Exact quantum query complexity

Ashley Montanaro, Richard Jozsa and Graeme Mitchison

Centre for Quantum Information and Foundations,
Department of Applied Mathematics and Theoretical Physics,
University of Cambridge

arXiv:1111.0475

Engineering and Physical Sciences Research Council

Query complexity

- Many important quantum algorithms operate in the query complexity model.

- In this model, we are given access to a hidden bit-string $x \in \{0,1\}^n$ via a black box which returns x_i when i is passed in.

- To implement this on a quantum computer, we imagine we have access to a unitary oracle which maps $|i\rangle|y\rangle \mapsto |i\rangle|y \oplus x_i\rangle$.

- We want to compute some (known) function $f(x)$ using the minimum worst-case number of queries.

Query complexity

- Define $D(f)$ ($Q_E(f)$) as the minimum number of classical (quantum) queries required to compute f with certainty.

- Similarly, $R_2(f)$ ($Q_2(f)$) is the minimum number of classical (quantum) queries required to compute f with worst-case success probability 2/3.

- Of course, $Q_2(f) \leqslant Q_E(f) \leqslant D(f)$ and $Q_2(f) \leqslant R_2(f) \leqslant D(f)$.

Query complexity separations

Many separations are known between quantum and classical query complexity.

- The Deutsch-Jozsa algorithm shows the existence of a partial function f (i.e. with a promise on the input) such that $Q_E(f) = O(1)$, but $D(f) = \Omega(n)$.

Query complexity separations

Many separations are known between quantum and classical query complexity.

- The Deutsch-Jozsa algorithm shows the existence of a partial function f (i.e. with a promise on the input) such that $Q_E(f) = O(1)$, but $D(f) = \Omega(n)$.
- In fact, it is known that if f is a partial function we can have $Q_E(f)$ exponentially smaller than even $R_2(f)$ [Brassard and Høyer '97].

Query complexity separations

Many separations are known between quantum and classical query complexity.

- The Deutsch-Jozsa algorithm shows the existence of a partial function f (i.e. with a promise on the input) such that $Q_E(f) = O(1)$, but $D(f) = \Omega(n)$.
- In fact, it is known that if f is a partial function we can have $Q_E(f)$ exponentially smaller than even $R_2(f)$ [Brassard and Høyer '97].
- If f is a total function (i.e. no promise on the input), we can have $Q_2(f) = O(\sqrt{R_2(f)})$ by Grover's algorithm.

Query complexity separations

Many separations are known between quantum and classical query complexity.

- The Deutsch-Jozsa algorithm shows the existence of a partial function f (i.e. with a promise on the input) such that $Q_E(f) = O(1)$, but $D(f) = \Omega(n)$.
- In fact, it is known that if f is a partial function we can have $Q_E(f)$ exponentially smaller than even $R_2(f)$ [Brassard and Høyer '97].
- If f is a total function (i.e. no promise on the input), we can have $Q_2(f) = O(\sqrt{R_2(f)})$ by Grover's algorithm.
- On the other hand, for all total functions f, $R_2(f) = O(Q_2(f)^6)$ [Beals et al '97].

So bounded-error quantum query complexity of total functions is fairly well understood.

What about exact quantum query complexity?

- It was shown by [Midrijanis '04] that for total functions f, $D(f) = O(Q_E(f)^3)$.

What about exact quantum query complexity?

- It was shown by [Midrijanis '04] that for total functions f, $D(f) = O(Q_E(f)^3)$.

- On the other hand, exact quantum algorithms can indeed be better than classical algorithms: [Cleve et al '98] showed that the parity of n bits,

$$f(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

can be computed exactly using only $\lceil n/2 \rceil$ quantum queries.

What about exact quantum query complexity?

- It was shown by [Midrijanis '04] that for total functions f, $D(f) = O(Q_E(f)^3)$.

- On the other hand, exact quantum algorithms can indeed be better than classical algorithms: [Cleve et al '98] showed that the parity of n bits,

$$f(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

  can be computed exactly using only $\lceil n/2 \rceil$ quantum queries.

- The algorithm is based on simply computing the parity of 2 bits using 1 quantum query.

Computing the parity of 2 bits in 1 query

1. Create the state $\frac{1}{2}(|1\rangle + |2\rangle)(|0\rangle - |1\rangle)$.

Computing the parity of 2 bits in 1 query

1. Create the state $\frac{1}{2}\left(|1\rangle + |2\rangle\right)\left(|0\rangle - |1\rangle\right)$.
2. Query the oracle to produce

$$\frac{1}{2}\left((-1)^{x_1}|1\rangle + (-1)^{x_2}|2\rangle\right)\left(|0\rangle - |1\rangle\right).$$

Computing the parity of 2 bits in 1 query

1. Create the state $\frac{1}{2}\left(|1\rangle + |2\rangle\right)\left(|0\rangle - |1\rangle\right)$.
2. Query the oracle to produce

$$\frac{1}{2}\left((-1)^{x_1}|1\rangle + (-1)^{x_2}|2\rangle\right)\left(|0\rangle - |1\rangle\right).$$

3. Perform a Hadamard on the first qubit to produce the state

$$\frac{1}{2}\left(((-1)^{x_1} + (-1)^{x_2})|1\rangle + ((-1)^{x_1} - (-1)^{x_2})|2\rangle\right).$$

Computing the parity of 2 bits in 1 query

1. Create the state $\frac{1}{2}\left(|1\rangle + |2\rangle\right)\left(|0\rangle - |1\rangle\right)$.
2. Query the oracle to produce

$$\frac{1}{2}\left((-1)^{x_1}|1\rangle + (-1)^{x_2}|2\rangle\right)\left(|0\rangle - |1\rangle\right).$$

3. Perform a Hadamard on the first qubit to produce the state

$$\frac{1}{2}\left(((-1)^{x_1} + (-1)^{x_2})|1\rangle + ((-1)^{x_1} - (-1)^{x_2})|2\rangle\right).$$

4. Measure the first qubit and output 0 if the outcome was 1, and 1 if the outcome was 2.

Observe that this algorithm is nonadaptive.

Other exact quantum query algorithms for total functions

Other exact quantum query algorithms for total functions

- Er...

Other exact quantum query algorithms for total functions

- Er...
- In fact, to my knowledge there are no other (non-trivial) exact quantum query algorithms for total functions known!

Other exact quantum query algorithms for total functions

- Er...
- In fact, to my knowledge there are no other (non-trivial) exact quantum query algorithms for total functions known!
- However, some authors have used the algorithm for parity as a subroutine, e.g. [Hayes et al '02] use it to compute the majority function using $n - O(\log n)$ queries.

Other exact quantum query algorithms for total functions

- Er...
- In fact, to my knowledge there are no other (non-trivial) exact quantum query algorithms for total functions known!
- However, some authors have used the algorithm for parity as a subroutine, e.g. [Hayes et al '02] use it to compute the majority function using $n - O(\log n)$ queries.
- But it has been open for 14+ years whether there exists a total function f such that $Q_E(f) < D(f)/2$.

Other exact quantum query algorithms for total functions

- Er...
- In fact, to my knowledge there are no other (non-trivial) exact quantum query algorithms for total functions known!
- However, some authors have used the algorithm for parity as a subroutine, e.g. [Hayes et al '02] use it to compute the majority function using $n - O(\log n)$ queries.
- But it has been open for 14+ years whether there exists a total function f such that $Q_E(f) < D(f)/2$.
- Could computing parities be all that exact quantum query algorithms for total functions can do?

Our results

We show that exact quantum query complexity is richer than just computing parities.

Our results

We show that exact quantum query complexity is richer than just computing parities.

- We present some new examples of total boolean functions f such that $Q_E(f)$ is a constant multiple of $D(f)$ (between $1/2$ and $2/3$). We show that these separations cannot be obtained by just computing parities of pairs of bits.

Our results

We show that exact quantum query complexity is richer than just computing parities.

- We present some new examples of total boolean functions f such that $Q_E(f)$ is a constant multiple of $D(f)$ (between $1/2$ and $2/3$). We show that these separations cannot be obtained by just computing parities of pairs of bits.
- These separations are based on concatenating small separations found for functions on small numbers of bits.

Our results

We show that exact quantum query complexity is richer than just computing parities.

- We present some new examples of total boolean functions f such that $Q_E(f)$ is a constant multiple of $D(f)$ (between $1/2$ and $2/3$). We show that these separations cannot be obtained by just computing parities of pairs of bits.
- These separations are based on concatenating small separations found for functions on small numbers of bits.
- For example, we have an exact quantum algorithm which uses 2 queries to compute the EXACT_2 function on 4 bits:

$$\mathrm{EXACT}_2(x) = 1 \text{ if } |x| = 2, \; \mathrm{EXACT}_2(x) = 0 \text{ otherwise.}$$

Our results

We show that exact quantum query complexity is richer than just computing parities.

- We present some new examples of total boolean functions f such that $Q_E(f)$ is a constant multiple of $D(f)$ (between $1/2$ and $2/3$). We show that these separations cannot be obtained by just computing parities of pairs of bits.
- These separations are based on concatenating small separations found for functions on small numbers of bits.
- For example, we have an exact quantum algorithm which uses 2 queries to compute the EXACT_2 function on 4 bits:

$$\mathrm{EXACT}_2(x) = 1 \text{ if } |x| = 2,\ \mathrm{EXACT}_2(x) = 0 \text{ otherwise.}$$

- In fact, we give optimal exact quantum query algorithms for every boolean function $f : \{0,1\}^3 \to \{0,1\}$.

Our results

We show that exact quantum query complexity is richer than just computing parities.

- We present some new examples of total boolean functions f such that $Q_E(f)$ is a constant multiple of $D(f)$ (between $1/2$ and $2/3$). We show that these separations cannot be obtained by just computing parities of pairs of bits.
- These separations are based on concatenating small separations found for functions on small numbers of bits.
- For example, we have an exact quantum algorithm which uses 2 queries to compute the EXACT_2 function on 4 bits:

$$\mathrm{EXACT}_2(x) = 1 \text{ if } |x| = 2, \;\; \mathrm{EXACT}_2(x) = 0 \text{ otherwise.}$$

- In fact, we give optimal exact quantum query algorithms for every boolean function $f : \{0,1\}^3 \to \{0,1\}$.
- We characterise the model of nonadaptive quantum query complexity in terms of a coding-theoretic quantity.

Numerical results

- Our analytical results were inspired by numerical results where we numerically evaluated the best possible success probability of quantum algorithms for all boolean functions on up to 4 bits (and all symmetric boolean function on up to 6 bits).

Numerical results

- Our analytical results were inspired by numerical results where we numerically evaluated the best possible success probability of quantum algorithms for all boolean functions on up to 4 bits (and all symmetric boolean function on up to 6 bits).
- This can be done using a semidefinite programming (SDP) formulation of quantum query complexity due to [Barnum, Saks and Szegedy '03].

Numerical results

- Our analytical results were inspired by numerical results where we numerically evaluated the best possible success probability of quantum algorithms for all boolean functions on up to 4 bits (and all symmetric boolean function on up to 6 bits).

- This can be done using a semidefinite programming (SDP) formulation of quantum query complexity due to [Barnum, Saks and Szegedy '03].

- Given a solution to the SDP, one can write down a quantum query algorithm achieving the same parameters.

Numerical results

- Our analytical results were inspired by numerical results where we numerically evaluated the best possible success probability of quantum algorithms for all boolean functions on up to 4 bits (and all symmetric boolean function on up to 6 bits).

- This can be done using a semidefinite programming (SDP) formulation of quantum query complexity due to [Barnum, Saks and Szegedy '03].

- Given a solution to the SDP, one can write down a quantum query algorithm achieving the same parameters.

- If the SDP gives a result which is close to exact, one can hope to write down an exact quantum algorithm.

Quantum query complexity SDP [BSS '03]

Given $f : \{0,1\}^n \to \{0,1\}$ and $t \in \mathbb{N}$, find a sequence of 2^n-dim real symmetric matrices $(M_i^{(j)})$, where $0 \leqslant i \leqslant n$ and $0 \leqslant j \leqslant t-1$, and 2^n-dim real symmetric matrices Γ_0, Γ_1, such that

$$
\begin{aligned}
\sum_{i=0}^{n} M_i^{(0)} &= E_0 \\
\sum_{i=0}^{n} M_i^{(j)} &= \sum_{i=0}^{n} E_i \circ M_i^{(j-1)} \text{ (for } 1 \leqslant j \leqslant t-1) \\
\Gamma_0 + \Gamma_1 &= \sum_{i=0}^{n} E_i \circ M_i^{(t-1)} \\
F_0 \circ \Gamma_0 &\geqslant (1-\epsilon) F_0, \quad F_1 \circ \Gamma_1 \geqslant (1-\epsilon) F_1.
\end{aligned}
$$

Here E_i is the matrix $\langle x|E_i|y\rangle = (-1)^{x_i + y_i}$, F_0 and F_1 are diagonal 0/1 matrices where $\langle x|F_z|x\rangle = 1$ if and only if $f(x) = z$, and $\circ$ is the Hadamard (entrywise) product of matrices.

Quantum query complexity SDP

Theorem [Barnum, Saks and Szegedy '03]

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 above SDP is feasible.

Further, given a solution to the above SDP, one can write down an explicit quantum algorithm achieving the same parameters.

The explicit algorithm (sketch)

- Any quantum query algorithm consists of a sequence of oracle calls interspersed with arbitrary unitary operators (which do not depend on the input).

The explicit algorithm (sketch)

- Any quantum query algorithm consists of a sequence of oracle calls interspersed with arbitrary unitary operators (which do not depend on the input).
- Divide the Hilbert space on which the quantum query algorithm operates into two registers (input and workspace).

The explicit algorithm (sketch)

- Any quantum query algorithm consists of a sequence of oracle calls interspersed with arbitrary unitary operators (which do not depend on the input).
- Divide the Hilbert space on which the quantum query algorithm operates into two registers (input and workspace).
- Define the state of the algorithm on input x at time j (i.e. just before the $(j+1)$'st query is made) to be

$$|\psi_x^{(j)}\rangle = \sum_{i=0}^{n} |i\rangle|\psi_{x,i}^{(j)}\rangle,$$

where

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

The explicit algorithm (sketch)

- Let O_x be the oracle operator $O_x|i\rangle = (-1)^{x_i}|i\rangle$, and set $O_x|0\rangle = |0\rangle$.

- If the $M_i^{(j)}$ matrices form a solution to the SDP, this implies there exists a unitary operator U_j such that $U_j O_x|\psi_x^{(j-1)}\rangle = |\psi_x^{(j)}\rangle$. Further, U_j can be found explicitly using the polar decomposition.

- Similarly, the constraints on Γ_0, Γ_1 can be used to show that there exists a U_t such that $U_t|\psi_x^{(t)}\rangle = |\gamma_x\rangle$ for all x, where $|\gamma_x\rangle$ is a state which can be measured to determine whether $f(x) = 0$ with success probability $\geqslant 1 - \epsilon$.

Solving the BSS SDP numerically

We used the CVX package for Matlab to solve this SDP. For example, we get the following results for all boolean functions on 3 bits (up to isomorphism):

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

Solving the BSS SDP numerically

We used the CVX package for Matlab to solve this SDP. For example, we get the following results for all boolean functions on 3 bits (up to isomorphism):

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

Highlighted functions display a separation $Q_E(f) < D(f)$.

Solving the BSS SDP numerically

We used the CVX package for Matlab to solve this SDP. For example, we get the following results for all boolean functions on 3 bits (up to isomorphism):

ID	Function	1 query	2 queries
1	$x_1 \wedge x_2 \wedge x_3$	0.800	0.980
6	$x_1 \wedge (x_2 \oplus x_3)$	0.667	1
7	$x_1 \wedge (x_2 \vee x_3)$	0.773	1
22	EXACT_2	0.571	1
23	MAJ	0.667	1
30	$x_1 \oplus (x_2 \vee x_3)$	0.667	1
53	$\mathrm{SEL}(x_1, x_2, x_3)$	0.854	1
67	$(x_1 \wedge x_2) \vee (\bar{x}_1 \wedge \bar{x}_2 \wedge x_3)$	0.773	1
105	PARITY	0.500	1
126	NAE	0.900	1

Red functions: optimal q. algm is based on computing parities.

EXACT_2

We now give a simple and explicit exact quantum algorithm for the EXACT_2 function on 4 bits.

- Again let O_x be the oracle operator $O_x|i\rangle = (-1)^{x_i}|i\rangle$, with $O_x|0\rangle = |0\rangle$.

- 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.

EXACT_2 algorithm

1. Create the state
$$|\psi\rangle = \frac{1}{2}\sum_{i=1}^{4}|i\rangle.$$

EXACT_2 algorithm

1. Create the state
$$|\psi\rangle = \frac{1}{2}\sum_{i=1}^{4} |i\rangle.$$
2. Apply O_x, then U, then O_x again.

EXACT_2 algorithm

1. Create the state
$$|\psi\rangle = \frac{1}{2}\sum_{i=1}^{4} |i\rangle.$$
2. Apply O_x, then U, then O_x again.
3. Perform the measurement consisting of a projection onto the state $|\psi\rangle$ and its orthogonal complement.

EXACT_2 algorithm

1. Create the state
$$|\psi\rangle = \frac{1}{2}\sum_{i=1}^{4} |i\rangle.$$
2. Apply O_x, then U, then O_x again.
3. Perform the measurement consisting of a projection onto the state $|\psi\rangle$ and its orthogonal complement.
4. If the outcome is $|\psi\rangle$, output 1, and otherwise 0.

EXACT_2 algorithm

1. Create the state
$$|\psi\rangle = \frac{1}{2}\sum_{i=1}^{4} |i\rangle.$$
2. Apply O_x, then U, then O_x again.
3. Perform the measurement consisting of a projection onto the state $|\psi\rangle$ and its orthogonal complement.
4. If the outcome is $|\psi\rangle$, output 1, and otherwise 0.

Theorem

The above algorithm uses 2 queries and computes the EXACT_2 function on 4 bits with certainty.

EXACT_2 algorithm

1. Create the state
$$|\psi\rangle = \frac{1}{2}\sum_{i=1}^{4}|i\rangle.$$
2. Apply O_x, then U, then O_x again.
3. Perform the measurement consisting of a projection onto the state $|\psi\rangle$ and its orthogonal complement.
4. If the outcome is $|\psi\rangle$, output 1, and otherwise 0.

Theorem

The above algorithm uses 2 queries and computes the EXACT_2 function on 4 bits with certainty.

The idea behind this algorithm can be extended to give an algorithm which distinguishes between $|x| = n/2$ and $|x| \in \{0, 1, n-1, n\}$, for all even n, using 2 queries.

Remaining functions on 3 bits

- This algorithm can clearly also be used to compute EXACT_2 on 3 bits.

Remaining functions on 3 bits

- This algorithm can clearly also be used to compute EXACT_2 on 3 bits.

- For the other functions on 3 bits ($x_1 \wedge (x_2 \vee x_3)$ and $(x_1 \wedge x_2) \vee (\bar{x_1} \wedge \bar{x_2} \wedge x_3)$) we also found explicit exact quantum query algorithms.

Remaining functions on 3 bits

- This algorithm can clearly also be used to compute EXACT_2 on 3 bits.
- For the other functions on 3 bits ($x_1 \wedge (x_2 \vee x_3)$ and $(x_1 \wedge x_2) \vee (\bar{x_1} \wedge \bar{x_2} \wedge x_3)$) we also found explicit exact quantum query algorithms.
- This was via a somewhat painful process of manually rounding real-valued solutions to the SDP to produce rational, exact solutions.

Remaining functions on 3 bits

- This algorithm can clearly also be used to compute EXACT_2 on 3 bits.

- For the other functions on 3 bits ($x_1 \wedge (x_2 \vee x_3)$ and $(x_1 \wedge x_2) \vee (\bar{x_1} \wedge \bar{x_2} \wedge x_3)$) we also found explicit exact quantum query algorithms.

- This was via a somewhat painful process of manually rounding real-valued solutions to the SDP to produce rational, exact solutions.

- But could there be an optimal quantum query algorithm for these functions based only on computing the parity of pairs of bits?

No!

Proposition

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.

No!

Proposition

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 sketch: the function computed by any decision tree on parities with depth D 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 $D-1$.

No!

Proposition

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 sketch: the function computed by any decision tree on parities with depth D 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 $D-1$.
- The functions EXACT$_2$ on 3 bits, $x_1 \wedge (x_2 \vee x_3)$ and $(x_1 \wedge x_2) \vee (\bar{x_1} \wedge \bar{x_2} \wedge x_3)$ all have degree 3.
- Therefore, optimal quantum algorithms for these functions cannot be obtained by computing parities of pairs of bits.

From small separations to growing separations

- Given a function $f : \{0, 1\}^k \to \{0, 1\}$ such that $Q_E(f) < D(f)$, we can amplify this separation.

From small separations to growing separations

- Given a function $f : \{0,1\}^k \to \{0,1\}$ such that $Q_E(f) < D(f)$, we can amplify this separation.
- Just define a new function $f_n : \{0,1\}^{nk} \to \{0,1\}$ by dividing the input into blocks $b_1, \ldots, b_n$ of k bits each, and set

$$f_n(x_1, \ldots, x_{nk}) = g(f(b_1), f(b_2), \ldots, f(b_n))$$

for some $g : \{0,1\}^n \to \{0,1\}$ such that $D(g) = n$.

From small separations to growing separations

- Given a function $f : \{0,1\}^k \to \{0,1\}$ such that $Q_E(f) < D(f)$, we can amplify this separation.
- Just define a new function $f_n : \{0,1\}^{nk} \to \{0,1\}$ by dividing the input into blocks $b_1, \ldots, b_n$ of k bits each, and set

$$f_n(x_1, \ldots, x_{nk}) = g(f(b_1), f(b_2), \ldots, f(b_n))$$

  for some $g : \{0,1\}^n \to \{0,1\}$ such that $D(g) = n$.
- Then $D(f_n) = n\,D(f)$ and $Q_E(f_n) \leqslant n\,Q_E(f)$.

From small separations to growing separations

- Given a function $f : \{0,1\}^k \to \{0,1\}$ such that $Q_E(f) < D(f)$, we can amplify this separation.
- Just define a new function $f_n : \{0,1\}^{nk} \to \{0,1\}$ by dividing the input into blocks $b_1, \dots, b_n$ of k bits each, and set

$$f_n(x_1, \dots, x_{nk}) = g(f(b_1), f(b_2), \dots, f(b_n))$$

  for some $g : \{0,1\}^n \to \{0,1\}$ such that $D(g) = n$.
- Then $D(f_n) = n\, D(f)$ and $Q_E(f_n) \leqslant n\, Q_E(f)$.

Example

Define $\text{EXACT}_2^\ell : \{0,1\}^{4\ell} \to \{0,1\}$ as follows. Split the input x into blocks containing 4 bits each, and set $\text{EXACT}_2^\ell(x) = 1$ if each block contains exactly two 1s.
Then $Q_E(\text{EXACT}_2^\ell) = 2\ell$ and $D(\text{EXACT}_2^\ell) = 4\ell$.

Nonadaptive exact quantum query complexity

We now turn to essentially the strictest non-trivial model of query complexity imaginable: nonadaptive query complexity.

- A nonadaptive (classical or quantum) query algorithm cannot choose queries based on the result of previous queries.
- In other words, the queries must all be made up front, in parallel.

Nonadaptive exact quantum query complexity

We now turn to essentially the strictest non-trivial model of query complexity imaginable: nonadaptive query complexity.

- A nonadaptive (classical or quantum) query algorithm cannot choose queries based on the result of previous queries.
- In other words, the queries must all be made up front, in parallel.
- Let $D^{na}(f)$, $Q_E^{na}(f)$ be the nonadaptive classical and quantum exact query complexities of f.

Nonadaptive exact quantum query complexity

We now turn to essentially the strictest non-trivial model of query complexity imaginable: nonadaptive query complexity.

- A nonadaptive (classical or quantum) query algorithm cannot choose queries based on the result of previous queries.
- In other words, the queries must all be made up front, in parallel.
- Let $D^{na}(f)$, $Q_E^{na}(f)$ be the nonadaptive classical and quantum exact query complexities of f.

Proposition

For any total boolean function f depending on n variables,

$$D^{na}(f) = n.$$

Nonadaptive exact quantum query complexity

Nonadaptive quantum query complexity is more complicated. But it turns out that we can still completely characterise it.

- For any $f : \{0,1\}^n \to \{0,1\}$, define the subspace

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

Nonadaptive exact quantum query complexity

Nonadaptive quantum query complexity is more complicated. But it turns out that we can still completely characterise it.

- For any $f : \{0,1\}^n \to \{0,1\}$, define the subspace

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

- For any subspace $S \subseteq \{0,1\}^n$, let $S^\perp$ denote the orthogonal subspace to S, i.e. $S^\perp = \{x : x \cdot s = 0, \forall s \in S\}$.

Nonadaptive exact quantum query complexity

Nonadaptive quantum query complexity is more complicated. But it turns out that we can still completely characterise it.

- For any $f : \{0,1\}^n \to \{0,1\}$, define the subspace

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

- For any subspace $S \subseteq \{0,1\}^n$, let $S^\perp$ denote the orthogonal subspace to S, i.e. $S^\perp = \{x : x \cdot s = 0, \forall s \in S\}$.

Theorem

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

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

Here $d(x,y)$ is the Hamming distance between x and y.

Nonadaptive exact quantum query complexity

In fact, the following explicit algorithm succeeds with certainty and achieves the above bound.

1. For some k, let $t \in \{0,1\}^n$ be a bit-string such that $\max_{y \in S_f^{\perp}} d(t,y) = k$.

Nonadaptive exact quantum query complexity

In fact, the following explicit algorithm succeeds with certainty and achieves the above bound.

1. For some k, let $t \in \{0,1\}^n$ be a bit-string such that $\max_{y \in S_f^\perp} d(t,y) = k$.
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.

Nonadaptive exact quantum query complexity

In fact, the following explicit algorithm succeeds with certainty and achieves the above bound.

1. For some k, let $t \in \{0,1\}^n$ be a bit-string such that $\max_{y \in S_f^\perp} d(t,y) = k$.
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.
3. Perform Hadamards on every qubit of the resulting state and measure to get outcome $\tilde{x}$.

Nonadaptive exact quantum query complexity

In fact, the following explicit algorithm succeeds with certainty and achieves the above bound.

1. For some k, let $t \in \{0,1\}^n$ be a bit-string such that $\max_{y \in S_f^\perp} d(t,y) = k$.
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.
3. Perform Hadamards on every qubit of the resulting state and measure to get outcome $\tilde{x}$.
4. Output $f(\tilde{x})$.

Nonadaptive exact quantum query complexity

In fact, the following explicit algorithm succeeds with certainty and achieves the above bound.

1. For some k, let $t \in \{0,1\}^n$ be a bit-string such that $\max_{y \in S_f^\perp} d(t,y) = k$.
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.
3. Perform Hadamards on every qubit of the resulting state and measure to get outcome $\tilde{x}$.
4. Output $f(\tilde{x})$.

Proposition

$$f(\tilde{x}) = f(x).$$

Some consequences

We can harness this characterisation to prove a number of results. For example, we have the following corollaries.

- If $f : \{0,1\}^n \to \{0,1\}$ depends on all n input bits, $Q_E^{na}(f) \geqslant \lceil n/2 \rceil$. This was previously known [AM '10].

Some consequences

We can harness this characterisation to prove a number of results. For example, we have the following corollaries.

- If $f : \{0,1\}^n \to \{0,1\}$ depends on all n input bits, $Q_E^{na}(f) \geqslant \lceil n/2 \rceil$. This was previously known [AM '10].

- If $f : \{0,1\}^n \to \{0,1\}$ does not satisfy $f(x) = f(x+a)$ for some a, $Q_E^{na}(f) = n$.

Some consequences

We can harness this characterisation to prove a number of results. For example, we have the following corollaries.

- If $f : \{0,1\}^n \to \{0,1\}$ depends on all n input bits, $Q_E^{na}(f) \geqslant \lceil n/2 \rceil$. This was previously known [AM '10].
- If $f : \{0,1\}^n \to \{0,1\}$ does not satisfy $f(x) = f(x+a)$ for some a, $Q_E^{na}(f) = n$.
- So almost all functions have $Q_E^{na}(f) = n$.

Some consequences

We can harness this characterisation to prove a number of results. For example, we have the following corollaries.

- If $f : \{0,1\}^n \to \{0,1\}$ depends on all n input bits, $Q_E^{na}(f) \geqslant \lceil n/2 \rceil$. This was previously known [AM '10].
- If $f : \{0,1\}^n \to \{0,1\}$ does not satisfy $f(x) = f(x+a)$ for some a, $Q_E^{na}(f) = n$.
- So almost all functions have $Q_E^{na}(f) = n$.
- For any $f : \{0,1\}^n \to \{0,1\}$ such that $f(x) = f(\bar{x})$ for all x, $Q_E^{na}(f) \leqslant n-1$.

Symmetric boolean functions

We can also prove the following quadrichotomy for symmetric boolean functions (functions $f : \{0,1\}^n \to \{0,1\}$ such that $f(x)$ depends only on $|x|$).

Corollary

If $f : \{0,1\}^n \to \{0,1\}$ is 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 $Q_E^{na}(f) = n$.

Conclusions

- There is more to exact quantum query complexity than computing parities.
- We've numerically computed the quantum query complexity of all boolean functions on up to 4 bits and used this to develop new quantum algorithms.
- As always, the basic open question still remains: can we achieve $Q_E(f) < D(f)/2$?

Conclusions

- There is more to exact quantum query complexity than computing parities.
- We've numerically computed the quantum query complexity of all boolean functions on up to 4 bits and used this to develop new quantum algorithms.
- As always, the basic open question still remains: can we achieve $Q_E(f) < D(f)/2$?

Our numerical results inspire many tantalising conjectures. For example:

Conjecture

For any n, the EXACT_k function on n bits can be computed exactly using $\max\{k, n-k\}$ quantum queries.

Thanks!

`arXiv:1111.0475`

(joint work with Richard Jozsa and Graeme Mitchison)