A New Quantum Lower-Bound Method, with Applications to Direct Product Theorems and Time-Space Tradeoffs

Robert Špalek*

joint work with Andris Ambainis† and Ronald de Wolf*

*CWI, Amsterdam   †University of Waterloo

Quantum algorithms

- Grover search: find a given number in an unsorted database of n records in time

$$O(\sqrt{n})$$

- element distinctness: find a collision $x_i = x_j$ in time

$$O(n^{2/3})$$

Quantum algorithms

- Grover search: find a given number in an unsorted database of n records in time

$$O(\sqrt{n})$$

- element distinctness: find a collision $x_i = x_j$ in time

$$O(n^{2/3})$$

Quantum query complexity

- allow quantum superposition, unitary evolution, and measurements
- count the number of queries, one query maps

$$|i, z\rangle \to (-1)^{x_i} |i, z\rangle$$

$i =$ queried bit

$z =$ the rest of memory

Quantum query lower bounds

Adversary method

- [Bennett, Bernstein, Brassard & Vazirani, 1994]
  tight lower bound $\Omega(\sqrt{n})$ for Grover search
  known 2 years before discovering the algorithm

Quantum query lower bounds

Adversary method

- [Bennett, Bernstein, Brassard & Vazirani, 1994]
  tight lower bound $\Omega(\sqrt{n})$ for Grover search
  known 2 years before discovering the algorithm
- [Ambainis, 2000 & 2003] generalized to all functions
- easy to use, gives strong bounds

Quantum query lower bounds

Adversary method

- [Bennett, Bernstein, Brassard & Vazirani, 1994]
  tight lower bound $\Omega(\sqrt{n})$ for Grover search
  known 2 years before discovering the algorithm
- [Ambainis, 2000 & 2003] generalized to all functions
- easy to use, gives strong bounds

Polynomial method
[Beals, Buhrman, Cleve, Mosca & de Wolf, 2000]

- incomparable to the adversary method
- hard to use for non-symmetric functions

Quantum query lower bounds

Adversary method

- [Bennett, Bernstein, Brassard & Vazirani, 1994]
  tight lower bound $\Omega(\sqrt{n})$ for Grover search
  known 2 years before discovering the algorithm
- [Ambainis, 2000 & 2003] generalized to all functions
- easy to use, gives strong bounds

Polynomial method
[Beals, Buhrman, Cleve, Mosca & de Wolf, 2000]

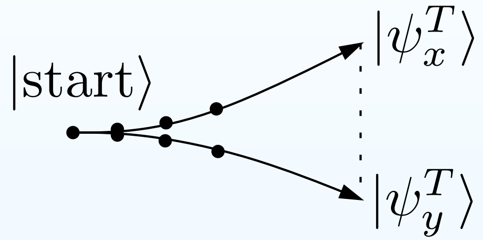
- incomparable to the adversary method
- hard to use for non-symmetric functions
- [Aaronson & Shi, 2002]
  tight lower bound $\Omega(n^{2/3})$ for element distinctness

Adversary lower bounds

- if an algorithm computes f, then it must *distinguish* between x, y such that $f(x) \neq f(y)$

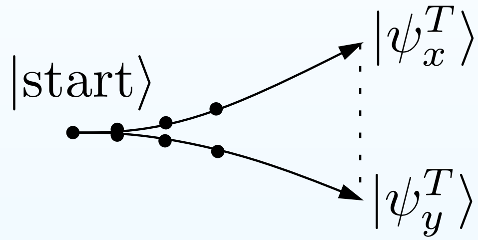
Adversary lower bounds

- if an algorithm computes f, then it must *distinguish* between x, y such that $f(x) \neq f(y)$
- computation starts in a fixed state and it has to diverge far enough after T queries for each such x, y



Adversary lower bounds

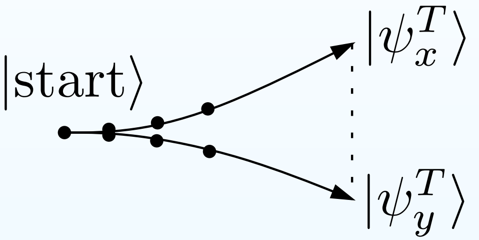
- if an algorithm computes f, then it must *distinguish* between x, y such that $f(x) \neq f(y)$
- computation starts in a fixed state and it has to diverge far enough after T queries for each such x, y



- prove that one query cannot change the scalar product too much (for an average x, y) $\Longrightarrow$ lower bound on T

Adversary lower bounds

- if an algorithm computes f, then it must *distinguish* between x, y such that $f(x) \neq f(y)$
- computation starts in a fixed state and it has to diverge far enough after T queries for each such x, y



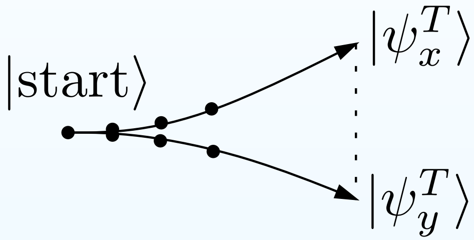
- prove that one query cannot change the scalar product too much (for an average x, y) $\Longrightarrow$ lower bound on T

Limitations

1. weak bounds for exponentially small success probability

Adversary lower bounds

- if an algorithm computes f, then it must *distinguish* between x, y such that $f(x) \neq f(y)$
- computation starts in a fixed state and it has to diverge far enough after T queries for each such x, y



- prove that one query cannot change the scalar product too much (for an average x, y) $\Longrightarrow$ lower bound on T

Limitations

1. weak bounds for exponentially small success probability
2. [Š & Szegedy, Zhang, 2004]
   bounds limited by $\sqrt{C_0 C_1}$ for total functions
   C_z is the z-certificate complexity of f

Our new lower-bound method

- does not suffer from the 1^{st} limitation
  and *possibly* not even from the 2^{nd}

Our new lower-bound method

- does not suffer from the 1^{st} limitation and *possibly* not even from the 2^{nd}
- extends the adversary method above by taking into account the *current knowledge* of the algorithm at each step (the adversary method is oblivious to this and its bound is the same for each query)

Our new lower-bound method

- does not suffer from the 1^{st} limitation and *possibly* not even from the 2^{nd}
- extends the adversary method above by taking into account the *current knowledge* of the algorithm at each step (the adversary method is oblivious to this and its bound is the same for each query)
- based on subspace analysis of the density matrix

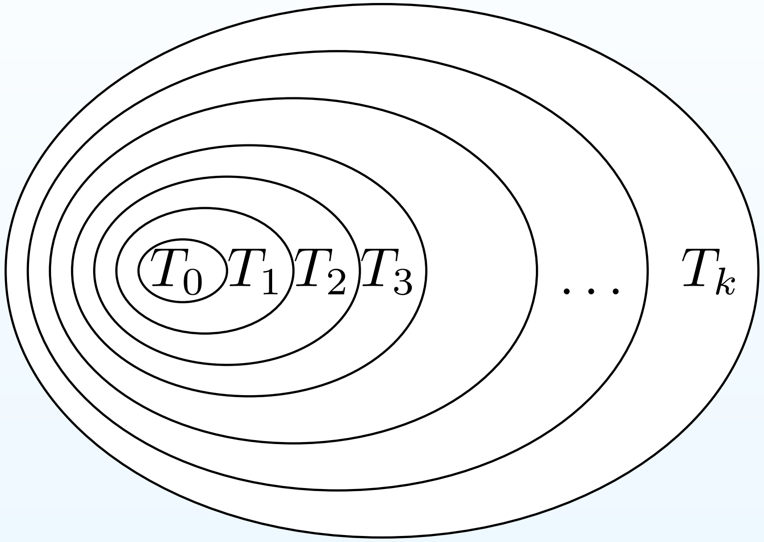
Our new lower-bound method

- does not suffer from the 1$^{\mathrm{st}}$ limitation and *possibly* not even from the 2$^{\mathrm{nd}}$
- extends the adversary method above by taking into account the *current knowledge* of the algorithm at each step (the adversary method is oblivious to this and its bound is the same for each query)
- based on subspace analysis of the density matrix

Applications

$$\left.\begin{array}{l} \bullet\ k\text{-fold search (find } k \text{ ones)} \\ \bullet\ \text{direct product theorems} \\ \bullet\ \text{time-space tradeoffs} \end{array}\right\} \text{explained in a moment}$$

Subspaces for k-fold search



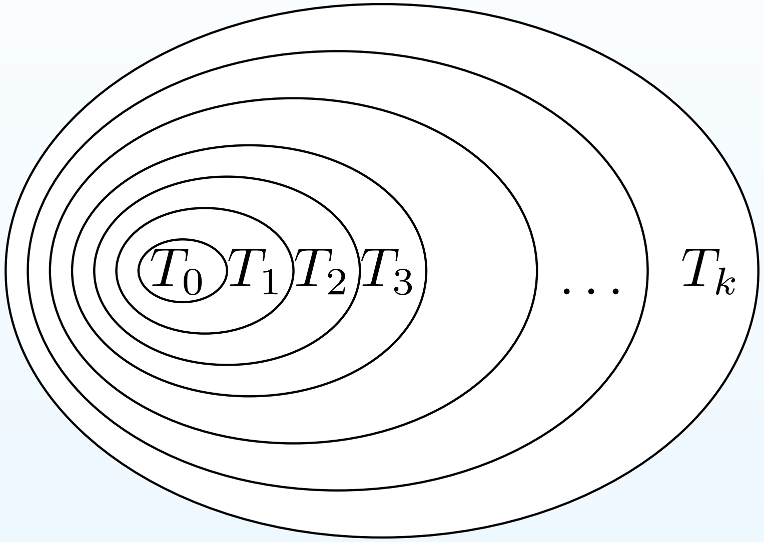
$$T_0 \subseteq T_1 \subseteq \cdots \subseteq T_k$$

T_j "know" at most j ones

spanned by

$$|\psi_{i_1 \dots i_j}\rangle = \sum_{\substack{x:|x|=k \\ x_{i_1}=\cdots=x_{i_j}=1}} |x\rangle$$

Subspaces for k-fold search



$$T_0 \subseteq T_1 \subseteq \cdots \subseteq T_k$$

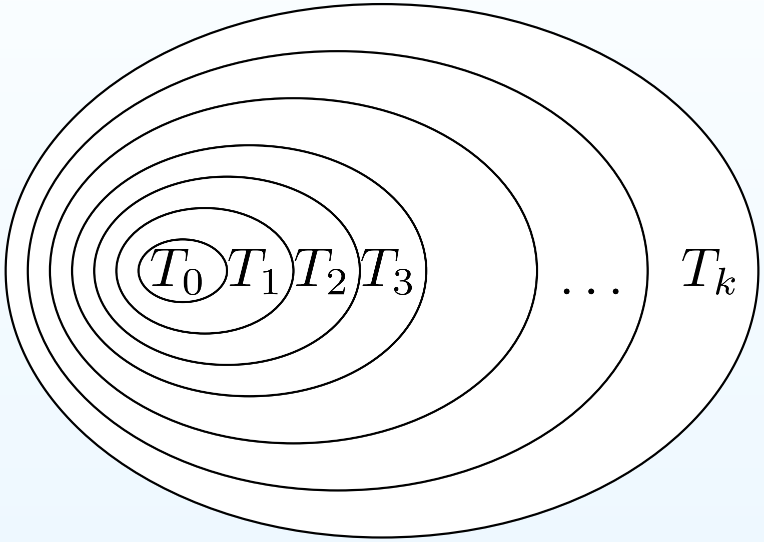
T_j “know” at most j ones

spanned by

$$|\psi_{i_1\dots i_j}\rangle = \sum_{\substack{x:|x|=k \\ x_{i_1}=\dots=x_{i_j}=1}} |x\rangle$$

T_0 starting state

Subspaces for k-fold search



$$T_0 \subseteq T_1 \subseteq \cdots \subseteq T_k$$

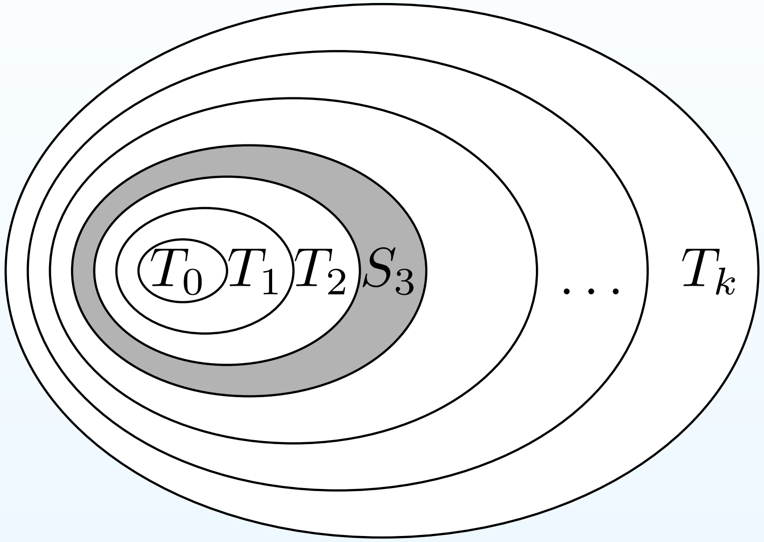
T_j "know" at most j ones
spanned by

$$|\psi_{i_1 \dots i_j}\rangle = \sum_{\substack{x : |x| = k \\ x_{i_1} = \dots = x_{i_j} = 1}} |x\rangle$$

T_0 starting state

T_k entire input space

Subspaces for k-fold search



$$T_0 \subseteq T_1 \subseteq \cdots \subseteq T_k$$

T_j "know" at most j ones

spanned by

$$|\psi_{i_1 \ldots i_j}\rangle = \sum_{\substack{x : |x| = k \\ x_{i_1} = \cdots = x_{i_j} = 1}} |x\rangle$$

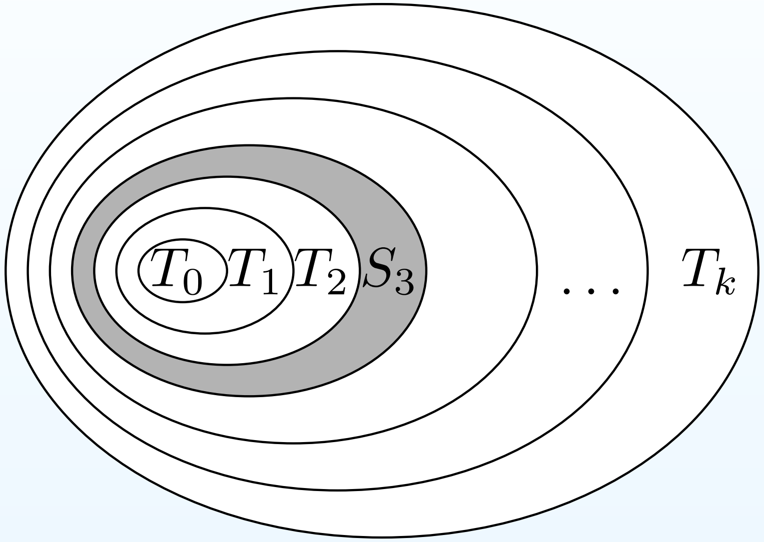
T_0 starting state

T_k entire input space

S_j "know" exactly j ones

$$S_j = T_j \cap T_{j-1}^{\perp}$$

Subspaces for k-fold search



$$T_0 \subseteq T_1 \subseteq \cdots \subseteq T_k$$

T_j "know" at most j ones

spanned by

$$|\psi_{i_1\dots i_j}\rangle = \sum_{\substack{x:|x|=k \\ x_{i_1}=\dots=x_{i_j}=1}} |x\rangle$$

T_0 starting state

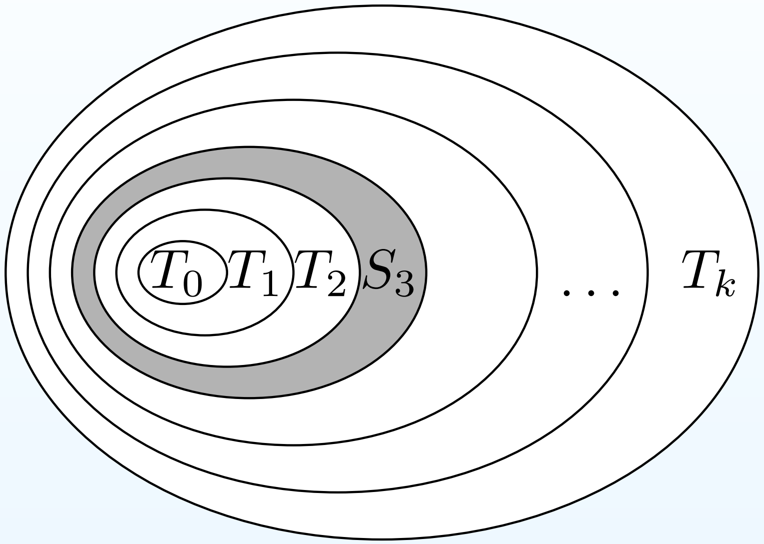
T_k entire input space

S_j "know" exactly j ones

$$S_j = T_j \cap T_{j-1}^{\perp}$$

- in the beginning, all amplitude is in $T_0 = S_0$

Subspaces for k-fold search



$$T_0 \subseteq T_1 \subseteq \cdots \subseteq T_k$$

T_j "know" at most j ones

spanned by

$$|\psi_{i_1 \dots i_j}\rangle = \sum_{\substack{x:|x|=k \\ x_{i_1}=\dots=x_{i_j}=1}} |x\rangle$$

T_0 starting state

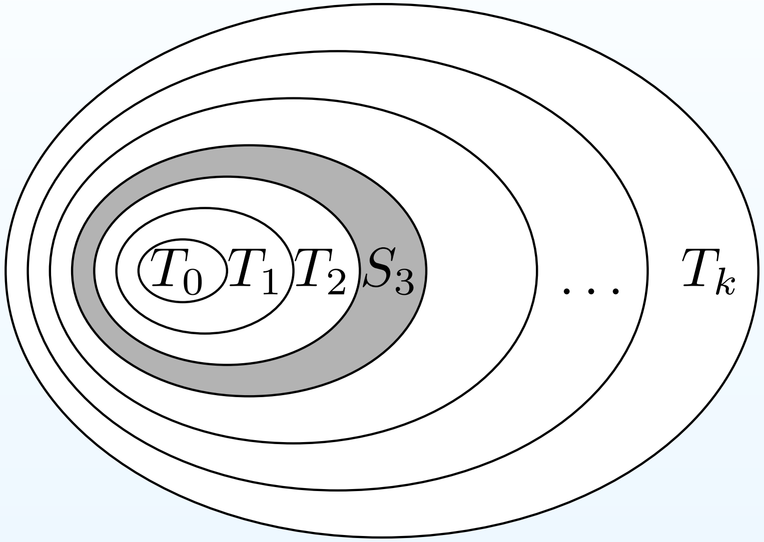
T_k entire input space

S_j "know" exactly j ones

$$S_j = T_j \cap T_{j-1}^{\perp}$$

- in the beginning, all amplitude is in $T_0 = S_0$
- 1 query moves $\leq \sqrt{\frac{k}{n}}$-fraction of amplitude from S_j to S_{j+1}

Subspaces for k-fold search



$$T_0 \subseteq T_1 \subseteq \cdots \subseteq T_k$$

T_j "know" at most j ones

spanned by

$$|\psi_{i_1 \ldots i_j}\rangle = \sum_{\substack{x:|x|=k \\ x_{i_1}=\cdots=x_{i_j}=1}} |x\rangle$$

T_0 starting state

T_k entire input space

S_j "know" exactly j ones

$$S_j = T_j \cap T_{j-1}^{\perp}$$

- in the beginning, all amplitude is in $T_0 = S_0$
- 1 query moves $\leq \sqrt{\frac{k}{n}}$-fraction of amplitude from S_j to S_{j+1}
- to succeed, much amplitude has to be in higher subspaces

Direct Product Theorems

Direct Product Theorems

- We need $Q_\varepsilon(f)$ queries to compute f with error ε. How hard is it to compute k independent instances $f(x_1), \ldots, f(x_k)$?

Direct Product Theorems

- We need $Q_\varepsilon(f)$ queries to compute f with error ε. How hard is it to compute k independent instances $f(x_1), \ldots, f(x_k)$?
- Relation between total number of queries and overall success probability

DPT: $Q_\varepsilon(f^{(k)}) = \Omega(k \cdot Q_{\frac{1}{3}}(f))$ for $\varepsilon = 1 - 2^{-O(k)}$

Easy to prove for $\varepsilon = \frac{1}{3}$, hard for ε close to 1

Direct Product Theorems

- We need $Q_\varepsilon(f)$ queries to compute f with error ε.
  How hard is it to compute k independent instances
  $f(x_1), \ldots, f(x_k)$?
- Relation between total number of queries
  and overall success probability

  DPT: $Q_\varepsilon(f^{(k)}) = \Omega(k \cdot Q_{\frac{1}{3}}(f))$ for $\varepsilon = 1 - 2^{-O(k)}$

  Easy to prove for $\varepsilon = \frac{1}{3}$, hard for ε close to 1
- It is not known whether the DPT holds in general!
  There are counter-examples for average-case complexity

Direct Product Theorems

- We need $Q_\varepsilon(f)$ queries to compute f with error ε.
  How hard is it to compute k independent instances $f(x_1), \ldots, f(x_k)$?
- Relation between total number of queries and overall success probability

  DPT: $Q_\varepsilon(f^{(k)}) = \Omega(k \cdot Q_{\frac{1}{3}}(f))$ for $\varepsilon = 1 - 2^{-O(k)}$

  Easy to prove for $\varepsilon = \frac{1}{3}$, hard for ε close to 1
- It is not known whether the DPT holds in general!
  There are counter-examples for average-case complexity
- [Klauck, Š, de Wolf, FOCS 2004]
  Tight quantum DPT for OR using the polynomial method

Quantum DPT for Symmetric Functions

- *Symmetric* function f depends only on the number of ones

Quantum DPT for Symmetric Functions

- *Symmetric* function f depends only on the number of ones
- Tight quantum DPT for all symmetric functions

$$Q_\varepsilon(f^{(k)}) = \Omega(k \cdot Q_{\frac{1}{3}}(f)) \qquad \text{for } \varepsilon = 1 - 2^{-O(k)}$$

using our new lower-bound method

Quantum DPT for Symmetric Functions

- *Symmetric* function f depends only on the number of ones
- Tight quantum DPT for all symmetric functions

$$Q_\varepsilon(f^{(k)}) = \Omega(k \cdot Q_{\frac{1}{3}}(f)) \qquad \text{for } \varepsilon = 1 - 2^{-O(k)}$$

  using our new lower-bound method

- Classically, the DPT was already known [KŠW'04]

Quantum DPT for Symmetric Functions

- *Symmetric* function f depends only on the number of ones
- Tight quantum DPT for all symmetric functions

$$Q_\varepsilon(f^{(k)}) = \Omega(k \cdot Q_{\frac{1}{3}}(f)) \qquad \text{for } \varepsilon = 1 - 2^{-O(k)}$$

  using our new lower-bound method

- Classically, the DPT was already known [KŠW'04]
- Tight 1-sided quantum DPT for t-*threshold* functions

$$Q_\varepsilon(f^{(k)}) = \Omega(k \cdot Q_{\frac{1}{3}}(f)) \qquad \text{for } \varepsilon = 1 - 2^{-O(k \cdot t)}$$

  using the polynomial method

Time-Space Tradeoffs

Time-Space Tradeoffs

- A relation between the running time and space complexity

The more memory is available,
the faster the algorithm could possibly run.

Time-Space Tradeoffs

- A relation between the running time and space complexity

The more memory is available,
the faster the algorithm could possibly run.

- Example: sorting of N numbers
  - Classically

$$TS = \Theta(N^2)$$

Time-Space Tradeoffs

- A relation between the running time and space complexity

The more memory is available,
the faster the algorithm could possibly run.

- Example: sorting of N numbers
  - Classically
  $$TS = \Theta(N^2)$$
  - Quantumly
  $$T^2 S = \tilde{\Theta}(N^3)$$
  using the DPT for OR [KŠW'04]

Evaluating Solutions to Systems of Linear Inequalities

- A fixed $N \times N$ zero-one matrix
  x non-negative integer input vector of length N

  The task is to determine which inequalities are true

$$Ax \geq (t, \ldots, t)$$

Evaluating Solutions to Systems of Linear Inequalities

- A fixed $N \times N$ zero-one matrix
  x non-negative integer input vector of length N

  The task is to determine which inequalities are true

$$Ax \geq (t, \ldots, t)$$

- We study the query complexity with bounded error

  Classically $TS = \tilde{\Theta}(N^2)$

Evaluating Solutions to Systems of Linear Inequalities

- A fixed $N \times N$ zero-one matrix
  x non-negative integer input vector of length N

  The task is to determine which inequalities are true

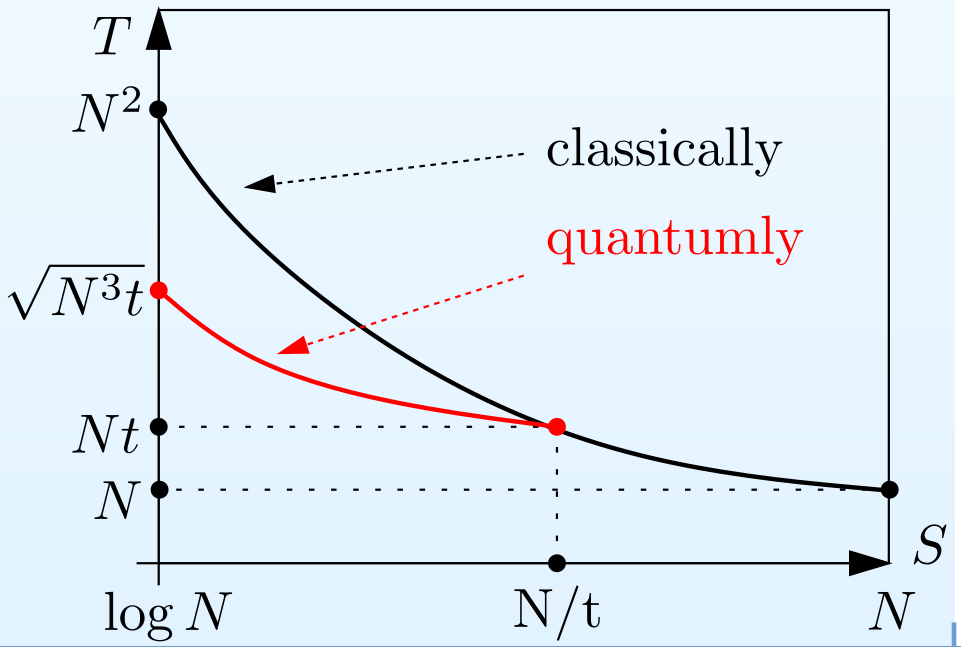
$$Ax \geq (t, \ldots, t)$$

- We study the query complexity with bounded error

  Classically $TS = \tilde{\Theta}(N^2)$

  Quantumly

$$T^2 S = \tilde{\Theta}(N^3 t) \quad S \leq \frac{N}{t}$$
$$TS = \tilde{\Theta}(N^2) \quad S > \frac{N}{t}$$



Time-Space Tradeoff for Linear Inequalities

Quantum algorithm uses

- Grover search
- quantum counting

to find non-zero inputs faster than classically

Time-Space Tradeoff for Linear Inequalities

Quantum algorithm uses

- Grover search
- quantum counting

to find non-zero inputs faster than classically

Matching lower bound proved as follows

- Fix a *hard matrix* A.

Time-Space Tradeoff for Linear Inequalities

Quantum algorithm uses

- Grover search
- quantum counting

to find non-zero inputs faster than classically

Matching lower bound proved as follows

- Fix a *hard matrix* A.
- Slice the circuit. Deciding k inequalities in one slice allows computing k *non-overlapping* threshold functions.

Time-Space Tradeoff for Linear Inequalities

Quantum algorithm uses

- Grover search
- quantum counting

to find non-zero inputs faster than classically

Matching lower bound proved as follows

- Fix a *hard matrix* A.
- Slice the circuit. Deciding k inequalities in one slice allows computing k *non-overlapping* threshold functions.
- Replace (unknown) starting state by completely mixed state. Success probability goes down to 2^{-S}.

Time-Space Tradeoff for Linear Inequalities

Quantum algorithm uses

- Grover search
- quantum counting

to find non-zero inputs faster than classically

Matching lower bound proved as follows

- Fix a *hard matrix* A.
- Slice the circuit. Deciding k inequalities in one slice allows computing k *non-overlapping* threshold functions.
- Replace (unknown) starting state by completely mixed state. Success probability goes down to 2^{-S}.
- By DPT, we still need many queries in each slice.

Time-Space Tradeoff for Linear Inequalities

Quantum algorithm uses

- Grover search
- quantum counting

to find non-zero inputs faster than classically

Matching lower bound proved as follows

- Fix a *hard matrix* A.
- Slice the circuit. Deciding k inequalities in one slice allows computing k *non-overlapping* threshold functions.
- Replace (unknown) starting state by completely mixed state. Success probability goes down to 2^{-S}.
- By DPT, we still need many queries in each slice.

$\Longrightarrow$ (tight) lower bound on T as a function of S

Summary and open problems

- a new quantum lower bound method
  based on analysis of subspaces of the density matrix
- tight quantum direct product theorem for all symmetric functions
- optimal time-space tradeoff for evaluating solutions to systems of linear inequalities
  - with small space, quantum computers are faster
  - with large space, classical are as good as quantum

Summary and open problems

- a new quantum lower bound method
  based on analysis of subspaces of the density matrix
- tight quantum direct product theorem for all symmetric functions
- optimal time-space tradeoff for evaluating solutions to systems of linear inequalities
  - with small space, quantum computers are faster
  - with large space, classical are as good as quantum

Open problems

- binary AND-OR tree: $O(n^{0.753})$, $\Omega(\sqrt{n})$
- triangle finding: $O(n^{1.3})$, $\Omega(n)$
- verification of matrix products: $O(n^{5/3})$, $\Omega(n^{3/2})$