(n-2)$, which, with (8) and (9), gives $\alpha^2 = (n-2)^2/n$ and $\beta^2 = 4/n$.

Finally, we would like to show orthogonality for the weight zero sequence, i.e. orthogonality of $V_b|\phi\rangle$ and $U|\phi\rangle$. However, this follows from what we proved above, using

$$|\phi\rangle = \frac{1}{n-2}\sum_{i=1}^{n}|\tau_i\rangle,$$

together with the fact that $U|\tau_i\rangle$ is invariant under O_{e_i}.

6 Exact quantum query algorithms for small functions

Having whetted our appetite with the EXACT_2 problem, we now turn to quantum algorithms for other small boolean functions. For each function on n bits we considered, we calculated, via the SDP of Definition 1, the best possible success probability achievable by quantum algorithms making t queries, for $t = 1, \ldots, n-1$ (any function can clearly be computed exactly using n queries). We did this for all functions on up to 4 input bits, and for all symmetric functions on up to 6 bits. We used the convex optimisation package CVX [11] for Matlab, which allows optimisation problems to be specified in a simple and intuitive way; see Appendix B for source code. The CVX package allows the choice of underlying solvers SeDuMi and SDPT3. We used SeDuMi for the results given below, and also checked the results with SDPT3, which gave the same values up to a difference of at most 0.001. The numerical results for functions on 4 bits, and symmetric functions on up to 6 bits, are deferred to Appendix A.

Note that there is a basic issue with calculating *exact* quantum query complexity numerically, which is that one receives a numerical solution from the SDP solver, which is not exact. If the SDP solver claims that there exists a quantum query algorithm that computes some function f using k queries with success probability at least 0.999, for example, one cannot be sure that this algorithm is actually exact. In the case of all functions on up to 3 bits, we therefore give explicit optimal *exact* quantum query algorithms. These algorithms were obtained by a somewhat laborious process of taking the numerically obtained (real-valued, approximate) solutions to the SDP and using these as a guide to find exact solutions.

For completeness, we begin by giving optimal exact quantum query algorithms for all boolean functions of 1 and 2 bits. In what follows, the tables are indexed by function ID; the ID of each function is the integer obtained by converting its truth table from binary. Columns give the optimal success probability that can be achieved by quantum algorithms making $1, \ldots, n-1$ queries. Entries are starred when there is a *nonadaptive* exact quantum algorithm using that number of queries (see Section 7).

6.1 Functions of up to 2 bits

Up to isomorphism, the only non-constant function of 1 bit is $f(x_1) = x_1$, which clearly requires exactly one query. In the case of 2 bits, there are two classes of functions.

ID	Function	1 query
1	$x_1 \wedge x_2$	0.900
6	$x_1 \oplus x_2$	1*

An optimal quantum algorithm for the function $x_1 \oplus x_2$ proceeds as follows [5]. Input the state $\frac{1}{\sqrt{2}}(|1\rangle + |2\rangle)$ into the oracle to produce $\frac{1}{\sqrt{2}}((-1)^{x_1}|1\rangle + (-1)^{x_2}|x_2\rangle)$. Perform a Hadamard gate (with respect to the basis $\{|1\rangle, |2\rangle\}$), measure in the basis $\{|1\rangle, |2\rangle\}$, and output 0 if "1" is measured, and 1 if "2" is measured. It is easy to see that this algorithm succeeds with certainty.

6.2 Functions of 3 bits

The following table lists all boolean functions depending on 3 bits, up to isomorphism.

ID	Function	1 query	2 queries	$\mathbb{F}_2$ degree	D(f)
1	$x_1 \wedge x_2 \wedge x_3$	0.800	0.980	3	3
6	$x_1 \wedge (x_2 \oplus x_3)$	0.667	1*	2	3
7	$x_1 \wedge (x_2 \vee x_3)$	0.773	1	3	3
22	EXACT_2	0.571	1	3	3
23	MAJ	0.667	1	2	3
30	$x_1 \oplus (x_2 \vee x_3)$	0.667	1	2	3
53	$\mathrm{SEL}(x_1, x_2, x_3)$	0.854	1	2	2
67	$(x_1 \wedge x_2) \vee (\bar{x_1} \wedge \bar{x_2} \wedge x_3)$	0.773	1	3	3
105	PARITY	0.500	1*	1	3
126	NAE	0.900	1*	2	3

Observe that the AND function requires 3 queries to be computed exactly; in fact, it has been known for some time that AND on n bits has $Q_E(\mathrm{AND}) = n$ [3]. For most of the other functions, an optimal exact quantum algorithm is easy to determine, based only on classical queries and computing the parity of two bits using one query.

- $x_1 \wedge (x_2 \oplus x_3)$: Query x_1 and evaluate $x_2 \oplus x_3$ using one query.
- MAJ: First evaluate $x_1 \oplus x_2$. If the answer is 1, then output x_3, otherwise output x_1. This works because if x_1 and x_2 are different, then there will be at least two 1's in total if and only if x_3 is 1. If x_1 and x_2 are the same, there will be at least two 1's if and only if x_1 is 1.
- $x_1 \oplus (x_2 \vee x_3)$: This function is equivalent to $(\bar{x_3} \wedge x_2) \vee (x_3 \wedge (x_1 \oplus x_2))$. So query x_3 first, then either query x_2 or $x_1 \oplus x_2$.
- $\mathrm{SEL}(x_1, x_2, x_3)$: Query x_1 first, then either output x_2 or x_3.
- PARITY: Evaluate $x_1 \oplus x_2$, query x_3, and take the exclusive OR of the two.
- NAE: This function is equivalent to $(x_1 \oplus x_2) \vee (x_1 \oplus x_3)$.

However, the three remaining functions (EXACT_2, $x_1 \wedge (x_2 \vee x_3)$ and $(x_1 \wedge x_2) \vee (\bar{x_1} \wedge \bar{x_2} \wedge x_3)$) do not have such straightforward optimal algorithms. Indeed, by Proposition 3, they cannot be computed using 2 queries by any algorithm which is a decision tree on parity queries.

In the case of EXACT_2, we obtain an optimal algorithm by appending an additional zero bit and computing EXACT_2 on 4 bits (see Section 5). We now give quantum query algorithms for the two remaining functions. Rather than writing out the unitary operators arising in the algorithm explicitly, we simply give expressions for matrices forming a exact solution to the query complexity SDP. We stress that, given these matrices, one can follow the procedure of Section 4 to find an

explicit quantum algorithm completely mechanically. The matrices are fully specified by their non-zero eigenvalues and eigenvectors; note that, for readability, we have neither normalised nor orthogonalised the eigenvectors.

6.2.1 $x_1 \wedge (x_2 \vee x_3)$

Matrix	Eigenvalues	Eigenvectors
$M_0^{(0)}, M_1^{(0)}, M_2^{(0)}, M_3^{(0)}$	2	$(1,1,1,1,1,1,1,1)$
$M_0^{(1)}$	3/2	$(1,1,1,1,0,0,0,0)$
	1	$\{(-1,1,-1,1,0,2,0,2),(0,-1,1,0,0,-1,1,0)\}$
$M_1^{(1)}$	1	$(1,0,0,-1,2,1,1,0)$
	1/2	$(-1,0,-1,0,0,1,0,1)$
$M_2^{(1)}$	1	$(1,0,0,-1,2,1,1,0)$
	1/2	$(-1,-1,0,0,0,0,1,1)$
$M_3^{(1)}$	3/4	$\{(0,1,0,1,0,1,0,1),(0,-1,1,0,0,-1,1,0)\}$
Γ_0	5/2	$(3,2,2,3,2,0,0,0)$
	1	$\{(0,-1,0,0,1,0,0,0),(0,-1,1,0,0,0,0,0)\}$
	1/2	$(-1,0,0,1,0,0,0,0)$
Γ_1	3/2	$\{(0,0,0,0,0,1,0,1),(0,0,0,0,0,-1,1,0)\}$

6.2.2 $(x_1 \wedge x_2) \vee (\bar{x_1} \wedge \bar{x_2} \wedge x_3)$

Matrix	Eigenvalues	Eigenvectors
$M_0^{(0)}, M_1^{(0)}, M_2^{(0)}, M_3^{(0)}$	2	$(1,1,1,1,1,1,1,1)$
$M_0^{(1)}$	1	$(-2,-1,-1,0,-1,0,0,1)$
	3/4	$(0,0,-1,-1,2,2,1,1)$
	1/4	$(0,0,1,1,0,0,1,1)$
$M_1^{(1)}$	1	$(-2,-1,-1,0,-1,0,0,1)$
	3/4	$(0,-1,2,1,-1,-2,1,0)$
	1/4	$(0,-1,0,-1,1,0,1,0)$
$M_2^{(1)}$	1	$(0,1,1,2,1,2,2,3)$
	3/4	$(0,-1,0,-1,1,0,1,0)$
	1/4	$(0,-1,2,1,-1,-2,1,0)$
$M_3^{(1)}$	1	$(0,3,-1,2,-1,2,-2,1)$
	3/4	$(0,0,1,1,0,0,1,1)$
	1/4	$(0,0,-1,-1,2,2,1,1)$
Γ_0	$\frac{1}{4}(5+\sqrt{5})$	$(1+\sqrt{5},0,-1+\frac{1}{2}(5+\sqrt{5}),1,-1+\frac{1}{2}(5+\sqrt{5}),1,0,0)$
	3/2	$(0,0,0,-1,0,1,0,0)$
	1	$(0,0,-1,0,1,0,0,0)$
	$\frac{1}{4}(5-\sqrt{5})$	$(1-\sqrt{5},0,-1+\frac{1}{2}(5-\sqrt{5}),1,-1+\frac{1}{2}(5-\sqrt{5}),1,0,0)$
Γ_1	3/2	$\{(0,-1,0,0,0,0,0,1),(0,1,0,0,0,0,1,0)\}$

7 Nonadaptive exact quantum query complexity

We now turn to a very restricted model of exact query complexity, in which the algorithm's queries are required to be nonadaptive. In other words, the choice of which input variables to query cannot depend on the result of previous queries, so the algorithm must choose which variables to query at the start. For a boolean function f, let $D^{na}(f)$, $Q_E^{na}(f)$ be the nonadaptive quantum and classical exact query complexities of f, i.e. the minimum number of nonadaptive queries required to compute f with certainty. This model is extremely restricted classically, as we see from the following easy proposition.

Proposition 6. *For any boolean function f depending on n variables, $D^{na}(f) = n$.*

Proof. A nonadaptive exact classical query algorithm $\mathcal{A}$ making k queries is specified by a list of k fixed variables which are queried. If $k < n$, there must exist a variable i which is not queried, but on which f depends. Thus there must exist an input x such that if bit i is flipped, $\mathcal{A}$ does not notice the difference, so $\mathcal{A}$ cannot be correct on every input. □

In the case of nonadaptive *quantum* query complexity, things are more interesting. An optimal quantum algorithm for PARITY is nonadaptive and uses $\lceil n/2 \rceil$ quantum queries [5, 10], so quantum algorithms can indeed achieve an advantage over classical algorithms in this model. However, it is also known that this separation by a factor of 2 is optimal for total functions in the nonadaptive model [16].

We now introduce some additional notation. For any boolean function $f : \{0,1\}^n \to \{0,1\}$, define

$$S_f := \{z : \forall x, f(x) = f(x+z)\},$$

where addition is over the group $\mathbb{Z}_2^n$; i.e. S_f is the set of translations of the input under which f is invariant. Note that S_f is a subspace of $\{0,1\}^n$. For any subspace $S \subseteq \{0,1\}^n$, let $S^\perp$ denote the orthogonal subspace to S, i.e. $S^\perp = \{x : x \cdot s = 0, \forall\, s \in S\}$. Finally, let $d(x,S)$ denote the *maximum* Hamming distance between a bit-string $x \in \{0,1\}^n$ and a subset $S \subseteq \{0,1\}^n$: $d(x,S) = \max_{y \in S} d(x,y)$. Then we have the following theorem.

Theorem 7. *For any boolean function $f : \{0,1\}^n \to \{0,1\}$,*

$$Q_E^{na}(f) = \min_{x \in \{0,1\}^n} \max_{y \in S_f^\perp} d(x,y) = \min_{x \in \{0,1\}^n} d(x, S_f^\perp).$$

We have thus completely characterised the nonadaptive quantum query complexity of f. In the coding theory literature, the quantity $\min_{x \in \{0,1\}^n} d(x, S_f^\perp)$ is known as the *radius* of the code $S_f^\perp$ [6]. Observe that Theorem 7 implies that $Q_E^{na}(f)$ can be computed exactly in time polynomial in 2^n. We now prove this theorem and then draw some corollaries. In the proof it will be convenient to use the notation

$$\hat{f}(x) = \frac{1}{2^n} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} f(y)$$

for the Fourier transform (over $\mathbb{Z}_2^n$) of some function $f : \{0,1\}^n \to \mathbb{R}$.

Proof of Theorem 7. Any nonadaptive quantum algorithm making k queries to the input x corresponds to a choice of a state $|\psi\rangle$, which is input to k copies of the oracle (i.e. the unitary operator $O_x^{\otimes k}$), followed by a projective measurement to determine whether $f(x) = 0$ or $f(x) = 1$. Label

computational basis states by a length k string of integers $(i_1, \ldots, i_k)$ in the range $\{0, \ldots, n\}$; each such string represents a list of variables queried, with 0 representing a "null query" which does nothing. $O_x^{\otimes k}$ acts on these basis states by mapping

$$|i_1, \ldots, i_k\rangle \mapsto (-1)^{x_{i_1}+\cdots+x_{i_k}}|i_1, \ldots, i_k\rangle.$$

Now note that we can restrict ourselves to query strings in non-decreasing order, and containing at most one of each integer between 1 and n. The first of these is because querying any permutation of a string is equivalent to querying the string itself. The second is because querying the same index twice does nothing, and hence is equivalent to the null query.

These strings are now in obvious one-to-one correspondence with the set of n-bit strings of Hamming weight at most k. Thus the state we obtain from applying $O_x^{\otimes k}$ to an arbitrary input state is of the form

$$|\psi_x\rangle = \sum_{s\in\{0,1\}^n, |s|\leq k} (-1)^{s\cdot x}\alpha_s|s\rangle.$$

A nonadaptive quantum query algorithm computing f exactly using k queries exists if and only if there exists a set $\{\alpha_s\}$ such that $\langle\psi_x|\psi_y\rangle = 0$ for all x, y such that $f(x) \neq f(y)$, or in other words

$$\sum_{s\in\{0,1\}^n, |s|\leq k} (-1)^{s\cdot(x+y)}|\alpha_s|^2 = 0.$$

For brevity, write $w(s) = |\alpha_s|^2$. The above constraint says that, for all $z \notin S_f$,

$$\sum_{s\in\{0,1\}^n, |s|\leq k} (-1)^{s\cdot z} w(s) = 0.$$

Considering $w(s)$ as a function $w : \{0,1\}^n \to \mathbb{R}$ such that $w(s) = 0$ for $|s| > k$, in Fourier-analytic terminology the constraint says that

$$\widehat{w}(z) = 0 \text{ if } z \notin S_f,$$

i.e. that $\widehat{w}$ is only supported on the subspace S_f. This is equivalent to the constraint that

$$w(s) = \frac{1}{|S_f^\perp|}\sum_{t\in S_f^\perp} w(s+t) \text{ for all } s.$$

To see this, define the function $P_{S_f}(x) = [x \in S_f]$, and note that $\widehat{P_{S_f}}(t) = \frac{1}{|S_f^\perp|}[t \in S_f^\perp]$. Letting $*$ denote convolution over $\mathbb{Z}_2^n$ (i.e. $(f*g)(x) = \sum_y f(y)g(x+y)$), by Fourier duality we have

$$\begin{aligned} P_{S_f}w = w \quad &\Leftrightarrow \quad \widehat{P_{S_f}} * \widehat{w} = \widehat{w} \Leftrightarrow \sum_t w(s+t)\widehat{P_{S_f}}(t) = w(s) \text{ for all } s \\ &\Leftrightarrow \quad \frac{1}{|S_f^\perp|}\sum_{t\in S_f^\perp} w(s+t) = w(s) \text{ for all } s. \end{aligned}$$

Thus w is uniform on cosets of $S_f^\perp$. As w is not identically zero, there must be a coset $t + S_f^\perp$ such that every element $s \in t + S_f^\perp$ has Hamming weight at most k. If $S_f = 0$, then $S_f^\perp = \{0,1\}^n$ and hence has only one coset, which contains 1^n. Hence we must have $k = n$. More generally, we have that $Q_E^{na}(f)$ is the minimal k such that there exists a t satisfying $|s| \leq k$ for all $s \in t + S_f^\perp$. In other words, $Q_E^{na}(f)$ is the minimal k such that there exists a t satisfying $d(s,t) \leq k$ for all $s \in S_f^\perp$. □

We observe from this proof that it is without loss of generality that any nonadaptive exact quantum algorithm can be described as picking a coset of $S_f^\perp$ and querying everything in that subset uniformly. Explicitly, we have the following algorithm.

1. Let $t \in \{0,1\}^n$ be a bit-string such that $d(t, S_f^\perp) = k$.
2. Produce the state of n qubits $\frac{1}{|S_f^\perp|^{1/2}} \sum_{s \in t + S_f^\perp} (-1)^{s \cdot x} |s\rangle$ at a cost of k queries to the oracle.
3. Perform Hadamards on every qubit of the resulting state and measure to get outcome $\tilde{x}$.
4. Output $f(\tilde{x})$.

One can easily verify that in fact $f(\tilde{x}) = f(x)$ with certainty. We note that this is reminiscent of an algorithm of van Dam [7] which learns x itself with bounded error using $n/2 + O(\sqrt{n})$ queries to the oracle. Here, we also compute a partial Fourier transform from which $f(x)$ can be determined, but our algorithm succeeds with certainty.

We now draw some corollaries from Theorem 7.

Corollary 8. *For any boolean function $f : \{0,1\}^n \to \{0,1\}$ such that f is not invariant under any translation, $Q_E^{na}(f) = n$.*

Proof. If f is not invariant under any translation, $S_f = \emptyset$ and hence $S_f^\perp = \{0,1\}^n$. For any bit-string $x \in \{0,1\}^n$, there exists a $y \in \{0,1\}^n$ such that $d(x,y) = n$. Hence $Q_E^{na}(f) = n$. □

One could also have observed this corollary by noting that $Q_E^{na}(f)$ depends only on S_f, and for the AND function (which has $Q_E(\text{AND}) = n$ [3]), $S_{\text{AND}} = \emptyset$. The corollary implies that only an exponentially small fraction of boolean functions f have $Q_E^{na}(f) < n$. One class of functions that do satisfy this is given by the following corollary.

Corollary 9. *For any boolean function $f : \{0,1\}^n \to \{0,1\}$ such that $f(x) = f(\bar{x})$ for all x, $Q_E^{na}(f) \leq n - 1$.*

Proof. The constraint $f(x) = f(\bar{x})$ for all x is equivalent to $f(x + 1^n) = f(x)$ for all x, so $\{0^n, 1^n\} \subseteq S_f$, implying $S_f^\perp \subseteq \{x : |x| \text{ even}\}$. If n is odd, then $d(0^n, S_f^\perp) \leq n - 1$ (as 1^n has odd Hamming weight, there is no even weight bit string distance n from 0^n). Similarly, if n is even, $d(10^{n-1}, S_f^\perp) \leq n - 1$. Hence $Q_E^{na}(f) \leq n - 1$. □

An explicit nonadaptive quantum algorithm achieving this query complexity proceeds as follows. Evaluate $y_k := x_1 \oplus x_k$, for $2 \leq k \leq n$, at a cost of $n-1$ queries in total, then output $f(0, y_2, \ldots, y_n)$. If $x_1 = 0$, this is simply $f(x)$, while if $x_1 = 1$, this is $f(\bar{x}) = f(x)$.

Corollary 10. *For any boolean function $f : \{0,1\}^n \to \{0,1\}$ that depends on all n input bits, $Q_E^{na}(f) \geq \lceil n/2 \rceil$.*

Proof. We need to show that, for any bit-string x, we can find an element $y \in S_f^\perp$ such that $d(x,y) \geq n/2$. As f depends on all its input bits, for all $i \in \{1, \ldots, n\}$, $e_i \notin S_f$. This implies that, for all $i \in \{1, \ldots, n\}$, there is at least one element of $S_f^\perp$ whose i'th bit is 1. Thus, if we pick an element $y \in S_f^\perp$ at random, each bit of y will be 0 or 1 with equal probability, so the expectation of $d(x,y)$ is exactly $n/2$. Therefore, $d(x, S_f^\perp) \geq n/2$. □

Corollary 10 was previously proven in [16] via a different method. We can also show that functions whose nonadaptive exact quantum query complexity is minimal are of very restricted form.

Corollary 11. *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function that depends on all n input bits and such that $Q_E^{na}(f) = n/2$. Then there exists an x such that $d(x,y) = n/2$ for all $y \in S_f^\perp$.*

Proof. By the proof of Corollary 10, $\mathbb{E}_{y \in S_f^\perp} d(x,y) = n/2$. Thus, if $d(x,y) \leq n/2$ for *all* $y \in S_f^\perp$, we must have $d(x,y) = n/2$ for all $y \in S_f^\perp$. □

7.1 Symmetric boolean functions

It turns out that we can apply Theorem 7 to completely characterise the nonadaptive exact quantum query complexity of symmetric boolean functions, via the following quadrichotomy.

Theorem 12. *Let $f : \{0,1\}^n \to \{0,1\}$ be symmetric. Then exactly one of the following four possibilities is true.*

1. *f is constant and $Q_E^{na}(f) = 0$.*
2. *f is the PARITY function or its negation and $Q_E^{na}(f) = \lceil n/2 \rceil$.*
3. *f satisfies $f(x) = f(\bar{x})$ (but is not constant, the PARITY function or its negation) and $Q_E^{na}(f) = n - 1$.*
4. *f is none of the above and $Q_E^{na}(f) = n$.*

We will prove Theorem 12 using the following lemma, whose proof is given afterwards.

Lemma 13. *Let $f : \{0,1\}^n \to \{0,1\}$ be symmetric and satisfy $f(x) = f(x+a)$ for all $x \in \{0,1\}^n$, for some a with $1 \leq |a| \leq n-1$. Then, if $|a|$ is odd, f is constant. If $|a|$ is even, f is constant, PARITY or its negation.*

Proof of Theorem 12. First assume there is an a with $1 \leq |a| \leq n-1$ such that $f(x) = f(x+a)$ for all $x \in \{0,1\}^n$. Then, by Lemma 13, f is constant, PARITY or its negation. If f is constant then clearly $Q_E^{na}(f) = 0$. If f is PARITY or its negation, by the result [10] of Farhi et al. $Q_E^{na}(f) = \lceil n/2 \rceil$. On the other hand, if there is no a with $1 \leq |a| \leq n-1$ such that $f(x) = f(x+a)$ for all $x \in \{0,1\}^n$, but there is such an a with $|a| = n$, $f(x) = f(\bar{x})$ and by Corollary 9 $Q_E^{na}(f) = n - 1$. Finally, if there is no $a \neq 0^n$ such that $f(x) = f(x+a)$ for all $x \in \{0,1\}^n$, by Corollary 8 $Q_E^{na}(f) = n$. □

It will be convenient to prove Lemma 13 using Fourier analysis, based on the following well-known fact.

Fact 14. *For any $f : \{0,1\}^n \to \mathbb{R}$ and for any $a \in \{0,1\}^n$, if $f(x) = f(x+a)$ for all $x \in \{0,1\}^n$, then for all b such that $|a \wedge b|$ is odd, $\hat{f}(b) = 0$.*

Proof of Lemma 13. Note that, because f is symmetric, if $\hat{f}(s) = 0$ for some s with $|s| = k$, $\hat{f}(t) = 0$ for *all* t with $|t| = k$. Without loss of generality, assume a consists of j ones followed by $n-j$ zeroes (i.e. is of the form $1 \ldots 10 \ldots 0$). Consider the bit-string s which is 1 on the set $\{1, \ldots, 2k+1\}$ for some $0 \leq k \leq (j-1)/2$, and the set $\{n-\ell+1, \ldots, \ell\}$, for some $1 \leq \ell \leq j$. By Fact 14, for any such bit-string s, $\hat{f}(s) = 0$. By varying k and ℓ, we can vary $|s|$ arbitrarily between 1 and either n

(if $|a|$ is odd), or $n-1$ (if $|a|$ is even). Thus, if $|a|$ is odd, $\hat{f}(s) = 0$ for all $s \neq 0^n$, so f is constant. Otherwise, if $|a|$ is even, $\hat{f}(s) = 0$ for all $s \notin \{0^n, 1^n\}$. The only boolean functions satisfying this are constant functions, PARITY and its negation. □

8 Open problems

It is a very tempting conjecture that $Q_E(\mathrm{EXACT}_k) = \max\{k, n-k\}$. It is easy to see that the lower bound $Q_E(\mathrm{EXACT}_k) \geq \max\{k, n-k\}$ holds; by setting $\min\{k, n-k\}$ input bits to 0 we obtain a function equivalent to the AND function on $\ell := \max\{k, n-k\}$ bits, which has exact quantum query complexity ℓ. So it would suffice to prove the upper bound $Q_E(\mathrm{EXACT}_{n/2}) \leq n/2$ for all even n to prove this conjecture. To see this, note that for any ℓ, an algorithm for the function EXACT_ℓ on n bits gives an algorithm for EXACT_ℓ on $n' \geq n$ bits, simply by appending extra zero bits.

More generally, the problem of finding any boolean function f such that $Q_E(f) < D(f)/2$ still remains. We are hopeful that the numerical techniques used in this paper may prove helpful in resolving this question.

Acknowledgements

AM was supported by an EPSRC Postdoctoral Research Fellowship.

A Numerical results for functions on up to 6 bits

In this appendix we collate our numerical results concerning the optimal success probability achievable by quantum algorithms for all boolean functions on 4 input bits, and symmetric boolean functions on 5 and 6 input bits. We split the results into sections according to the number of bits on which the functions depend. Note that each section on functions of k bits does not include functions which only depend on fewer than k bits. In the following tables, entries are starred when there is a nonadaptive exact quantum algorithm using that number of queries (see Section 7). An entry "1" means that the SDP solver claims a solution with success probability greater than 0.999; note that this does not strictly speaking imply the existence of an *exact* algorithm using that number of queries. We use the notation $\mathrm{SYM}(c_0, \ldots, c_n)$ to mean the symmetric function f such that $f(x) = c_{|x|}$. In the tables of symmetric functions, we simply identify each function with the vector $(c_0, \ldots, c_n)$.

Note that for all non-constant symmetric f, the decision tree complexity $D(f) = n$, but this is not the case for $Q_E(f)$. For most functions f on 4 bits, $D(f)$ is easily verified to be 4 via polynomial degree arguments; we calculated $D(f)$ for the remaining functions f in an ad hoc fashion.

A.1 Functions of 4 bits

ID	Function	1 query	2 queries	3 queries	D(f)
1	$x_1 \wedge x_2 \wedge x_3 \wedge x_4$	0.735	0.962	0.996	4
6		0.654	0.931	1*	4
7		0.750	0.954	1	4
22		0.572	0.906	1	4
23		0.667	0.926	1	4
24	$x_1 \wedge \neg\mathrm{NAE}(\bar{x}_2, x_3, x_4)$	0.654	0.931	1*	4
25		0.640	0.961	1	4
27	$x_1 \wedge \mathrm{SEL}(x_4, x_2, x_3)$	0.667	0.965	1	3
30		0.600	0.956	1	4
31		0.718	0.970	1	4
61		0.643	0.976	1	4
105		0.500	0.900	1*	4
107		0.571	0.941	1	4
111		0.662	0.968	1*	4
126	$x_1 \wedge \mathrm{NAE}(x_2, x_3, x_4)$	0.667	0.947	1*	4
127		0.727	0.972	1	4
278	EXACT_3	0.529	0.884	1	4
279	Th_3	0.643	0.900	1	4
280		0.572	0.906	1	4
281		0.600	0.956	1	4
282		0.571	0.936	1	4
283		0.637	0.959	1	4
286		0.546	0.932	1	4
287		0.659	0.945	1	4
300		0.571	0.936	1	4
301		0.572	0.964	1	4
303	$\mathrm{SEL}(x_3, x_1 \wedge x_2, \mathrm{SEL}(x_4, x_1, x_2))$	0.644	0.966	1	3
316		0.562	0.962	1	4
317	$\mathrm{SEL}(x_3, x_1 \wedge x_2, \mathrm{SEL}(x_2, x_1, x_4))$	0.572	0.980	1	3
318		0.546	0.956	1	4
319		0.640	0.972	1	4
360		0.529	0.884	1	4
361		0.500	0.916	1	4
362		0.546	0.932	1	4
363		0.546	0.955	1	4
366		0.546	0.956	1	4
367		0.571	0.969	1	4
382		0.546	0.923	1	4
383		0.600	0.946	1	4
384	$\mathrm{NAE}(\bar{x}_1, x_2, x_3, x_4)$	0.800	0.980	1*	4
385		0.750	0.954	1	4
386		0.640	0.961	1	4
387		0.667	0.965	1	4
390		0.571	0.936	1	4
391		0.637	0.959	1	4

ID	Function	1 query	2 queries	3 queries	D(f)
393	$\mathrm{SEL}(x_3, x_1 \wedge \bar{x}_4, x_2 \wedge x_4)$	0.667	0.965	1	3
395		0.724	0.963	1	4
399		0.751	0.980	1	4
406		0.500	0.916	1	4
407		0.572	0.940	1	4
408		0.600	0.956	1	4
409		0.643	0.976	1	4
410		0.572	0.964	1	4
411		0.656	0.969	1	4
414		0.546	0.955	1	4
415		0.642	0.965	1	4
424		0.667	0.926	1	4
425		0.637	0.959	1	4
426		0.718	0.970	1	4
427	$\mathrm{SEL}(x_3, x_1 \wedge \bar{x}_4, \mathrm{SEL}(x_4, x_1, x_2))$	0.751	0.980	1	3
428		0.637	0.959	1	4
429	$\mathrm{SEL}(x_2, x_1 \wedge \bar{x}_4, \mathrm{SEL}(x_3, x_1, x_4))$	0.656	0.969	1	3
430		0.644	0.966	1	4
431		0.710	0.977	1	4
444		0.572	0.980	1	4
445		0.641	0.965	1	4
446		0.572	0.969	1	4
447		0.667	0.980	1	4
488		0.643	0.900	1	4
489		0.572	0.940	1	4
490		0.659	0.945	1	4
491		0.642	0.965	1	4
494		0.640	0.972	1	4
495	$\mathrm{SEL}(x_3, x_1, \mathrm{SEL}(x_4, x_1, x_2))$	0.667	0.980	1	3
510		0.600	0.946	1	4
829		0.563	0.975	1	4
854		0.598	0.955	1	4
855		0.714	0.969	1	4
856		0.572	0.964	1	4
857		0.579	0.961	1	4
858	$\mathrm{SEL}(x_1, x_2 \wedge x_3, x2 \oplus x_4)$	0.572	0.980	1	3
859		0.628	0.974	1	4
862		0.572	0.966	1	4
863	$\mathrm{SEL}(x_1, x_2 \wedge x_3, x2 \vee x_4)$	0.667	0.986	1	3
872		0.546	0.932	1	4
873		0.500	0.946	1	4
874		0.598	0.955	1	4
875		0.572	0.951	1	4
876		0.546	0.956	1	4
877		0.545	0.961	1	4
878		0.572	0.966	1	4
879		0.600	0.966	1	4
892		0.563	0.975	1	4

ID	Function	1 query	2 queries	3 queries	D(f)
893		0.571	0.966	1	4
894		0.572	0.947	1	4
961		0.718	0.970	1	4
965	$\mathrm{SEL}(x_2, x_1 \wedge \bar{x}_3, \mathrm{SEL}(x_1, x_3, x_4))$	0.751	0.980	1	3
966		0.644	0.966	1	4
967		0.710	0.977	1	4
980		0.659	0.945	1	4
981		0.714	0.969	1	4
982		0.572	0.951	1	4
983		0.661	0.965	1	4
984	$\mathrm{SEL}(x_1, x_2 \wedge x_3, \mathrm{SEL}(x_4, \bar{x}_3, \bar{x}_2))$	0.644	0.966	1	3
985		0.628	0.974	1	4
987	$\mathrm{SEL}(x_4, \mathrm{SEL}(x_3, x_1, x_2), \mathrm{SEL}(x_2, x_1, x_3))$	0.661	0.965	1	3
988		0.640	0.972	1	4
989	$\mathrm{SEL}(x_1, x_2 \wedge x_3, \bar{x}_3 \vee x_4)$	0.667	0.986	1	3
990		0.600	0.966	1	4
1632	$(x_1 \oplus x_2) \wedge (x_3 \oplus x_4)$	0.667	1*	1*	4
1633		0.562	0.962	1	4
1634		0.643	0.976	1	4
1635		0.572	0.980	1	4
1638		0.667	0.947	1*	4
1639		0.641	0.965	1	4
1641		0.500	0.936	1*	4
1643		0.561	0.966	1	4
1647	$\mathrm{MAJ}(x_1, x_2, x_3 \oplus x_4)$	0.667	1	1*	4
1650		0.656	0.969	1	4
1651		0.628	0.974	1	4
1654		0.641	0.965	1	4
1656		0.546	0.956	1	4
1657		0.500	0.964	1	4
1658		0.571	0.966	1	4
1659		0.571	0.962	1	4
1662		0.600	0.954	1	4
1680	$(x_1 \oplus x_2) \wedge (x_1 \oplus x_3 \oplus x_4)$	0.500	0.900	1*	4
1681		0.500	0.916	1	4
1683		0.500	0.946	1	4
1686		0.500	0.936	1*	4
1687		0.500	0.964	1	4
1695	$\mathrm{SEL}(x_3 \oplus x_4, x_1, x_2)$	0.500	1	1*	3
1712		0.571	0.941	1	4
1713		0.546	0.955	1	4
1714		0.572	0.940	1	4
1715		0.572	0.951	1	4
1716		0.546	0.955	1	4
1717		0.545	0.961	1	4
1718		0.561	0.966	1	4
1719	$\mathrm{SEL}(x_4, \mathrm{SEL}(x_2, x_1, x_3), \mathrm{SEL}(x_3, x_2, x_1))$	0.572	0.962	1	3
1721		0.500	0.964	1	4

ID	Function	1 query	2 queries	3 queries	D(f)
1725		0.529	0.955	1	4
1776		0.662	0.967	1*	4
1777		0.572	0.969	1	4
1778		0.642	0.965	1	4
1782	$\mathrm{SEL}(x_2, x_1, x_3 \oplus x_4)$	0.667	1	1*	3
1785	$x_1 \oplus (x_2 \wedge (x_3 \oplus x_4))$	0.500	1	1*	4
1910		0.600	0.954	1	4
1912		0.546	0.923	1	4
1913		0.529	0.955	1	4
1914		0.572	0.947	1	4
1918		0.572	0.922	1	4
1968		0.662	0.968	1*	4
1969		0.642	0.965	1	4
1972		0.572	0.969	1	4
1973		0.600	0.966	1	4
1974		0.572	0.962	1	4
1980	$\mathrm{SEL}(x_3, \mathrm{SEL}(x_4, x_1, x_2), x_1 \oplus x_2)$	0.571	0.966	1	3
2016	$\mathrm{SEL}(x_1, x_2 \wedge (x_3 \vee x_4), \bar{x}_2 \wedge (\bar{x}_3 \vee \bar{x}_4))$	0.773	1	1*	4
2017		0.640	0.972	1	4
2018		0.710	0.977	1	4
2019		0.667	0.986	1	4
2022	$\mathrm{SEL}(x_3, \mathrm{SEL}(x_2, x_1, x_4), \mathrm{SEL}(x_1, x_2, \bar{x}_4))$	0.661	0.965	1	3
2025		0.571	0.966	1	4
2032		0.727	0.971	1	4
2033		0.667	0.980	1	4
2034	$\mathrm{SEL}(x_2, x_1, \mathrm{SEL}(x_4, x_3, \bar{x}_1))$	0.667	0.980	1	3
2040		0.600	0.946	1	4
5736	EXACT_2	0.572	1	1*	4
5737	SYM(0,0,1,0,1)	0.500	0.962	1	4
5738		0.563	0.975	1	4
5739		0.500	0.980	1	4
5742		0.572	0.947	1	4
5758		0.572	0.922	1	4
5761		0.500	0.860	1	4
5763		0.500	0.907	1	4
5766		0.500	0.936	1*	4
5767		0.500	0.933	1	4
5769		0.500	0.907	1	4
5771		0.500	0.946	1	4
5774		0.561	0.966	1	4
5782		0.500	0.962	1	4
5783		0.500	0.954	1	4
5784		0.500	0.946	1	4
5785		0.500	0.933	1	4
5786		0.500	0.964	1	4
5787		0.500	0.955	1	4
5790	$\mathrm{SEL}(x_3, \mathrm{SEL}(x_4, x_1, x_2), x_2 \oplus x_4)$	0.500	0.980	1	3
5801		0.500	0.933	1	4

ID	Function	1 query	2 queries	3 queries	D(f)
5804		0.545	0.961	1	4
5805		0.500	0.955	1	4
5820		0.529	0.955	1	4
5865		0.500	0.954	1	4
6014	SYM(0,0,1,1,0)	0.600	0.874	1	4
6030	$\mathrm{SEL}(x_3, \mathrm{SEL}(x_4, x_1, x_2), \mathrm{SEL}(x_4, x_2, \bar{x}_1))$	0.667	1	1*	3
6038		0.500	0.980	1	4
6040		0.572	0.951	1	4
6042	$\mathrm{SEL}(x_4, \mathrm{SEL}(x_3, x_1, x_2), \mathrm{SEL}(x_2, x_3, \bar{x}_1))$	0.572	0.962	1	3
6060		0.600	0.966	1	4
6120	$x_1 \oplus \mathrm{MAJ}(x_2, x_3, x_4)$	0.667	1	1*	4
6375	$x_1 \oplus \neg\mathrm{NAE}(\bar{x}_2, x_3, x_4)$	0.500	0.900	1*	4
6625		0.500	0.946	1	4
6627		0.500	0.955	1	4
6630		0.500	0.954	1	4
7128	Sorted input bits [1]	0.854	1	1*	3
7140	$x_1 \oplus \mathrm{SEL}(x_4, x_2, x_3)$	0.500	1	1*	3
7905		0.500	0.900	1*	4
27030	PARITY	0.500	1*	1*	4

A.2 Symmetric functions of 5 bits

Function	1 query	2 queries	3 queries	4 queries
(0,0,0,0,0,1)	0.693	0.925	0.988	0.999
(0,0,0,0,1,0)	0.516	0.761	0.972	1
(0,0,0,0,1,1)	0.640	0.798	0.974	1
(0,0,0,1,0,0)	0.530	0.616	1	1
(0,0,0,1,0,1)	0.500	0.593	0.995	1
(0,0,0,1,1,0)	0.546	0.758	1	1
(0,0,0,1,1,1)	0.600	0.728	1	1
(0,0,1,0,0,1)	0.500	0.640	0.988	1
(0,0,1,0,1,0)	0.500	0.517	1	1
(0,0,1,0,1,1)	0.500	0.534	1	1
(0,0,1,1,0,0)	0.600	0.874	1	1*
(0,0,1,1,0,1)	0.500	0.856	0.998	1
(0,0,1,1,1,0)	0.616	0.762	0.969	1
(0,1,0,0,0,1)	0.500	0.728	0.967	1
(0,1,0,0,1,0)	0.500	0.860	1	1*
(0,1,0,1,0,1)	0.500	0.500	1*	1*
(0,1,0,1,1,0)	0.500	0.616	0.998	1
(0,1,1,0,0,1)	0.500	0.784	0.998	1
(0,1,1,1,1,0)	0.736	0.962	0.996	1*

A.3 Symmetric functions of 6 bits

Function	1 query	2 queries	3 queries	4 queries	5 queries
(0,0,0,0,0,0,1)	0.663	0.900	0.980	0.997	0.9999
(0,0,0,0,0,1,0)	0.511	0.684	0.940	0.993	1
(0,0,0,0,0,1,1)	0.640	0.738	0.946	0.993	1
(0,0,0,0,1,0,0)	0.516	0.572	0.878	1	1
(0,0,0,0,1,0,1)	0.500	0.541	0.875	0.999	1
(0,0,0,0,1,1,0)	0.527	0.751	0.904	1	1
(0,0,0,0,1,1,1)	0.589	0.710	0.901	1	1
(0,0,0,1,0,0,0)	0.530	0.616	1	1	1*
(0,0,0,1,0,0,1)	0.500	0.614	0.980	0.997	1
(0,0,0,1,0,1,0)	0.500	0.504	0.946	1	1
(0,0,0,1,0,1,1)	0.500	0.525	0.952	1	1
(0,0,0,1,1,0,0)	0.546	0.667	0.864	1	1
(0,0,0,1,1,0,1)	0.500	0.625	0.860	1	1
(0,0,0,1,1,1,0)	0.556	0.721	0.905	1	1
(0,0,1,0,0,0,1)	0.500	0.583	0.882	0.997	1
(0,0,1,0,0,1,0)	0.500	0.527	0.839	1	1
(0,0,1,0,0,1,0)	0.500	0.541	0.843	1	1
(0,0,1,0,1,0,0)	0.500	0.517	1	1	1*
(0,0,1,0,1,0,1)	0.500	0.500	0.985	1	1
(0,0,1,0,1,1,0)	0.500	0.520	0.940	1	1
(0,0,1,1,0,0,1)	0.500	0.712	0.867	1	1
(0,0,1,1,0,1,0)	0.500	0.513	0.840	1	1
(0,0,1,1,1,0,0)	0.616	0.762	0.969	1	1*
(0,0,1,1,1,0,1)	0.500	0.722	0.965	1	1
(0,0,1,1,1,1,0)	0.625	0.702	0.939	0.992	1
(0,1,0,0,0,0,1)	0.500	0.652	0.934	0.992	1
(0,1,0,0,0,1,0)	0.500	0.728	0.967	1	1*
(0,1,0,0,1,0,1)	0.500	0.500	0.836	1	1
(0,1,0,0,1,1,0)	0.500	0.667	0.863	1	1
(0,1,0,1,0,0,1)	0.500	0.500	0.934	1	1
(0,1,0,1,0,1,0)	0.500	0.500	1*	1*	1*
(0,1,0,1,1,1,0)	0.500	0.553	0.880	0.999	1
(0,1,1,0,0,0,1)	0.500	0.758	0.908	1	1
(0,1,1,0,1,1,0)	0.500	0.640	0.988	1	1*
(0,1,1,1,1,1,0)	0.693	0.925	0.988	0.999	1*

B Source code

The following is an example of how the CVX package [11] can be used to determine quantum query complexity. In this case, we calculate the minimal error probability over all quantum algorithms using 2 queries to compute some function $f : \{0,1\}^3 \to \{0,1\}$ (given as a column vector).

```
cvx_begin
```

```
  cvx_precision best;

  % variables m_i^j : 0 <= i <= n, 0 <= j <= t-1
  variable m00(8,8) symmetric; variable m10(8,8) symmetric;
  variable m20(8,8) symmetric; variable m30(8,8) symmetric;
  variable m01(8,8) symmetric; variable m11(8,8) symmetric;
  variable m21(8,8) symmetric; variable m31(8,8) symmetric;

  variable g0(8,8) symmetric; variable g1(8,8) symmetric;

  variable epss;

  minimise( epss );

  subject to

    % Input condition.
    m00 + m10 + m20 + m30 == ones(8,8);

    % Running conditions (between 1 and t-1).
    m01 + m11 + m21 + m31 == E0 .* m00 + E1 .* m10 + E2 .* m20 + E3 .* m30;

    % Output matches last but one query.
    g0 + g1 == E0 .* m01 + E1 .* m11 + E2 .* m21 + E3 .* m31;

    % Output constraints.
    diag(g0) >= (1-epss)*(1-f);
    diag(g1) >= (1-epss)*f;

    % Semidefinite constraints.
    m00 == semidefinite(8); m10 == semidefinite(8);
    m20 == semidefinite(8); m30 == semidefinite(8);
    m01 == semidefinite(8); m11 == semidefinite(8);
    m21 == semidefinite(8); m31 == semidefinite(8);

    g0 == semidefinite(8); g1 == semidefinite(8);

cvx_end
```

References

[1] A. Ambainis. Polynomial degree vs. quantum query complexity. *J. Comput. Syst. Sci.*, 72(2):220–238, 2006. quant-ph/0305028.

[2] H. Barnum, M. Saks, and M. Szegedy. Quantum query complexity and semi-definite programming. In *Proc. 18th Annual IEEE Conf. Computational Complexity*, pages 179–193, 2003.

[3] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. quant-ph/9802049.

[4] G. Brassard and P. Høyer. An exact quantum polynomial-time algorithm for Simon's problem. In *Theory of Computing and Systems, Proceedings of the Fifth Israeli Symposium on*, pages 12–23, 1997. quant-ph/9704027.

[5] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proc. R. Soc. Lond. A*, 454(1969):339–354, 1998. quant-ph/9708016.

[6] G. Cohen, M. Karpovsky, H. Mattson, Jr., and J. Schatz. Covering radius – survey and recent results. *IEEE Trans. Inform. Theory*, 31(3):328–343, 1985.

[7] W. van Dam. Quantum oracle interrogation: Getting all information for almost half the price. In *Proc. 39th Annual Symp. Foundations of Computer Science*, pages 362–367. IEEE, 1998. quant-ph/9805006.

[8] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proc. Roy. Soc. London Ser. A*, 439(1907):553–558, 1992.

[9] Alina Dubrovska and Taisija Mischenko-Slatenkova. Computing boolean functions: Exact quantum query algorithms and low degree polynomials, 2006. quant-ph/0607022.

[10] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. A limit on the speed of quantum computation in determining parity. *Phys. Rev. Lett.*, 81:5442–5444, 1998. quant-ph/9802045.

[11] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. http://cvxr.com/cvx, April 2011.

[12] T. Hayes, S. Kutin, and D. van Melkebeek. The quantum black-box complexity of majority. *Algorithmica*, 34(4):480–501, 2002. quant-ph/0109101.

[13] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[14] P. Høyer and R. Špalek. Lower bounds on quantum query complexity. *Bulletin of the European Association for Theoretical Computer Science*, 87:78–103, 2005. quant-ph/0509153.

[15] G. Midrijānis. Exact quantum query complexity for total Boolean functions, 2004. quant-ph/0403168.

[16] A. Montanaro. Nonadaptive quantum query complexity. *Information Processing Letters*, 110(24):1110–1113, 2010. arXiv:1001.0018.

[17] B. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. In *Proc. 50th Annual Symp. Foundations of Computer Science*, pages 544–551, 2009. arXiv:0904.2759.

[18] B. Reichardt and R. Špalek. Span-program-based quantum algorithm for evaluating formulas. In *Proc. 40th Annual ACM Symp. Theory of Computing*, pages 103–112, 2008. arXiv:0710.2630.

[19] Alina Vasilieva. Quantum query algorithm constructions for computing AND, OR and MAJORITY boolean functions, 2007. arXiv:0710.5592.

[20] Alina Vasilieva. Exact quantum query algorithm for error detection code verification, 2009. arXiv:0904.3660.

Classical simulation complexity of extended Clifford circuits

Richard Jozsa[1] and Maarten Van den Nest[2]

[1]*DAMTP, Centre for Mathematical Sciences, University of Cambridge, Wilberforce Road, Cambridge CB3 0WA, U.K.*

[2]*Max Planck Institut für Quantenoptik, Hans-Kopfermann-Str. 1, D-85748 Garching, Germany.*

Abstract

Clifford gates are a winsome class of quantum operations combining mathematical elegance with physical significance. The Gottesman-Knill theorem asserts that Clifford computations can be classically efficiently simulated but this is true only in a suitably restricted setting. Here we consider Clifford computations with a variety of additional ingredients: (a) strong vs. weak simulation, (b) inputs being computational basis states vs. general product states, (c) adaptive vs. non-adaptive choices of gates for circuits involving intermediate measurements, (d) single line outputs vs. multi-line outputs. We consider the classical simulation complexity of all combinations of these ingredients and show that many are not classically efficiently simulatable (subject to common complexity assumptions such as P not equal to NP). Our results reveal a surprising proximity of classical to quantum computing power viz. a class of classically simulatable quantum circuits which yields universal quantum computation if extended by a purely classical additional ingredient that does not extend the class of quantum processes occurring.

1 Introduction

The notion of classical simulation of quantum computation provides a mathematically precise tool for studying fundamental questions that are often only vaguely formulated – questions of the relationship between classical and quantum computing power and the computational possibilities engendered by particular kinds of quantum resources. We may consider a restricted class $\mathcal{A}$ of quantum circuits defined by specified limited quantum ingredients and ask whether it can be classically efficiently simulated or not. Computational hardness is notoriously difficult to establish and in the latter case we are generally content to establish that the efficient simulation of $\mathcal{A}$ would imply some further property such as P=NP, that is widely regarded as implausible. In the former case of $\mathcal{A}$ being classically efficiently simulatable we may consider enlarging $\mathcal{A}$ to $\mathcal{A}'$ by inclusion of some extra specific quantum ingredient P and investigating the resulting change in classical simulation complexity. If for example, $\mathcal{A}'$ allows universal quantum computation then in this mathematically precise sense, P may be regarded as an "essential resource for quantum computational power" (relative to a background of quantum effects that are "computationally lame"). Below we will

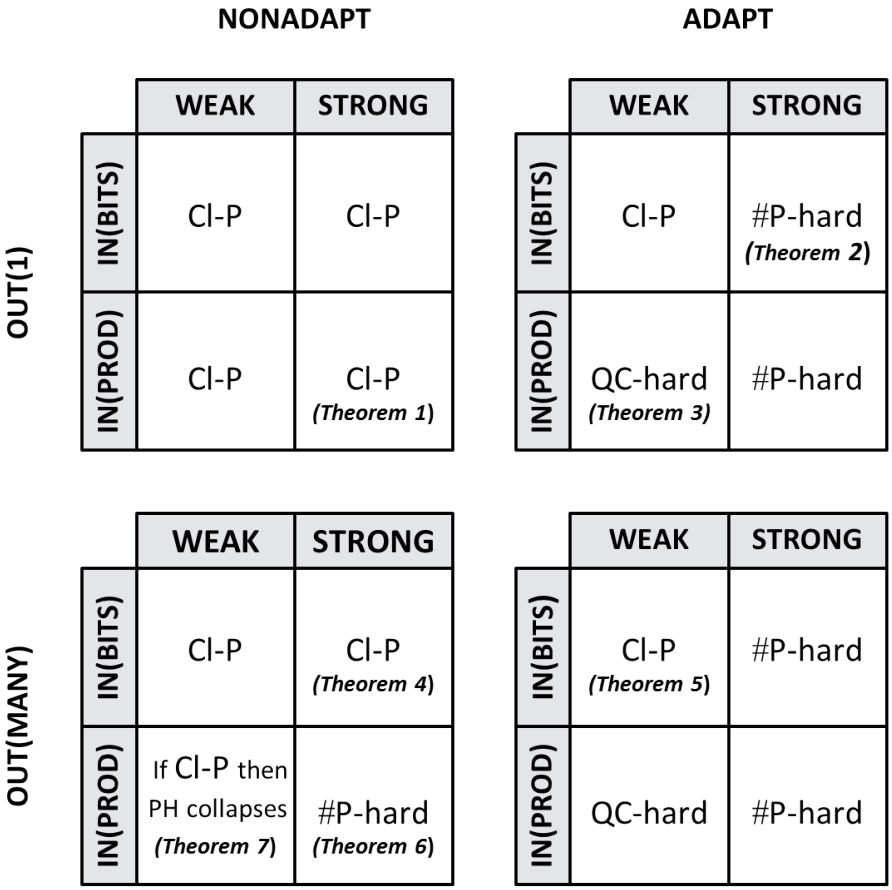

Figure 1: Classical simulation complexities for sets of Clifford computational tasks. The acronyms are as defined in the main body of the text. The seven cases containing numbered theorems are proved in section 4. All other cases are easily seen to be implied by these seven cases.

see examples of seemingly quite modest expansions of resources leading to dramatic changes in simulation complexity, indicating that the computational landscape between classical and quantum computing power is a richly complex one.

This paper is devoted to developing a case study of simulation of Clifford circuits supplemented with a variety of extra ingredients. The choice of Clifford circuits is a particularly interesting and relevant one for a variety of reasons. Clifford computations provide one of the earliest significant examples of classically simulatable quantum computations in the Gottesman-Knill theorem [1] (see also [2, 3, 4, 5, 6]), showing in particular that the presence of non-trivial entanglements in a quantum computation is not necessarily a signature of computational speed-up. Clifford gates also have a rich associated pure mathematical theory that may be drawn upon in the study of simulation properties, as well as having a rich physical and practical significance in the theory and implementation of quantum computation. For example Clifford operations feature prominently in the theory of quantum error correction and fault tolerance [7, 2, 8] and in measurement-based quantum computation

[9, 10].

It is well known that the Clifford gates supplemented with any non-Clifford operation generate a dense subgroup of $U(2^n)$ and are hence universal for quantum computation [11, 12]. Here we will consider extensions of Clifford circuits of a different, perhaps seemingly more innocuous kind. More precisely we will characterise the classical simulation complexity of sixteen cases of extended Clifford circuits that are defined by four binary choices. Our main results are summarised in figure 1. The acronyms in figure 1 that define the extensions and their classical simulation complexities are all explained in detail in section 2 below, and briefly they are as follows: IN(BITS) and IN(PROD) refer to allowing computational basis states and general product states as inputs. OUT(1) and OUT(MANY) refer to having single bit and multi-bit outputs. NONADAPT and ADAPT refer to circuits with intermediate measurements, with the circuit gates being respectively fixed or chosen adaptively as a function of previous measurement outcomes. WEAK and STRONG refer to two notions of classical simulation that provide respectively a sample of the output distribution and a calculation of actual probability values. In the body of the tables, Cl-P denotes that classical efficient simulation is possible, QC-hard denotes that universal quantum computation is possible, and #P-hard asserts that classical simulation could be used to solve arbitrary problems in the classical class #P (and hence NP too).

These results demonstrate a remarkable sensitivity of the classical simulation complexity of Clifford circuits under various small modifications. In section 3 we highlight some interesting comparisons amongst these simulation complexities. In particular the issue of the last sentence of the abstract above is discussed in Example 2 of section 3. Finally in section 4 we provide proofs of all results given in figure 1. For completeness we indicate proofs for all sixteen cases. Some cases were previously known (cf references in our text) but to the best of our knowledge others have not previously been given in the literature.

Finally we mention here some related work on the classical simulation complexity of various extensions and generalizations of Clifford circuits. See [4] for simulation of Clifford circuits supplemented with few non-stabilizer (pure or mixed) inputs and/or few non-Clifford gates; see [8] for quantum computing with adaptive Clifford circuits with product state inputs (we will revisit this scenario as one of the sixteen cases in figure 1); see [13, 14] for simulations of Clifford circuits supplemented with certain non-Clifford gates by restricting the circuit structure; see [15, 16, 17, 18, 19] for generalizations of the Gottesman-Knill theorem to higher-dimensional systems; see [20] for generalizations of Clifford circuits based on projective normalisers of finite unitary groups.

2 Preliminary definitions and notations

Clifford circuits: NONADAPT and ADAPT

Let I, X, Y, Z denote the standard 1-qubit Pauli matrices [2] (amongst which we include the identity matrix). An *n-qubit Pauli operator* is any operator of the form $P = \gamma P_1 \otimes \ldots \otimes P_n$ where $\gamma \in \{\pm 1, \pm i\}$ and each P_i is a Pauli matrix.

An n-qubit unitary operation C is called a *Clifford operation* if the set of all Pauli operators is preserved under conjugation by C i.e. for any n-qubit Pauli operator P, $P' =$

$CPC^\dagger$ is again a Pauli operator. It is known that C is Clifford iff C can be expressed as a circuit of the following gates (cf [2]): the 1-qubit Hadamard gate H, the phase gate $T = \mathrm{diag}(1, i)$ and the 2-qubit controlled-Z gate CZ, which we call *basic Clifford gates*. Moreover any n-qubit Clifford operation can be expressed as a circuit of $O(n^2)$ basic Clifford gates (see [3] and theorem 10.6 of [2]).

A *unitary Clifford circuit* is a circuit comprising only the basic Clifford gates. The size of the circuit is the number of gates of which it consists.

As a further extension we will allow measurements in the body of the circuit. The term *measurement* will always mean a single qubit measurement in the computational basis. Let $M_i(x)$ denote a measurement of the i^{th} qubit line with outcome $x \in \{0, 1\}$. Then a *Clifford circuit with K intermediate measurements* has the form

$$C_0\, M_{i_1}(x_1)\, C_1\, M_{i_2}(x_2)\, C_2\, \ldots\, M_{i_K}(x_{i_K})\, C_K \tag{1}$$

where C_i are unitary Clifford circuits (possibly of size zero). We assume that measurements are non-destructive and the measured qubit, set to the designated post-measurement state, may generally be an input into subsequent operations e.g. $M_{i_1}(x_1)$ sets qubit line i_1 to $|x_1\rangle$ which may then be input into C_1.

A *non-adaptive Clifford circuit* is a Clifford circuit with intermediate measurements in which the choice of operations in the circuit does not depend on the outcomes of (previous) measurements. Hence such a circuit is fully defined by eq. (1) where the C_j's and measurement line labels i_j's are fixed a priori.

By the term *adaptive Clifford circuit* we will mean a process of the form eq. (1) in which the choice of operations is allowed to depend on previous measurement outcomes. To make this dependency explicit we can expand the notation of eq. (1) as

$$C_0\, M_{i_1}(x_1)\, C_1(x_1)\, M_{i_2(x_1)}(x_2)\, C_2(x_1, x_2) \ldots\, M_{i_K(x_1,\ldots x_{K-1})}(x_k)\, C_K(x_1, \ldots, x_K). \tag{2}$$

Note that the size of the circuits $C_j(x_1, \ldots, x_j)$ may vary with $x_1, \ldots, x_j$. The total number N of operations in the adaptive circuit eq. (2) is defined to be the maximum number of elementary Clifford gates and measurements over all possible choices of measurement outcomes $x_1, \ldots, x_K$. Alternatively we could uniformise the size of each C_j by including additional identity gates to make its size independent of $x_1, \ldots, x_j$. Similarly to uniformise the number of intermediate measurements in the adaptive process (as a function of measurement outcomes) we could formally allow $i_{j+1}(i_1, \ldots, i_j)$ to be zero (for qubit lines labelled 1 to n) to indicate that a measurement is not performed, but replaced by an identity gate.

The scenarios of non-adaptive and adaptive Clifford circuits will be denoted respectively by the acronyms NONADAPT and ADAPT.

We mention here two elementary simplifications of circuit structures that will be useful in proofs of classical simulation properties. Stated informally we have the following facts (with formal statements and proofs given in lemmas 2 and 3 in section 4 below):
(i) without loss of generality (wlog) non-adaptive circuits may be assumed to be unitary;
(ii) in (adaptive or non-adaptive) circuits with intermediate measurements, wlog the measured qubits may be assumed to always be discarded after measurement (and not used in subsequent operations). Furthermore for adaptive circuits the choice of lines for intermediate measurements may be assumed to be non-adaptive.

Inputs and outputs: IN(BITS), IN(PROD), OUT(1) and OUT(MANY)

In addition to the circuit itself there are two further ingredients for the full specification of a computational process viz. specification of the input and of the output. We will distinguish two classes of input states – computational basis input states, denoted by the acronym IN(BITS), and general product state inputs, denoted IN(PROD). For outputs we will distinguish the scenarios of a single bit output (resulting from a specified final 1-qubit measurement), denoted OUT(1), and the scenario of a many-bit output, denoted OUT(MANY). In the latter case, for an n-qubit circuit the output $y_{j_1} \dots y_{j_l}$ results from a final measurement on a specified set $1 \leq j_1 < \dots < j_l \leq n$ of l lines and generally we can have $l = O(n)$.

Classical simulations: WEAK, STRONG and Cl-P

A *description of a Clifford computational task* T with N operations on n qubits is made up of the following ingredients:
(i) a description of an (adaptive, non-adaptive or unitary) Clifford circuit on n lines comprising N operations. For unitary or non-adaptive circuits we give a list of N basic Clifford gates and intermediate measurements on specified qubit lines; for adaptive circuits (cf eq. (2)) we require that each $C_j(x_1, \dots, x_j)$ and $i_{j+1}(i_1, \dots, i_j)$ is given as a function computable in classical poly(N) time;
(ii) specification of an input state $|\psi\rangle$ which we always take to be either a computational basis state or a general product state;
(iii) specification of one or more output measurement lines $1 \leq j_1 < \dots < j_l \leq n$.
Let $p(y_{j_1}, \dots, y_{j_l})$ denote the output probability distribution of the corresponding quantum process.

We will consider sets of computational tasks subject to the restrictions introduced above viz. the eight combinations of ADAPT vs. NONADAPT, IN(BITS) vs. IN(PROD) and OUT(1) vs. OUT(MANY). In each case it is natural to assume that the total length (as a classical bit string) of the full description (i), (ii) and (iii) of the computational task is $O(\text{poly}(N))$. In particular we assume that there are no extraneous qubit lines that are not acted upon (so $n = O(N)$) and we assume that input product states are specified with $O(\text{poly}(N))$ bits. The latter technical issue of accuracy (for our later purposes of simulation complexity characterisations) may be addressed by setting up a suitable notion of approximation, but we do not elaborate it here for sake of clarity and conceptual transparency.

We introduce two notions of classical simulation for Clifford computational tasks. A *weak classical simulation* for a set of computational tasks is a classical randomised computation which, given a description of a task T as input, outputs a *sample* of the output distribution $p(y_{j_1}, \dots, y_{j_l})$ of T. A *strong classical simulation* for a set of tasks is a classical computation whose input is a description of a task T and bit values for a subset of its output lines. The output is the value of the corresponding marginal probability of the output distribution of T i.e. we have a classical computation of any desired output probability or marginal probability of T.

A weak or strong classical simulation is called *efficient* if the corresponding classical computation runs in classical poly(N) time. (Again here for strong simulation, as previously

noted for IN(PROD), there is a further technical issue of precision and more formally we would require the output probability or marginal to be computed to k bits of precision in poly(N, k) time).

We will use the acronyms WEAK (resp. STRONG) to indicate that we are considering a weak (resp. strong) classical simulation for a set of tasks. We will use the acronym Cl-P ("classical poly time") to assert that the associated simulation is efficient.

If the computational task T is implemented as a quantum process it will require $O(N)$ quantum computational steps so existence of an efficient weak classical simulation implies that T offers no quantum computational time benefit over classical computation (up to the usual polynomial overheads of resources commonly accepted in complexity theory).

In the case of OUT(MANY) there are generally exponentially many output probabilities $p(y_{j_1}, \ldots, y_{j_l})$ (as l may be $O(n)$) so in strong efficient simulation we cannot ask for a computation of them all. The inclusion of computation of marginals in the definition of strong simulation guarantees the following eponymously desirable result.

Lemma 1. *Let $p(x_1, \ldots, x_n)$ be a probability distribution over n binary variables. Suppose that each of the n marginals $p(x_1), p(x_1, x_2), \ldots, p(x_1, \ldots, x_n)$ may be efficiently classically computed for any choice of values $x_1, \ldots, x_n$, and suppose that any 1-bit distribution $\{p_o, 1 - p_o\}$ with p_0 efficiently computable, may be efficiently sampled. Then $p(x_1, \ldots, x_n)$ may be efficiently sampled.*
Hence for any set of computational tasks, efficient strong classical simulation implies efficient weak classical simulation.

For a proof see proposition 1 of [21].

For completeness we mention that there are also notions of weak simulations which incorporate various types of approximations [14, 22]; these will however not be relevant for the present work.

Complexity measures: QC-hardness and #P-hardness

We will also be interested in establishing that some sets of Clifford computational tasks are unlikely to have efficient classical simulations and for this purpose we introduce some further complexity notions.

Consider the following classical computational task called #SAT (cf [23]):
INPUT: a Boolean function f from n bits to one bit (given say as a bit string encoding a formula in standard 3-cnf form [23]).
PROBLEM: determine the number $\#f$ of n-bit strings $\mathbf{x}$ with $f(\mathbf{x}) = 1$.
Note that #SAT is a generalisation of the well known NP-complete problem SAT [23] (which asks only if $\#f$ is non-zero) so it is very unlikely that #SAT has a poly time classical algorithm; indeed the latter would imply equality of the complexity classes P and NP, and also that any problem in #P may be computed in poly time [23].

A set $\mathcal{A}$ of Clifford computational tasks T is called *#P-hard* if an efficient strong simulation of $\mathcal{A}$ would give rise to an efficient classical solution of #SAT. More precisely $\mathcal{A}$ is #P-hard if given any input f of size N for the #SAT problem, it may be converted by a classical poly(N) time computation ϕ into (a description of) a task $\phi(f)$ in $\mathcal{A}$ with the

following property: $\#f$ may be computed by a classical poly(N) time algorithm from the results of strong classical simulation of $\phi(f)$. Hence efficient strong classical simulation of $\mathcal{A}$ would imply P=NP and that #P is computable in poly time.

Finally we introduce a notion of *QC-hardness* for a set $\mathcal{A}$ of Clifford computational tasks. This will be used to indicate that $\mathcal{A}$ is unlikely to have an efficient weak classical simulation. Broadly speaking $\mathcal{A}$ will be QC-hard ("quantum computing hard") if it is rich enough to encode universal quantum computation, so then efficient classical weak simulation of $\mathcal{A}$ would imply that all quantum computation could be classically efficiently simulated. More precisely we will adopt the following definition. Let $\mathcal{G}$ be the set of basic unitary Clifford gates together with the phase gate $S = \mathrm{diag}(1, e^{i\pi/4})$. It is known [2] that $\mathcal{G}$ is a universal set of gates for quantum computation. Let C be any circuit of gates from $\mathcal{G}$ with a specified computational basis state input and 1-bit output from measurement of a specified output line. Then $\mathcal{A}$ is QC-hard if any such C may be simulated by a member $\phi(C)$ of $\mathcal{A}$ i.e. $T = \phi(C)$ has 1-bit output whose probability distribution for the given computational basis input coincides with that of C. Here as before, ϕ is a poly time translation of the description of C into the description of a member of $\mathcal{A}$. Hence efficient weak classical simulation of $\mathcal{A}$ would imply efficient classical simulation of universal quantum computing.

3 Main results – statement and discussion

We now consider the sixteen sets of Clifford computational tasks defined by all combinations of the following four binary choices

(i) NONADAPT vs. ADAPT
(ii) IN(BITS) vs. IN(PROD)
(iii) OUT(1) vs. OUT(MANY)
(iv) WEAK vs. STRONG.

The corresponding simulation complexities that we will prove, are summarised in figure 1.

The original Gottesman-Knill theorem [1] asserts that efficient classical simulations exist for the case ADAPT, IN(BITS), OUT(MANY) and WEAK (cf. theorem 5 below). In contrast, we find here that eight of the sixteen cases of extended Clifford circuits are not (likely to be) classically efficiently simulatable.

We draw attention to some examples of extreme changes of simulation complexity resulting from seemingly modest modifications in the defining computational resources. Perhaps the most significant such comparisons for issues of computing power will be cases involving only weak simulations, since implementing the circuit itself as a quantum process yields only one sample of the output probability distribution, in contrast to the far greater information resulting from strong simulation (cf. [6, 14]).

Example 1. For the case of non-adaptive (or equivalently, unitary) circuits with general product state inputs, we have that 1-bit outputs are classically efficiently simulatable in both weak and strong senses. However allowing just many bit outputs results in #P hardness for strong simulation and a more subtle certification of hardness (related to PH collapse) for weak simulation. This indicates a significant increase of computational power associated

to the passage from OUT(1) to OUT(MANY) i.e. merely sampling more lines of the same class of quantum processes (and see also [22] where a similar phenomenon is observed for computational processes defined by commuting quantum circuits).

On the other hand, in the case of adaptive circuits with computational basis inputs the passage from one to many output lines remains classically efficiently simulatable in the weak sense (all other adaptive cases already being QC- or #P-hard).

Note also that in our definitions of classical simulation we ask for simulation only of the output distribution of the computational task, and not of intermediate measurement distributions (if there are any). Indeed inclusion of the latter could elevate an OUT(1) scenario to OUT(MANY) via consideration of intermediate measurement outcomes together with the single bit output of the task, and the associated simulation complexity could radically change. □

Example 2. Another particularly interesting comparison is that of weak simulation for general product state inputs and single bit outputs, with the transition from non-adaptive to adaptive circuits i.e. we compare
Case A: IN(PROD), OUT(1), WEAK, NONADAPT to
Case B: IN(PROD), OUT(1), WEAK, ADAPT.
Case A admits efficient weak classical simulation whereas case B is QC-hard. But now note that the passage from Case A to Case B involves the inclusion of a purely *classical* extra resource viz. classical adaptive choice of gates, without introducing any new gates. Furthermore the class of quantum processes occurring in runs of Case B is *exactly the same* as the class occurring in runs of Case A, since any single actual run of an adaptive circuit can occur as a run of a non-adaptive circuit (that non-adaptively prescribes the sequence of gates that were adaptively chosen). Indeed from an experimentalist's point of view cases A and B may be claimed to be totally indistinguishable in the following sense: suppose an experimentalist $\mathcal{E}$ has the ability to implement basic Clifford gates and measurements. A theorist $\mathcal{T}$ directs $\mathcal{E}$ by announcing one by one, a sequence of basic gates and measurements, which $\mathcal{E}$ successively implements. For each measurement instruction $\mathcal{E}$ announces the measurement outcome before further instructions from $\mathcal{T}$. Then $\mathcal{E}$ cannot tell whether $\mathcal{T}$ is choosing gates adaptively (Case B) or not (Case A) – the demands on $\mathcal{E}$'s laboratory are exactly the same, and case B results in no new quantum processes. Yet Case B can perform universal quantum computation whereas Case A is fully classically efficiently simulatable. □

The sixteen cases of extended Clifford circuits give rise to a rich landscape of simulation complexities. Apart from Cl-P, QC-hardness and #P-hardness, we will see in the course of the proofs that connections to other major complexity classes appear as well. For example (cf. remark below theorem 5), uniform families of adaptive Clifford circuits with computational basis inputs have precisely the same power as a universal randomised classical computation. Thus the class of languages decidable by such Clifford processes in poly-time with bounded error, is precisely BPP. What is more, just changing from computational basis inputs to arbitrary product state inputs (and keeping the other parameters equal) yields universal quantum computation, so in the same poly-time bounded error setting, these Clifford processes now give precisely BQP [8]. We will also find that post-selected non-adaptive Clifford circuits with product state inputs have the same power as BQP with postselection (cf. theorem 7 and [8]), which is known to coincide with the class PP [24]. Finally we recall

that the simulation complexity of non-adaptive Clifford circuits with computational basis inputs and single bit outputs is known to be characterized by the class $\oplus\mathrm{L} \subseteq \mathrm{P}$ [4].

4 Proofs of main results

In this section we give proofs of theorems 1 to 7 that appear as seven of the sixteen cases depicted in figure 1. For the remaining cases it may be easily checked that they all follow from the seven basic cases using the following simple facts:
(i) if a set of tasks is Cl-P then any subset is Cl-P too;
(ii) if a subset of tasks is QC- or #P-hard then the full set is QC- or #P-hard too;
(iii) replacing IN(BITS) by IN(PROD), or replacing OUT(1) by OUT(MANY), increases the set of computational tasks;
(iv) by lemma 1, if strong simulation is Cl-P, then weak simulation is Cl-P too (keeping all other resource choices unchanged).

We will use the following notations relating to bit strings and Pauli operators. For any n-bit strings $\mathbf{a} = a_1 \dots a_n$ and $\mathbf{b} = b_1 \dots b_n$, $\mathbf{c} = \mathbf{a} + \mathbf{b}$ will denote the n-bit string with $c_i = a_i \oplus b_i$ (and $\oplus$ being addition mod 2), and $\mathbf{a} \cdot \mathbf{b} = a_1 b_1 \oplus \dots \oplus a_n b_n$ will denote the mod 2 inner product. We will also use the notation $X(\mathbf{a}) = X^{a_1} \otimes \dots \otimes X^{a_n}$ and $Z(\mathbf{a}) = Z^{a_1} \otimes \dots \otimes Z^{a_n}$. $|\mathbf{a}\rangle$ will denote the computational basis state corresponding to $\mathbf{a}$. Then the following properties are easily verified for any n-bit strings $\mathbf{x}, \mathbf{a}, \mathbf{a}'$:

$$\begin{aligned} &X(\mathbf{a})\,|\mathbf{x}\rangle = |\mathbf{x}+\mathbf{a}\rangle, \quad Z(\mathbf{a})\,|\mathbf{x}\rangle = (-1)^{\mathbf{x}\cdot\mathbf{a}}\,|\mathbf{x}\rangle\,, \\ &X(\mathbf{a})X(\mathbf{a}') = X(\mathbf{a}+\mathbf{a}'), \quad Z(\mathbf{a})Z(\mathbf{a}') = Z(\mathbf{a}+\mathbf{a}') \\ &X(\mathbf{a})Z(\mathbf{a}') = (-1)^{\mathbf{a}\cdot\mathbf{a}'}Z(\mathbf{a}')X(\mathbf{a}). \end{aligned} \tag{3}$$

Since $Y = iXZ$, any Pauli operator P can be written uniquely as $P = \alpha X(\mathbf{a})Z(\mathbf{b})$ for some $\alpha \in \{\pm 1, \pm i\}$ and n-bit strings $\mathbf{a}$ and $\mathbf{b}$. Labelling the α values by 2-bit strings $r_1 r_2$, we call the $(2n+2)$-bit string $(r_1 r_2, \mathbf{a}, \mathbf{b})$ the *label* of P and α the *phase* of P. If G is any basic Clifford gate (acting on specified qubit line(s), and extended by I on all other lines), and P is any n-qubit Pauli operator, then the label of $P' = GPG^\dagger$ can be easily computed from the label of P in $O(n)$ time. In fact only the phase of P and the label entries pertaining to the line(s) of action of the basic Clifford gate are modified.

We begin by proving two elementary simplifications of circuit structures that were mentioned in section 2. To formally establish these we use the following construction: let C be any (adaptive or non-adaptive) circuit on n lines with K intermediate measurements, input state $|\psi\rangle$ and final output measurements on lines $1 \le j_1 \dots < j_l \le n$. Introduce an enlarged unitary circuit C^* on $n+K$ lines defined as follows. For each intermediate measurement M_i on line i of C introduce an extra ancilla qubit (line $n+i$) in state $|0\rangle$ and replace the measurement operation by the unitary Clifford operation $CX_{i,n+i}$ (where $CX_{j,k}$ is the 2-qubit controlled-X operation with source j and target k).

Lemma 2. *Suppose C with input and output as above is a non-adaptive circuit with K intermediate measurements. Then there is a* unitary *circuit C' on $n+K$ lines which is equivalent to C in the following sense: if C' has input $|\psi\rangle\,|0\rangle \dots |0\rangle$ then measurement of lines $j_1, \dots j_l$ of C' will result in the same probability distribution of outputs as C on $|\psi\rangle$.*

Proof. We just take C' to be C^* as defined above. □

Lemma 3. *Suppose C as above is an adaptive circuit. Then there is an adaptive circuit $\tilde{C}$ on $n+K$ lines which is equivalent to C (in the sense given in lemma 2 above) and*
(i) in $\tilde{C}$ after each intermediate measurement $M_i(x_i)$ the line i and its post-measurement state are not further used in any subsequent operations of $\tilde{C}$. Furthermore the choice of line i here is always non-adaptive i.e. independent of previous measurement outcomes.
(ii) In $\tilde{C}$ the set of output lines $\{j_1, \ldots, j_l\}$ is disjoint from the set of intermediate measured lines.

Proof. To construct $\tilde{C}$ we take C^* as above, but after each extra $CX_{i,n+i}$ operation we immediately measure line $n+i$, and use its output as the result of the intermediate measurement M_i of C, for subsequent adaptations. □

We are now ready to prove our seven theorems.

NONADAPT, IN(PROD), OUT(1) and STRONG: Cl-P

Theorem 1. *Let $\mathcal{A}$ be the set of computational tasks defined by non-adaptive Clifford circuits, general product state inputs and single bit outputs. Then $\mathcal{A}$ may be strongly efficiently classically simulated.*

Proof. This result has been proved in [20] and we summarise the argument here. Using lemma 2 we may assume wlog that the Clifford circuit is unitary. Let $C = C_N \ldots C_1$ be a unitary Clifford circuit with product state input $|\alpha\rangle = |\alpha_1\rangle \ldots |\alpha_n\rangle$. Write $|\beta\rangle = C|\alpha\rangle$. We may assume that the output, with probabilities p_0, p_1 is obtained from line 1 (as the swap gate is Clifford). Let $A = Z \otimes I \otimes I \ldots \otimes I$. Then $p_0 - p_1 = \langle\beta| A |\beta\rangle = \langle\alpha| C^\dagger A C |\alpha\rangle$. Now A is a Pauli operator so after successive conjugations by the C_i's we get $C^\dagger A C = \gamma P_1 \otimes \ldots \otimes P_n$ where the label of the latter is easily computed in poly(N) time. Thus $p_0 - p_1 = \gamma \prod_{i=1}^n \langle\alpha_i| P_i |\alpha_i\rangle$ and the latter expression, being a product of n 2×2 matrix expectation values, is readily computable in poly(N) time, providing the efficient strong classical simulation (as $p_0 + p_1 = 1$ too). □

Remark. The simple method of the above proof does not generalise to the case of OUT(MANY) with $O(n)$ output lines. Indeed we will see (cf theorems 4 and 6 below) that this case is #P-hard but remains classically efficiently strongly simulatable if we restrict the input states to just computational basis states i.e. to IN(BITS).

ADAPT, IN(BITS), OUT(1) and STRONG: #P-hard

Theorem 2. *Let $\mathcal{A}$ be the set of computational tasks defined by adaptive Clifford circuits, computational basis state inputs and single bit outputs. Then the strong classical simulation of $\mathcal{A}$ is #P-hard.*

Proof. With the availability of adaptation we are able to apply the gate CX_{jk} or the identity gate I_{jk} (on lines j and k) chosen conditionally according to the result of a measurement on another line i. Thus if these lines are promised to be in computational basis states we can apply the Toffoli gate. (Note however that we cannot by this method apply the Toffoli

gate coherently on general quantum states because the adaptation requires a measurement on line i). Then with the availability of computational basis state inputs, using X and this Toffoli construction, we can efficiently implement universal classical computation. Thus if f is any Boolean function from n bits to one bit, we can implement the transformation $A_f : |\mathbf{x}\rangle |0\rangle \to |\mathbf{x}\rangle |f(\mathbf{x})\rangle$ (so long as the input is a computational basis state). Consider now the following process which is allowed in $\mathcal{A}$: starting with n qubits each in state $|0\rangle$, apply H to each and measure each to generate a uniformly random n-qubit computational basis state $|\mathbf{x}\rangle$. Then apply A_f and finally measure the qubit line of $|f(\mathbf{x})\rangle$ to give a single bit output. Clearly the probability of obtaining 1 is $\#f/2^n$ so strong simulation of this process is #P-hard. □

ADAPT, IN(PROD), OUT(1) and WEAK: QC-hard

Theorem 3. *Let $\mathcal{A}$ be the set of computational tasks defined by adaptive Clifford circuits, general product state inputs and single bit outputs. Then the weak classical simulation of $\mathcal{A}$ is QC-hard.*

Proof. This result is well known, see e.g. [8]. It suffices to show that within the given resource constraints, the phase gate $S = \mathrm{diag}(1, e^{i\pi/4})$ may be implemented on any desired qubit line. This is achieved by introducing an extra ancilla qubit labelled a, in state $\left|\frac{\pi}{4}\right\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$ (respecting availability of product state inputs) and then applying the process of lemma 4 below, and finally applying the Clifford gate $T = S^2$ to line i conditionally on the value of the ancilla measurement outcome (which is possible since adaptation is available). □

Lemma 4. *Let $|\psi\rangle_{1\ldots n}$ be an n-qubit state on lines 1 to n and let $S = \mathrm{diag}(1, e^{i\pi/4})$. Let $\left|\frac{\pi}{4}\right\rangle_a = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$ be an extra ancillary qubit. Then*

$$M_a(x)\, CX_{ai}\, |\psi\rangle \left|\tfrac{\pi}{4}\right\rangle_a \text{ results in } \begin{cases} S_i |\psi\rangle |0\rangle_a & \text{if } x = 0 \\ e^{i\pi/4} S_i^{-1} |\psi\rangle |1\rangle_a & \text{if } x = 1 \end{cases}$$

where S_i denotes the application of S to qubit i, and CX_{ai} is the application of CX to lines a and i with i as target line.

Proof of lemma. A straightforward calculation. □

NONADAPT, IN(BITS), OUT(MANY) and STRONG: Cl-P

Theorem 4. *Let $\mathcal{A}$ be the set of computational tasks defined by non-adaptive Clifford circuits, computational basis state inputs and multiple bit outputs. Then $\mathcal{A}$ may be strongly efficiently classically simulated.*

Proof. The techniques of [3] and alternatively [6] may be used to prove theorem 4. Here we give a proof using a third method. Let C be a non-adaptive Clifford circuit with computational basis input $|x_1 \ldots x_n\rangle$ and let $j_1, \ldots, j_m$ be any subset of the output lines. We will show that the corresponding marginal probability $p(y_1, \ldots, y_m)$ may be efficiently classically computed. We may assume the following standardised situation:

(i) C is unitary (by lemma 2);
(ii) $x_1 \dots x_n = 00\dots0$ and $y_1 \dots y_m = 00\dots0$ (since we can pre- and post- include extra X gates on lines where x_i or y_j are 1);
(iii) $j_1, \dots, j_m = 1, \dots, m$ for $m \leq n$ (since swap gates are Clifford operations).
Thus for C unitary with input $|0^n\rangle = |00\dots0\rangle$ let $p = \text{prob}(0\dots0)$ be the probability of obtaining 0 from measurement of each of the lines 1 to m. Using $|0\rangle\langle 0| = (I+Z)/2$ and writing $\mathbf{t} = t_1 \dots t_m$ for m-bit strings we have

$$\begin{aligned} p &= \tfrac{1}{2^m} \langle 0^n| C^\dagger \, (I_1 + Z_1) \otimes \dots \otimes (I_m + Z_m) \otimes I_{m+1} \otimes \dots \otimes I_n \, C \, |0^n\rangle \\ &= \tfrac{1}{2^m} \textstyle\sum_{\mathbf{t} \in \mathbb{Z}_2^m} \langle 0^n| C^\dagger \, \tilde{Z}(\mathbf{t}) \, C \, |0^n\rangle \end{aligned}$$

where $\tilde{Z}(\mathbf{t})$ is the n-qubit Pauli operator $Z(\mathbf{t}) \otimes I \dots \otimes I$ obtained by extending the m-qubit operator $Z(\mathbf{t})$ by $(n-m)$ I's. This is a sum with potentially exponentially many terms (e.g. if $m = O(n)$) yet it can be evaluated in poly(n) time as follows. Using the Clifford conjugation relations we have

$$\Gamma(\mathbf{t}) \equiv C^\dagger \, \tilde{Z}(\mathbf{t}) \, C = \sigma(\mathbf{t}) X(\mathbf{a}(\mathbf{t})) Z(\mathbf{b}(\mathbf{t})) \tag{4}$$

with $\sigma(\mathbf{t}) \in \{\pm 1, \pm i\}$ and $\mathbf{a}(\mathbf{t}), \mathbf{b}(\mathbf{t}) \in \mathbb{Z}_2^n$. Furthermore, for each $\mathbf{t}$ these labels can be computed efficiently.

Next, introduce basis vectors $\mathbf{e}_j = 0\dots010\dots0$ in $\mathbb{Z}_2^m$ (having 1 in the j^{th} slot) for $j = 1, \dots, m$. Then since $\mathbf{t} = \sum_i t_i \mathbf{e}_i$ and $\tilde{Z}(\mathbf{t}) = \tilde{Z}(\mathbf{e}_1)^{t_1} \dots \tilde{Z}(\mathbf{e}_m)^{t_m}$ we have

$$\mathbf{a}(\mathbf{t}) = \sum_{i=1}^{m} t_i \, \mathbf{a}(\mathbf{e}_i). \tag{5}$$

Next note that since $\langle 0| X |0\rangle = 0$ we have

$$\langle 0^n| \Gamma(\mathbf{t}) |0^n\rangle \neq 0 \quad \text{iff} \quad \mathbf{a}(\mathbf{t}) = 0^n.$$

Furthermore if $\mathbf{a}(\mathbf{t}) = 0^n$ then

$$\Gamma(\mathbf{t}) = \sigma(\mathbf{t}) \, Z(\mathbf{b}(\mathbf{t})) \tag{6}$$

and since $\Gamma(\mathbf{t})^2 = I = Z(\mathbf{b}(\mathbf{t}))^2$ we must have $\sigma(\mathbf{t}) \in \{\pm 1\}$, so that

$$\sigma(\mathbf{t}) = (-1)^{u(\mathbf{t})} \tag{7}$$

with $u(\mathbf{t}) \in \{0, 1\}$. Furthermore, using $\langle 0| Z |0\rangle = \langle 0| I |0\rangle = 1$ we get

$$\langle 0^n| \Gamma(\mathbf{t}) |0^n\rangle = \sigma(\mathbf{t}) \quad \text{if} \quad \mathbf{a}(\mathbf{t}) = 0^n.$$

Introducing $T_0 = \{\mathbf{t} : \mathbf{a}(\mathbf{t}) = 0^n\} \subseteq \mathbb{Z}_2^m$ we thus get

$$p = \frac{1}{2^m} \sum_{\mathbf{t} \in T_0} (-1)^{u(\mathbf{t})}.$$

Next we characterise T_0. We have $\mathbf{t} \in T_0$ iff $\mathbf{a}(\mathbf{t}) = 0^n$ so by eq. (5), T_0 is the subspace of $\mathbb{Z}_2^m$ given by the solution space of $A\mathbf{t} = 0$ where A is the $n \times m$ sized matrix with $\mathbf{a}(\mathbf{e}_i)$ for $i = 1, \dots m$ as the columns. Using the label update rules for Clifford conjugations,

all $\mathbf{a}(\mathbf{e}_i)$'s can be computed in poly(n) time. Thus we can compute a basis $\{\mathbf{c}_1, \ldots, \mathbf{c}_l\}$ of T_0 (and hence also the information of its dimension l) in poly(n) time. Then $\mathbf{t} \in T_0$ iff $\mathbf{t} = \sum_{i=1}^{l} s_i \mathbf{c}_i$ for $\mathbf{s} = s_1 \ldots s_l \in \mathbb{Z}_2^l$ and

$$p = \frac{1}{2^m} \sum_{\mathbf{s} \in \mathbb{Z}_2^l} (-1)^{u(\sum s_i \mathbf{c}_i)}.$$

Finally recalling that $Z(\mathbf{t}+\mathbf{t}') = Z(\mathbf{t})Z(\mathbf{t}')$ we see from eqs (6) and (7) that $u(\mathbf{t})$ is a linear function of $\mathbf{t}$, so writing $u(\mathbf{c}_i) = k_i$ and $\mathbf{k} = k_1 \ldots k_l$ we have

$$p = \frac{1}{2^m} \sum_{\mathbf{s} \in \mathbb{Z}_2^l} (-1)^{\mathbf{k} \cdot \mathbf{s}}.$$

Now $g(\mathbf{s}) = (-1)^{\mathbf{k} \cdot \mathbf{s}}$ is a balanced function for $\mathbf{k} \neq 0^l$ (i.e. taking values ± 1 equally often) so

$$p = \begin{cases} 2^l/2^m & \text{if} \quad \mathbf{k} = 0^l \\ 0 & \text{if} \quad \mathbf{k} \neq 0^l \end{cases}$$

concluding our efficient classical computation of p.
To summarise: given the description of the circuit C we first compute $\mathbf{a}(\mathbf{e}_i)$ for $i = 1, \ldots m$ (from the Clifford conjugation relations) giving the matrix A via columns. Then we compute any basis $\{\mathbf{c}_1, \ldots, \mathbf{c}_l\}$ of $\ker(A)$, and compute the l-bit string $\mathbf{k} = u(\mathbf{c}_1) \ldots u(\mathbf{c}_l)$ (again from the Clifford conjugation relations in eq. (4) with $\mathbf{t} = \mathbf{c}_i$ there). Then $p = 2^{l-m}$ if $\mathbf{k} = 0^l$ and $p = 0$ otherwise. □

ADAPT, IN(BITS), OUT(MANY) and WEAK: Cl-P

Theorem 5. *Let $\mathcal{A}$ be the set of computational tasks defined by adaptive Clifford circuits, computational basis state inputs and multiple bit outputs. Then $\mathcal{A}$ may be weakly efficiently classically simulated.*

Remark. Note that by theorem 2 strong simulation in this scenario even with *single* bit outputs, is #P-hard. The weak simulation that we give in the proof of theorem 5 below will use the strong simulation result of theorem 4. A different proof of theorem 5 may be given in terms of the stabiliser formalism (see [2], especially the Gottesman Knill theorem 10.7 therein) which develops a description of the evolving state through the course of the computation.

Remark. A family of Clifford computational tasks $\{T_n : n = 1, 2, \ldots\}$, where T_n acts on n qubits, is said to be uniform if the description of T_n can be computed in poly(n) time by a (deterministic) classical Turing machine on input of n. Theorem 5 shows that uniform families of adaptive Clifford circuits with computational basis state inputs and multiple bit outputs do not have additional power over polynomial-time randomised classical computation. Interestingly, the power of such uniform families of Clifford computational tasks in fact *coincides* with polynomial-time randomised classical computation. This follows from the constructions in the proof of theorem 2 where it was shown how to generate random bits and realize Toffoli gates with adaptive Clifford circuits acting on computational basis state inputs (see also [25] for related insights on realizing universal classical computation

with adaptive stabilizer measurements). Finally we note the interesting comparison with the case of product states (replacing computational basis states) as inputs (keeping all other parameters the same) where the associated uniform families of Clifford computational tasks have precisely the same power as universal *quantum* computation (which similarly immediately follows from the proof of theorem 3).

Proof. Let C be an adaptive circuit on n qubit lines with K intermediate measurements, input $|\mathbf{x}\rangle = |x_1\rangle \dots |x_n\rangle$ and l output lines. By lemma 3 we may wlog instead work with an extended circuit $\tilde{C}$ on $n+K$ lines having the following form (rearranging the order of lines in lemma 3): $\tilde{C}$ has input $|0\rangle_1 \dots |0\rangle_K |x_1\rangle_{K+1} \dots |x_n\rangle_{K+n}$ and the output measurements are on lines $K+1, \dots, K+l$ (wlog, as swap operations are Clifford). Furthermore the i^{th} intermediate measurement yielding outcome y_i for $1 \leq i \leq K$ is on line i and then line i is not further used in $\tilde{C}$. As such, these measurements can be viewed as outputs too with the caveat that subsequent choices of gates may depend on the values $y_1, y_2, \dots, y_K$ as they sequentially emerge. A full run of $\tilde{C}$ (including its l output measurements) samples an associated probability distribution $p(y_1, \dots, y_K, y_{K+1}, \dots, y_{K+l})$.

Now if $y_1, \dots, y_j$ for $j \leq K$ are specified then the circuit up to the j^{th} measurement becomes non-adaptive (i.e. the adaptive choices have been specified) and hence we can efficiently compute the marginal $p(y_1, \dots, y_j)$ by theorem 4. Similarly if $j \geq K+1$ all adaptations have been specified and by theorem 4 we can again efficiently compute the corresponding marginals $p(y_1, \dots, y_j)$. Hence by lemma 1 we can efficiently sample the distribution $p(y_1, \dots, y_K, y_{K+1}, \dots, y_{K+l})$ and the last l bits of the sample provides a weak efficient classical simulation of $\tilde{C}$ and hence of C too. □

NONADAPT, IN(PROD), OUT(MANY) and STRONG: #P-hard

Theorem 6. *Let $\mathcal{A}$ be the set of computational tasks defined by non-adaptive Clifford circuits, general product state inputs and multiple bit outputs. Then the strong classical simulation of $\mathcal{A}$ is #P-hard.*

Remark. Note that by theorem 1 the same scenario with just 1-bit outputs is classically strongly efficiently simulatable.

Proof. We will show that efficient strong simulation of $\mathcal{A}$ would imply efficient strong simulation of universal quantum computation and hence provide an efficient solution of the #SAT problem (using the process described in the proof of theorem 2 to express $\#f$ for any Boolean f as a probability value).

Thus let D be any quantum circuit comprising basic Clifford gates and S gates with a product state input, and single bit output denoted y. Consider again the process of lemma 4. In our present scenario for $\mathcal{A}$ we do not have adaptation available so we cannot implement S gates as we did in the proof of theorem 3. Instead we proceed as follows. Suppose there are K S gates in D. For each such gate introduce an ancilla in state $\left|\frac{\pi}{4}\right\rangle$ and replace the S gate by the sequence of operations in lemma 4, resulting in a non-adaptive circuit D' now involving only basic Clifford gates. Then D' has $K+1$ outputs viz. y and measurements of the K ancilla lines denoted $a_1, \dots, a_K$, and we have

$$\begin{aligned} \text{Prob}_D(y) &= \text{Prob}_{D'}(y \,|\, 0_{a_1} \dots 0_{a_K}) \\ &= \text{Prob}_{D'}(y\, 0_{a_1} \dots 0_{a_K}) / \text{Prob}_{D'}(0_{a_1} \dots 0_{a_K}). \end{aligned}$$

Strong classical efficient simulation of $\mathcal{A}$ (which allows multi-line outputs) implies that we can compute both of the D' probabilities in the above quotient and hence $\text{Prob}_D(y)$ i.e. we then get a strong efficient simulation of D. $\square$

NONADAPT, IN(PROD), OUT(MANY) and WEAK: collapse of PH

Theorem 7. *Let $\mathcal{A}$ be the set of computational tasks defined (as in theorem 6) by non-adaptive Clifford circuits, general product state inputs and multiple bit outputs. If $\mathcal{A}$ could be weakly efficiently classically simulated, then the polynomial hierarchy PH would collapse to its third level.*

Remark. For the definition of PH we refer to [23]. The proof of theorem 7 below rests on techniques introduced in [22] and below we will be content to describe the relationship of the class $\mathcal{A}$ in theorem 7 to the constructions of [22] and refer to the latter for further details of the proof.

Remark. Theorem 7 provides a partial answer to an open problem raised in [4] viz. the question of the computational power of non-adaptive Clifford circuits with product state inputs and multiple bit outputs.

Proof. Consider again the process of lemma 4. Now instead of utilising adaptation (as we did in the proof of theorem 3) or conditional probabilities (as we did above in theorem 6), we could alternatively implement S using the process of lemma 4 if we were able to *post-select* on measurement outcomes viz. we post-select the value 0 of the ancilla measurement. It follows that our class $\mathcal{A}$ *together with post-selection* contains universal quantum computation, and even more, universal quantum computation with post-selection. Aaronson [24] has shown that the class BQP with post-selection coincides with the classical class PP (cf [23] for definitions). Thus our class $\mathcal{A}$ with post-selection contains PP.

Now let $\mathcal{K}$ be any class of bounded error quantum circuits such that $\mathcal{K}$ with post-selection contains PP. Then (as elaborated in [22]) weak efficient classical simulation of $\mathcal{K}$ for output measurements on many lines, implies that $\mathcal{K}$ with post-selection is contained in BPP with post-selection [22]. Then according to a result of classical complexity theory (cf [22] for details), the latter inclusion (implying that PP is contained in BPP with post-selection) implies that PH collapses to its third level. Hence weak efficient classical simulation of our class $\mathcal{A}$ would imply this collapse. $\square$

Acknowledgments

RJ was supported in part by the EC networks Q-ESSENCE and QCS. Preliminary versions of this work were presented at the conferences AQIS12 (Suzhou China, September 2012) and QANSAS11 (Agra India, December 2011).

References

[1] Gottesman, D. 1999 The Heisenberg representation of quantum computers. In Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics, pp. 32-43. International Press. arXiv:quant-ph/9807006v1.

[2] Nielsen, M. and Chuang, I. 2000 Quantum Computation and Quantum Information. Cambridge University Press.

[3] Dehaene, J. and De Moor, B. 2003 The Clifford group, stabilizer states, and linear and quadratic operations over GF(2). Phys. Rev. A 68, 042318. arXiv:quant-ph/0304125.

[4] Aaronson S. and Gottesman, D. 2004 Improved simulation of stabilizer circuits. Phys. Rev. A 70, 052328. arXiv:quant-ph/0406196.

[5] Anders, S. and Briegel, H. J. 2006 Fast simulation of stabilizer circuits using a graph-state representation. Phys. Rev. A 73, 022334. arXiv:quant-ph/0504117.

[6] Van den Nest, M. 2010 Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond. Quant. Inf. Comp. 10 (3-4), pp. 0258-0271. arXiv:0811.0898.

[7] Gottesman, D. 1997 Stabilizer Codes and Quantum Error Correction. PhD thesis, California Institute of Technology, Pasadena, CA. arXiv:quant-ph/9705052.

[8] Bravyi, S. and Kitaev, A. 2005 Universal quantum computation with ideal Clifford gates and noisy ancillas. Phys. Rev. A 71, 022316. arXiv:quant-ph/0403025.

[9] Raussendorf, R. and Briegel, H. J. 2001 A one-way quantum computer. Phys. Rev. Lett. 86, 51885191. arXiv:quant-ph/0010033.

[10] Raussendorf, R., Browne, D. E. and Briegel, H. J. 2003 Measurement-based quantum computation with cluster states. Phys. Rev. A 68, 022312. arXiv:quant-ph/0301052.

[11] Nebe, G., Rains, E. M. and Sloane, N. J. A. 2000 The invariants of the Clifford groups. arXiv:math/0001038.

[12] Nebe, G., Rains, E. M. and Sloane, N. J. A. 2006 Self-Dual Codes and Invariant Theory (Springer: Berlin).

[13] Jozsa, R. and Miyake, A. 2008 Matchgates and classical simulation of quantum circuits. Proc. Roy. Soc. (Lond) A464, p3089-3106. arXiv:0804.4050.

[14] Van den Nest, M. 2011 Simulating quantum computers with probabilistic methods. Quant. Inf. Comp. 11 (9-10), 784-812. arXiv:0911.1624.

[15] Gottesman, D. 1999 Fault-Tolerant Quantum Computation with Higher-Dimensional Systems, Chaos Solitons Fractals 10:1749-1758. arXiv:quant-ph/9802007.

[16] Hostens, E., Dehaene J. and De Moor, B. 2005 Stabilizer states and Clifford operations for systems of arbitrary dimensions, and modular arithmetic. Phys. Rev. A 71, 042315. arXiv:quant-ph/0408190.

[17] de Beaudrap, N. 2013 A linearized stabilizer formalism for systems of finite dimension. Quant. Inf. Comp. 13, 73–115. arXiv:1102.3354.

[18] Van den Nest, M. 2012 Efficient classical simulations of quantum Fourier transforms and normalizer circuits over Abelian groups. arXiv:1201.4867.

[19] Bermejo-Vega, J. and Van den Nest, M. 2012 A Gottesman-Knill theorem for all finite Abelian groups. arXiv:1210.3637.

[20] Clark, S., Jozsa, R. and Linden, N. 2008 Generalised Clifford groups and simulation of associated quantum circuits. Quant. Inf. Comp. 8, 106-126. arXiv:quant-ph/0701103.

[21] Terhal, B. and DiVincenzo, D. 2004 Adaptive quantum computation, constant depth quantum circuits and arthur-merlin games. Quant. Inf. Comp. 4, 134–145. arXiv:quant-ph/0205133.

[22] Bremner, M., Jozsa, R. and Shepherd, D. 2011 Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. Proc. R. Soc. A 467, 459–472. arXiv:1005:1407.

[23] Arora, S. and Barak, B. 2009 Computational Complexity – a modern approach, Cambridge University Press.

[24] Aaronson, S. 2005 Quantum computing, post-selection and probabilistic polynomial time. Proc. R. Soc. A 461, 3473–3483. arXiv:quant-ph/0412.187.

[25] Anders, J. and Browne, D. E. 2009 Computational Power of Correlations. Phys. Rev. Lett. 102, 050502. arXiv:0805.1002.