Lower bounds for advised quantum computations

Aran Nayebi

anayebi@stanford.edu

(joint work with Luca Trevisan, Aleksandrs Belovs, and Scott Aaronson)

Logic Seminar, Stanford

May 20, 2014

Definitions: Deterministic model

- Recall that a deterministic Turing machine is in a fixed state at every time step. At every time step t, there is a fixed transformation f_t that determines which state the machine enters into next.

Definitions: Deterministic model

- Recall that a deterministic Turing machine is in a fixed state at every time step. At every time step t, there is a fixed transformation f_t that determines which state the machine enters into next.
- If the TM runs for T steps, and starts in the m bit state $s_0 \in \{0,1\}^n$, then the end state is completely determined as

$$f_T(f_{T-1}(\ldots f_2(f_1(s_0))\ldots)). \tag{1}$$

Definitions: Probabilistic model

- The basic model for probabilistic TMs is that at every time step, different states are possible with different probabilities.

Definitions: Probabilistic model

- The basic model for probabilistic TMs is that at every time step, different states are possible with different probabilities.
- For all states $a \in \{0,1\}^n$, let p_a denote the probability that the machine is in state a. So, $p_a \geq 0$ and $\sum_{a \in \{0,1\}^n} p_a = 1$.

Definitions: Probabilistic model

- The basic model for probabilistic TMs is that at every time step, different states are possible with different probabilities.
- For all states $a \in \{0,1\}^n$, let p_a denote the probability that the machine is in state a. So, $p_a \geq 0$ and $\sum_{a \in \{0,1\}^n} p_a = 1$.
- At each step, different state-to-state transitions occur with different probabilities. A transition at any time step t can be viewed as a $2^n \times 2^n$ matrix $M^{(t)}$, where each row and column index corresponds to one of the 2^n possible states.

Definitions: Probabilistic model

- $M^{(t)}(b, a)$ is the entry in the b-th row and a-th column of $M^{(t)}$, which indicates the probability that if the machine is in state a at time t, then it will enter into state b at time $t + 1$.

Definitions: Probabilistic model

- $M^{(t)}(b, a)$ is the entry in the b-th row and a-th column of $M^{(t)}$, which indicates the probability that if the machine is in state a at time t, then it will enter into state b at time $t + 1$.
- Let p_0 be the $2^n \times 1$ vector that represents the initial probability distribution over all 2^n states.

Definitions: Probabilistic model

- $M^{(t)}(b, a)$ is the entry in the b-th row and a-th column of $M^{(t)}$, which indicates the probability that if the machine is in state a at time t, then it will enter into state b at time $t + 1$.
- Let p_0 be the $2^n \times 1$ vector that represents the initial probability distribution over all 2^n states.
- If the machine runs for T steps, then the final probability distribution over the states of the machine (which will be a $2^n \times 1$ vector) is the following matrix-vector product:

$$M^{(T)} M^{(T-1)} \dots M^{(2)} M^{(1)} p_0. \tag{2}$$

Definitions: Quantum model

- The n-qubit quantum state $|s\rangle$ where $s \in \{0,1\}^n$ is a computational basis state. It is a $2^n \times 1$ vector with a 1 in its entry for s and 0 everywhere else.

Definitions: Quantum model

- The n-qubit quantum state $|s\rangle$ where $s \in \{0,1\}^n$ is a computational basis state. It is a $2^n \times 1$ vector with a 1 in its entry for s and 0 everywhere else.
- At every time step, the state of quantum computer is in a superposition of basis states. A pure quantum state q is an assignment of amplitudes to each basis state $|s\rangle \in \{0,1\}^n$. Let q_s be the amplitude of the basis state $|s\rangle$, then:

$$|q\rangle = \sum_{s \in \{0,1\}^n} q_s |s\rangle.$$

Definitions: Quantum model

- The n-qubit quantum state $|s\rangle$ where $s \in \{0,1\}^n$ is a computational basis state. It is a $2^n \times 1$ vector with a 1 in its entry for s and 0 everywhere else.
- At every time step, the state of quantum computer is in a superposition of basis states. A pure quantum state q is an assignment of amplitudes to each basis state $|s\rangle \in \{0,1\}^n$. Let q_s be the amplitude of the basis state $|s\rangle$, then:

$$|q\rangle = \sum_{s \in \{0,1\}^n} q_s |s\rangle.$$

- $q_s \in \mathbb{C}$ and $\sum_{s \in \{0,1\}^n} |q_s|^2 = 1$.

Definitions: Quantum model

- By the Born rule, if we measure the state q, then the probability that we observe the basis state $|s\rangle$ occurs with probability $|q_s|^2$.

Definitions: Quantum model

- By the Born rule, if we measure the state q, then the probability that we observe the basis state $|s\rangle$ occurs with probability $|q_s|^2$.
- At each step t, we have a unitary transformation $U^{(t)}$ that changes the quantum state q to $U^{(t)}q$.

Definitions: Quantum model

- By the Born rule, if we measure the state q, then the probability that we observe the basis state $|s\rangle$ occurs with probability $|q_s|^2$.
- At each step t, we have a unitary transformation $U^{(t)}$ that changes the quantum state q to $U^{(t)}q$.
- If q_0 is the initial state of computer, then after T steps, the quantum computer is in the final quantum state q_T:

$$q_T = U^{(T)}U^{(T-1)}\dots U^{(2)}U^{(1)}q_0. \tag{3}$$

Definitions: Quantum model

- By the Born rule, if we measure the state q, then the probability that we observe the basis state $|s\rangle$ occurs with probability $|q_s|^2$.
- At each step t, we have a unitary transformation $U^{(t)}$ that changes the quantum state q to $U^{(t)}q$.
- If q_0 is the initial state of computer, then after T steps, the quantum computer is in the final quantum state q_T:

$$q_T = U^{(T)}U^{(T-1)}\dots U^{(2)}U^{(1)}q_0. \tag{3}$$

- If we measure q_T, then that will be the output of the computer.

Definitions: Quantum query model

- Specifically, we will work in the query model, which will allow us to formalize the cost of a quantum algorithm.

Definitions: Quantum query model

- Specifically, we will work in the query model, which will allow us to formalize the cost of a quantum algorithm.
- The goal is to compute a Boolean function $f : \{0,1\}^n \to \{0,1\}$ given an input $x \in \{0,1\}^n$.

Definitions: Quantum query model

- Specifically, we will work in the query model, which will allow us to formalize the cost of a quantum algorithm.
- The goal is to compute a Boolean function $f : \{0,1\}^n \to \{0,1\}$ given an input $x \in \{0,1\}^n$.
- x is not given explicitly, but is instead stored in random access memory, and we are being charged unit cost for each *query* that we make to this memory.

Definitions: Quantum query model

- Specifically, we will work in the query model, which will allow us to formalize the cost of a quantum algorithm.
- The goal is to compute a Boolean function $f : \{0,1\}^n \to \{0,1\}$ given an input $x \in \{0,1\}^n$.
- x is not given explicitly, but is instead stored in random access memory, and we are being charged unit cost for each *query* that we make to this memory.
- A query essentially asks for and receives the i-th bit x_i of the input. Formally, a query is modeled unitarily as the operation O_x:

$$O_x : |i\rangle \mapsto (-1)^{x_i}|i\rangle.$$

  Thus, the phase of $|i\rangle$ is flipped if $x_i = 1$, and left unchanged if $x_i = 0$.

Definitions: Quantum query model

- A T-query quantum algorithm starts in a fixed state, usually the all zero state $|0\ldots0\rangle$, and then interleaves fixed unitary transformations $U_0, U_1, \ldots, U_T$ with queries.

Definitions: Quantum query model

- A T-query quantum algorithm starts in a fixed state, usually the all zero state $|0\ldots0\rangle$, and then interleaves fixed unitary transformations $U_0, U_1, \ldots, U_T$ with queries.
- The final state of the algorithm can be written as the following matrix-vector product:

$$U_T O_x U_{T-1} O_x \cdots O_x U_1 O_x U_0 |0\ldots0\rangle. \tag{4}$$

Definitions: Quantum query model

- A T-query quantum algorithm starts in a fixed state, usually the all zero state $|0\ldots0\rangle$, and then interleaves fixed unitary transformations $U_0, U_1, \ldots, U_T$ with queries.
- The final state of the algorithm can be written as the following matrix-vector product:

$$U_T O_x U_{T-1} O_x \cdots O_x U_1 O_x U_0 |0\ldots0\rangle. \tag{4}$$

- This state depends on the input x only via the T queries, and the output of the algorithm is obtained by a measurement of the final state (4).

Definitions: Quantum query model

- The query complexity of f is the minimal number of queries needed for an algorithm that outputs the correct value of $f(x)$ for every x in the domain of f (with error probability ϵ, which is usually taken to be $\frac{1}{3}$).

Definitions: Quantum query model

- The query complexity of f is the minimal number of queries needed for an algorithm that outputs the correct value of $f(x)$ for every x in the domain of f (with error probability ϵ, which is usually taken to be $\frac{1}{3}$).
- Note that we count only the number of queries to measure the complexity of the algorithm, while the intermediate unitary operations $U_0, U_1, \ldots, U_T$ are treated as costless.

Lower bound techniques

- There are two primary quantum lower bound techniques:

Lower bound techniques

- There are two primary quantum lower bound techniques:
- The polynomial method

Lower bound techniques

- There are two primary quantum lower bound techniques:
- The polynomial method
- The adversary method (based on the earlier hybrid method)

The polynomial method

- An n-variate multilinear polynomial p is a function $p : \mathbb{C}^n \to \mathbb{C}$ that can be written as

$$p(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} a_S \prod_{i \in S} x_i,$$

where $a_S \in \mathbb{C}$. Of course, $deg(p) = \max\{|S| : a_S \neq 0\}$.

The polynomial method

- An n-variate multilinear polynomial p is a function $p : \mathbb{C}^n \to \mathbb{C}$ that can be written as

$$p(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} a_S \prod_{i \in S} x_i,$$

  where $a_S \in \mathbb{C}$. Of course, $deg(p) = \max\{|S| : a_S \neq 0\}$.
- A useful property of T-query algorithms is that the amplitudes of their final state are degree-T n-variate polynomials.

The polynomial method

- Namely, for a T-query algorithm with input $x \in \{0,1\}^n$, the final state can be written as

$$\sum_{z \in \{0,1\}^m} q_z(x)|z\rangle,$$

where each $q_z(x)$ is a multilinear polynomial of degree at most T.

The polynomial method

- Namely, for a T-query algorithm with input $x \in \{0,1\}^n$, the final state can be written as

$$\sum_{z \in \{0,1\}^m} q_z(x)|z\rangle,$$

where each $q_z(x)$ is a multilinear polynomial of degree at most T.

- Therefore, the probability that the algorithm accepts is

$$\sum_{z \in \{0,1\}^m} |q_z(x)|^2,$$

which is a multilinear polynomial $p(x)$ of degree at most $2T$. Hence, $T \geq deg(p)/2$.

The adversary method

- Polynomial method applies to an even stronger model of computation where we allow any linear transformation on the state space, not just unitary ones. As a result, does not always provide the strongest possible lower bound for quantum query algorithms.

The adversary method

- Polynomial method applies to an even stronger model of computation where we allow any linear transformation on the state space, not just unitary ones. As a result, does not always provide the strongest possible lower bound for quantum query algorithms.
- An alternative method is known as the "quantum adversary method" which exploits unitarity in a crucial way, and has many (equivalent) versions. Gives optimal lower bounds for every Boolean function!

The adversary method

- The classical adversary/hybrid argument is as follows: an adversary runs the algorithm with one input and, after that, changes the input slightly so that the correct answer changes but the algorithm does not notice that.

The adversary method

- The classical adversary/hybrid argument is as follows: an adversary runs the algorithm with one input and, after that, changes the input slightly so that the correct answer changes but the algorithm does not notice that.
- Instead of a classical adversary that runs the algorithm with one input and then modifies the input, the more modern adversary method is that a quantum adversary runs the algorithm with a superposition of inputs. The modern (unweighted) adversary method requires one to construct a binary relation R on the sets of inputs on which f differs.

The hybrid argument

- Since this is the argument that will be the machinery for the lower bounds proven in this talk, we will discuss this technique in more detail.

The hybrid argument

- Since this is the argument that will be the machinery for the lower bounds proven in this talk, we will discuss this technique in more detail.
- Fix a quantum algorithm $\mathcal{A}$. The state of $\mathcal{A}$ between successive queries can be written as

$$|\phi\rangle = \sum_{c} \alpha_c |c\rangle,$$

where c runs over the computational basis states.

The hybrid argument

- Since this is the argument that will be the machinery for the lower bounds proven in this talk, we will discuss this technique in more detail.
- Fix a quantum algorithm $\mathcal{A}$. The state of $\mathcal{A}$ between successive queries can be written as

$$|\phi\rangle = \sum_c \alpha_c |c\rangle,$$

  where c runs over the computational basis states.
- During each query, each basis state c queries a particular bit position of the input. The main question is: how sensitive is the output of $\mathcal{A}$ to the modification of the input on a few bit positions?

The hybrid argument

- The *query magnitude* at bit position j of $|\phi\rangle = \sum_c \alpha_c |c\rangle$ is defined to be $q_j(|\phi\rangle) = \sum_{c \in C_j} |\alpha_c|^2$, where C_j is the set of all computational basis states that query bit position j.

The hybrid argument

- The *query magnitude* at bit position j of $|\phi\rangle = \sum_c \alpha_c |c\rangle$ is defined to be $q_j(|\phi\rangle) = \sum_{c \in C_j} |\alpha_c|^2$, where C_j is the set of all computational basis states that query bit position j.
- Now, suppose $\mathcal{A}$ runs for T steps on an input $x \in \{0,1\}^n$. Then the run of $\mathcal{A}$ on input x can be described as a sequence of states $|\phi_0\rangle, \ldots, |\phi_T\rangle$, where $|\phi_k\rangle$ is the state of $\mathcal{A}$ before the $k+1^{\text{st}}$ query.

The hybrid argument

- The *query magnitude* at bit position j of $|\phi\rangle = \sum_c \alpha_c |c\rangle$ is defined to be $q_j(|\phi\rangle) = \sum_{c \in C_j} |\alpha_c|^2$, where C_j is the set of all computational basis states that query bit position j.
- Now, suppose $\mathcal{A}$ runs for T steps on an input $x \in \{0,1\}^n$. Then the run of $\mathcal{A}$ on input x can be described as a sequence of states $|\phi_0\rangle, \ldots, |\phi_T\rangle$, where $|\phi_k\rangle$ is the state of $\mathcal{A}$ before the $k+1^{\text{st}}$ query.
- The *total query magnitude* at bit position j of $\mathcal{A}$ on input x is defined to be:

$$q_j(x) = \sum_{k=0}^{T-1} q_j(|\phi_k\rangle). \tag{5}$$

The hybrid argument

- By definition, $\sum_j q_j(x) = T$.

The hybrid argument

- By definition, $\sum_j q_j(x) = T$.
- It is tempting to think of $q_j(x)$ as the probability that $\mathcal{A}$ probes bit position j.

The hybrid argument

- By definition, $\sum_j q_j(x) = T$.
- It is tempting to think of $q_j(x)$ as the probability that $\mathcal{A}$ probes bit position j.
- However, this is misguided since there are no probabilities during the execution of the algorithm, just probability amplitudes that might interfere constructively or destructively.

The hybrid argument

- Nevertheless, if the total query magnitude of bit position j is very small, then $\mathcal{A}$ cannot distinguish whether the input x is modified by flipping its j-th bit:

Theorem ("Swapping lemma")

Let $|\phi_x\rangle$ and $|\phi_y\rangle$ denote the final states of $\mathcal{A}$ on inputs x and y, respectively. Then:

$$\||\phi_x\rangle - |\phi_y\rangle\| \leq \sqrt{T \sum_{j \in \Delta(x,y)} q_j(x)}, \tag{6}$$

where $\Delta(x, y) = \{j : \ x_j \neq y_j\}$.

Yao's box problem

- Let N, m be positive integers. Consider the following game to by played by A and B. There are N boxes with lids $\mathrm{BOX}_1, \mathrm{BOX}_2, \ldots, \mathrm{BOX}_N$ each containing a Boolean bit. In the preprocessing stage, Player A will inspect the bits and take notes using an m-bit pad. Afterwards, Player B will ask Player A a question of the form "What is in BOX_i?". Before answering the question, Player A is allowed to consult the m-bit pad and take off the lids of a chosen sequence of boxes not including BOX_i. The puzzle is, what is the minimum number of boxes A needs to examine in order to find the answer?

Classical lower bound

- On a classical computer, we get that $T \geq \lceil N/m \rceil - 1$, and in fact an optimal strategy that A may adopt is divide-and-conquer.

Classical lower bound

- On a classical computer, we get that $T \geq \lceil N/m \rceil - 1$, and in fact an optimal strategy that A may adopt is divide-and-conquer.
- Divide the boxes into m consecutive groups each containing no more than $\lceil N/m \rceil$ members.

Classical lower bound

- On a classical computer, we get that $T \geq \lceil N/m \rceil - 1$, and in fact an optimal strategy that A may adopt is divide-and-conquer.
- Divide the boxes into m consecutive groups each containing no more than $\lceil N/m \rceil$ members.
- In the preprocessing stage, A records for each group g the parity a_g of the bits in that group.

Classical lower bound

- On a classical computer, we get that $T \geq \lceil N/m \rceil - 1$, and in fact an optimal strategy that A may adopt is divide-and-conquer.
- Divide the boxes into m consecutive groups each containing no more than $\lceil N/m \rceil$ members.
- In the preprocessing stage, A records for each group g the parity a_g of the bits in that group.
- Then, to answer the query "What is in BOX_i?", A can lift the lids of all the boxes (except BOX_i) in the group k containing BOX_i and compute the parity b of these lids. Clearly, the bit in BOX_i is $a_k \oplus b$. Therefore, A never needs to lift more than $\lceil N/m \rceil - 1$ lids with this strategy.

Yao's advice model

- We define the model more precisely. Let F_N be the set of all possible 2^N bit patterns of the N boxes. Fix our m-bit advice string $\alpha \in \{0,1\}^m$.

Yao's advice model

- We define the model more precisely. Let F_N be the set of all possible 2^N bit patterns of the N boxes. Fix our m-bit advice string $\alpha \in \{0,1\}^m$.
- This induces a partition $D_\alpha \subseteq \{0,1\}^N$ and N partial Boolean functions f_j $(1 \leq j \leq N)$ with domain D_α^{-j} (namely, the set of all the $N-1$ bit strings formed from the strings in D_α with the j-th bit removed).

Yao's advice model

- We define the model more precisely. Let F_N be the set of all possible 2^N bit patterns of the N boxes. Fix our m-bit advice string $\alpha \in \{0,1\}^m$.
- This induces a partition $D_\alpha \subseteq \{0,1\}^N$ and N partial Boolean functions f_j $(1 \leq j \leq N)$ with domain D_α^{-j} (namely, the set of all the $N-1$ bit strings formed from the strings in D_α with the j-th bit removed).
- One can think of f_j as the function that computes the bit of the j-th box given advice string α from the remaining $N-1$ boxes.

Yao's advice model

- The following observation will be useful: For any two m-bit advice strings, α' and α'', it is <u>not</u> necessarily the case that $D_{\alpha'} \cap D_{\alpha''} = \emptyset$.

Yao's advice model

- The following observation will be useful: For any two m-bit advice strings, α' and α'', it is <u>not</u> necessarily the case that $D_{\alpha'} \cap D_{\alpha''} = \emptyset$.
- Therefore,

$$2^N \leq \sum_{\alpha \in \{0,1\}^m} |D_\alpha| \leq 2^{N+m},$$

from which it follows that $2^{N-m} \leq |D_\alpha| \leq 2^N$.

Quantum lower bound

Theorem

Let $\mathcal{D}(\psi)$ denote the probability distribution that results from a measurement of $|\psi\rangle$ in the computational basis. If $\||\phi\rangle - |\psi\rangle\| \leq \epsilon$, then $\|\mathcal{D}(\phi) - \mathcal{D}(\psi)\|_1 \leq 4\epsilon$.

Quantum lower bound

Theorem

Suppose $D \subseteq \{0,1\}^n$ of size 2^{n-m} and that we randomly select a set I of $m+1$ indices. Then with probability 1 there are two strings x and y in D that differ only in a subset of the coordinates in I.

Proof.

There are only $2^{n-(m+1)}$ different ways of fixing the coordinates not in I, and since this is less than the number of elements of D, there must be two or more elements of D that are identical outside of I. □

Quantum lower bound

Theorem

$Q_\epsilon(f_j) \geq (\epsilon/4)\sqrt{\frac{N-1}{m+1}}$.

Quantum lower bound

Proof.

Let I be a randomly selected set of $m+1$ indices, and let x and y be the two strings in D_α^{-j} that differ only in a subset of the coordinates in I. Therefore,

$$\begin{aligned} \||\phi_x\rangle - |\phi_y\rangle\| &\leq \sqrt{T \sum_{k \in \Delta(x,y)} q_k(x)} \\ &\leq \sqrt{T\left(\frac{T(m+1)}{N-1}\right)} = T\sqrt{\frac{m+1}{N-1}}. \end{aligned}$$

Assume that $\mathcal{A}$, the algorithm computing f_j, errs with probability bounded by ϵ. Then if $T < (\epsilon/4)\sqrt{\frac{N-1}{m+1}}$, it would follow that $\|\mathcal{D}(\phi) - \mathcal{D}(\psi)\|_1 < \epsilon$, a contradiction. Therefore, $T \geq (\epsilon/4)\sqrt{\frac{N-1}{m+1}}$. □

Quantum upper bound?

- Can we also obtain a quantum algorithm that solves this problem in $O(\sqrt{N/m})$ time?

Quantum upper bound?

- Can we also obtain a quantum algorithm that solves this problem in $O(\sqrt{N/m})$ time?
- If we use the same advice as Yao, then we are reduced to solving parity, which takes time $\Theta(N)$ on a quantum computer (this is optimal - can be shown by polynomial method).

Quantum upper bound?

- Can we also obtain a quantum algorithm that solves this problem in $O(\sqrt{N/m})$ time?
- If we use the same advice as Yao, then we are reduced to solving parity, which takes time $\Theta(N)$ on a quantum computer (this is optimal - can be shown by polynomial method).
- Say we change our advice to the number of boxes with a 1 in them. Our advice says that there are a total of r boxes with ones in them. If we are asked to find the value in box i, we know that the number of ones in the remaining $N-1$ boxes is either r or $r-1$. To find out the value of box i, we just have to figure out which case we are in.

Quantum upper bound?

- Can we also obtain a quantum algorithm that solves this problem in $O(\sqrt{N/m})$ time?
- If we use the same advice as Yao, then we are reduced to solving parity, which takes time $\Theta(N)$ on a quantum computer (this is optimal - can be shown by polynomial method).
- Say we change our advice to the number of boxes with a 1 in them. Our advice says that there are a total of r boxes with ones in them. If we are asked to find the value in box i, we know that the number of ones in the remaining $N-1$ boxes is either r or $r-1$. To find out the value of box i, we just have to figure out which case we are in.
- But this is just quantum counting. However, to distinguish exactly what case we are in for $r \geq N/2$, we cannot do better than $O(N)$ queries (which is the classical result).

Inverting a permutation

- We are given oracle access to a bijection $f : [N] \to [N]$, and $f(x)$ as input. Output: x.

Inverting a permutation

- We are given oracle access to a bijection $f : [N] \to [N]$, and $f(x)$ as input. Output: x.
- Without advice, the classical lower bound is $T \geq N$. The quantum lower bound is $T \geq \Omega(N)$ (optimal by Grover's algorithm).

Inverting a permutation

- We are given oracle access to a bijection $f : [N] \to [N]$, and $f(x)$ as input. Output: x.
- Without advice, the classical lower bound is $T \geq N$. The quantum lower bound is $T \geq \Omega(N)$ (optimal by Grover's algorithm).
- What is the quantum lower bound if we have advice? (asked by De at al. 2009)

Inverting a permutation

- Classically, if we have m bits of advice then we can invert a permutation in time $O(N/m)$ (due to Hellman).

Inverting a permutation

- Classically, if we have m bits of advice then we can invert a permutation in time $O(N/m)$ (due to Hellman).
- Basic idea ($m = O(\sqrt{N})$): Suppose for simplicity that f is a cyclic permutation and $N = m^2$ is a perfect square: then our advice will be $\sqrt{N}$ "equally spaced" points $x_1, \ldots, x_m$, such that $x_{i+1} = f^{(s)}(x_i)$.

Inverting a permutation

- Classically, if we have m bits of advice then we can invert a permutation in time $O(N/m)$ (due to Hellman).
- Basic idea ($m = O(\sqrt{N})$): Suppose for simplicity that f is a cyclic permutation and $N = m^2$ is a perfect square: then our advice will be $\sqrt{N}$ "equally spaced" points $x_1, \ldots, x_m$, such that $x_{i+1} = f^{(s)}(x_i)$.
- Given y, we compute $f(y), f(f(y))$, and so on, until for some j we reach a point $f^{(j)}(x)$ in the advice. Then we read the value $f^{(j-s)}(y)$, and by repeatedly computing f we eventually reach $f^{(-1)}(y)$.

Inverting a permutation

- Classically, if we have m bits of advice then we can invert a permutation in time $O(N/m)$ (due to Hellman).
- Basic idea ($m = O(\sqrt{N})$): Suppose for simplicity that f is a cyclic permutation and $N = m^2$ is a perfect square: then our advice will be $\sqrt{N}$ "equally spaced" points $x_1, \ldots, x_m$, such that $x_{i+1} = f^{(s)}(x_i)$.
- Given y, we compute $f(y), f(f(y))$, and so on, until for some j we reach a point $f^{(j)}(x)$ in the advice. Then we read the value $f^{(j-s)}(y)$, and by repeatedly computing f we eventually reach $f^{(-1)}(y)$.
- This takes $O(m)$ evaluations of f and lookups in the advice, so both time and advice complexity are approximately $O(\sqrt{N})$.

Quantum lower bound

- We prove a tradeoff of $m \cdot T^2 = \tilde{\Omega}(\epsilon N)$, where $\mathcal{A}$ successfully inverts f on an ϵ fraction of inputs. We provide a simple proof sketch for $\epsilon = 1$.

Quantum lower bound

- We prove a tradeoff of $m \cdot T^2 = \tilde{\Omega}(\epsilon N)$, where $\mathcal{A}$ successfully inverts f on an ϵ fraction of inputs. We provide a simple proof sketch for $\epsilon = 1$.
- To show that $m \geq \tilde{\Omega}(N/T^2)$, we will first show the existence of a "good" set $G \subseteq [N]$ with low query magnitude.

Quantum lower bound

- For ease of notation, let $M_{y,z}$ be the total query magnitude of $f^{-1}(z)$ in the computation $\mathcal{A}^f(y)$ (we will take $M_{y,y} = 0$).

Quantum lower bound

- For ease of notation, let $M_{y,z}$ be the total query magnitude of $f^{-1}(z)$ in the computation $\mathcal{A}^f(y)$ (we will take $M_{y,y} = 0$).
- By definition of query magnitude, $\forall y$, $\sum_z M_{y,z} = T$.

Quantum lower bound

- For ease of notation, let $M_{y,z}$ be the total query magnitude of $f^{-1}(z)$ in the computation $\mathcal{A}^f(y)$ (we will take $M_{y,y} = 0$).
- By definition of query magnitude, $\forall y$, $\sum_z M_{y,z} = T$.
- Construct a random set $C \subseteq [N]$ by taking each $z \in [N]$ with probability δ/T^2, for some $\delta > 0$. (δ will be determined later)

Quantum lower bound

- For each fixed y,

$$\begin{aligned} \mathop{\mathbb{E}}_{C}\left[\sum_{z\in C} M_{y,z}\right] &= \mathop{\mathbb{E}}_{C}\left[\sum_{z\in[N]} I_{z\in C} M_{y,z}\right] \\ &= \sum_{z\in[N]} \frac{\delta}{T^2} M_{y,z} = \frac{\delta}{T}. \end{aligned} \tag{7}$$

Quantum lower bound

- For each fixed y,

$$\begin{aligned}\mathop{\mathbb{E}}_{C}\left[\sum_{z\in C} M_{y,z}\right] &= \mathop{\mathbb{E}}_{C}\left[\sum_{z\in[N]} I_{z\in C} M_{y,z}\right] \\ &= \sum_{z\in[N]} \frac{\delta}{T^2} M_{y,z} = \frac{\delta}{T}.\end{aligned} \tag{7}$$

- By Markov's inequality, for each fixed y,

$$\mathop{\mathbb{P}}_{C}\left[\sum_{z\in C} M_{y,z} < \frac{2\delta}{T}\right] > \frac{1}{2}. \tag{8}$$

Quantum lower bound

- It is clear that the two events:

Quantum lower bound

- It is clear that the two events:
- $\sum_{z \in C} M_{y,z} < \frac{2\delta}{T}$

Quantum lower bound

- It is clear that the two events:
- $\sum_{z \in C} M_{y,z} < \frac{2\delta}{T}$
- $y \in C$

Quantum lower bound

- It is clear that the two events:
- $\sum_{z \in C} M_{y,z} < \frac{2\delta}{T}$
- $y \in C$
- are independent.

Quantum lower bound

- So we have that

$$\mathop{\mathbb{P}}_{C}\left[(y \in C) \wedge \left(\sum_{z \in C} M_{y,z} < \frac{2\delta}{T}\right)\right] > \frac{\delta}{2T^2}. \tag{9}$$

Quantum lower bound

- So we have that

$$\mathop{\mathbb{P}}_{C}\left[(y \in C) \wedge \left(\sum_{z \in C} M_{y,z} < \frac{2\delta}{T}\right)\right] > \frac{\delta}{2T^2}. \tag{9}$$

- Therefore,

$$\mathop{\mathbb{E}}_{C}\left[\#y \in C.\ \sum_{z \in C} M_{y,z} < \frac{2\delta}{T}\right] > \frac{N\delta}{2T^2}. \tag{10}$$

Quantum lower bound

- Let $G = \{y \in C \mid \sum_{z \in C} M_{y,z} < \frac{2\delta}{T}\}$.

Quantum lower bound

- Let $G = \{y \in C \mid \sum_{z \in C} M_{y,z} < \frac{2\delta}{T}\}$.
- Hence, by (9) and (10), there exists a set C such that

$$|G| > \frac{N\delta}{2T^2}. \tag{11}$$

Quantum lower bound

- Let $G = \{y \in C \mid \sum_{z \in C} M_{y,z} < \frac{2\delta}{T}\}$.
- Hence, by (9) and (10), there exists a set C such that

$$|G| > \frac{N\delta}{2T^2}. \tag{11}$$

- In fact, (11) implies that $m \geq \tilde{\Omega}(N/T^2)$.

Quantum lower bound

- To see this, let

$$L = \{(f^{-1}(y), y) \mid \ y \notin G\}.$$

Quantum lower bound

- To see this, let

$$L = \{(f^{-1}(y), y) \mid \ y \notin G\}.$$

- Then for every $y \in [N]$ we can reconstruct $f^{-1}(y)$ as such:

Quantum lower bound

- To see this, let

$$L = \{(f^{-1}(y), y) \mid \ y \notin G\}.$$

- Then for every $y \in [N]$ we can reconstruct $f^{-1}(y)$ as such:
- If $y \notin G$, then the pair $(f^{-1}(y), y) \in L$. Otherwise, if $y \in G$, then define the oracle h as follows:

$$h(z) = \begin{cases} f(z) \text{ if } f(z) \notin G \\ y \text{ if } f(z) \in G \end{cases} \tag{12}$$

Quantum lower bound

- By the swapping lemma, if we take $\delta = \frac{1}{3200}$, then with probability $\geq 90\%$, $\mathcal{A}^h(f(x))$ outputs x.

Quantum lower bound

- By the swapping lemma, if we take $\delta = \frac{1}{3200}$, then with probability $\geq 90\%$, $\mathcal{A}^h(f(x))$ outputs x.
- Specifying G takes $\log \binom{N}{|G|}$ bits.

Quantum lower bound

- By the swapping lemma, if we take $\delta = \frac{1}{3200}$, then with probability $\geq 90\%$, $\mathcal{A}^h(f(x))$ outputs x.
- Specifying G takes $\log \binom{N}{|G|}$ bits.
- Specifying L takes $\log \binom{N}{|G|}$ bits (since we need to specify $f^{-1}(G)$) and then we are left with specifying $f : N \setminus G \to f(N \setminus G)$, which takes $\log(N - |G|)!$ bits.

Quantum lower bound

- By the swapping lemma, if we take $\delta = \frac{1}{3200}$, then with probability $\geq 90\%$, $\mathcal{A}^h(f(x))$ outputs x.
- Specifying G takes $\log \binom{N}{|G|}$ bits.
- Specifying L takes $\log \binom{N}{|G|}$ bits (since we need to specify $f^{-1}(G)$) and then we are left with specifying $f : N \setminus G \to f(N \setminus G)$, which takes $\log(N - |G|)!$ bits.
- Finally, the number of bits left to specify the advice is:

$$\log N! - \left(2 \log \binom{N}{|G|} + \log(N - |G|)!\right) \sim |G|/polylog(N).$$

Quantum upper bound?

- Can we invert a permutation in time $O(\sqrt{N/m})$?

Quantum upper bound?

- Can we invert a permutation in time $O(\sqrt{N/m})$?
- It seems unlikely that any classical advice other than iterates of f would be useful for solving this problem. Moreover, the bottleneck seems to be the speed at which one can iterate f.

Quantum upper bound?

- Can we invert a permutation in time $O(\sqrt{N/m})$?
- It seems unlikely that any classical advice other than iterates of f would be useful for solving this problem. Moreover, the bottleneck seems to be the speed at which one can iterate f.
- Function iteration is not sped up by a quantum computer.

Quantum advice

- The advice we considered here was classical, as that seemed to be most intuitive.

Quantum advice

- The advice we considered here was classical, as that seemed to be most intuitive.
- Advice can also be quantum. Difficulty here is we cannot copy quantum advice (due to the "no cloning" theorem).

Quantum advice

- The advice we considered here was classical, as that seemed to be most intuitive.
- Advice can also be quantum. Difficulty here is we cannot copy quantum advice (due to the "no cloning" theorem).
- Cannot adapt our current proof for permutation inversion directly because we have to get the elements of G repeatedly with the same piece of advice.

Other lower bound methods with advice

- Another direction would be to possibly improve the lower bounds we obtain (maybe by some variant of the adversary method?)

Other lower bound methods with advice

- Another direction would be to possibly improve the lower bounds we obtain (maybe by some variant of the adversary method?)
- Only other technique I am aware of is that of Nishimura and Yamakami. However, it only applies to the case of nonadaptive queries and classical advice, so it would not apply to the problem of permutation inversion as our techniques do.

Other lower bound methods with advice

- Another direction would be to possibly improve the lower bounds we obtain (maybe by some variant of the adversary method?)
- Only other technique I am aware of is that of Nishimura and Yamakami. However, it only applies to the case of nonadaptive queries and classical advice, so it would not apply to the problem of permutation inversion as our techniques do.
- Difficulty applying standard techniques such as the polynomial method or adversary method for the simpler box problem since f_j is partial and D_α^{-j} is arbitrary (so it is not clear how to construct weighted or unweighted scheme).

Upper bounds

- Prove upper bounds for the box and permutation inversion problems with classical (or quantum) advice.

Upper bounds

- Prove upper bounds for the box and permutation inversion problems with classical (or quantum) advice.
- I suspect that with classical advice one cannot do better than the classical results, but with quantum advice a quantum speedup may be possible.

Upper bounds

- Prove upper bounds for the box and permutation inversion problems with classical (or quantum) advice.
- I suspect that with classical advice one cannot do better than the classical results, but with quantum advice a quantum speedup may be possible.
- However, if there is a quantum speedup for the case of classical advice, it will be a novel algorithm, since simply using Grover's search algorithm as a subroutine (or with randomness) does not seem to help.

Selected References

- De, A., L. Trevisan, and M. Tulsiani. "Non-uniform attacks against one-way functions and PRGs". *Electronic Colloquium on Computational Complexity* Report No. 113 (2009). `http://eccc.hpi-web.de/report/2009/113/`.
- de Wolf, R. "Quantum Computing: Lecture Notes". `http://homepages.cwi.nl/~rdewolf/qc13.html`.
- Nishimura, H. and T. Yamakami. "An algorithmic argument for nonadaptive query complexity lower bounds on advised quantum computation". Full version at: `http://arxiv.org/abs/quant-ph/0312003`.
- Vazirani, U. "On the power of quantum computation". *Phil. Trans. R. Soc. Lond.* A **365** (1998): 1759-1768.