Mathematical Challenges in Quantum Algorithms

Ashley Montanaro

Department of Computer Science,
University of Bristol

18 September 2014

Introduction

There are many things we can do with our quantum computers. For example:

- Factorise large integers and hence break RSA;
- Efficiently simulate quantum-mechanical systems;
- Solve certain search and optimisation problems faster than possible classically;
- …

See the Quantum Algorithm Zoo (`http://math.nist.gov/quantum/zoo/`) for ~~214~~ 219 papers on quantum algorithms.

Introduction

Nevertheless, many embarrassingly fundamental open problems remain in the study of quantum computing:

- What we can do;
- What we can't do;
- Why we can do what we can.

Introduction

Nevertheless, many embarrassingly fundamental open problems remain in the study of quantum computing:

- What we can do;
- What we can't do;
- Why we can do what we can.

In this talk I will discuss a personal selection of a few of these open problems. Notably, they (mostly) rest on purely classical mathematical questions!

Hidden subgroup problems

Hidden subgroup problem (e.g. [Boneh and Lipton '95])

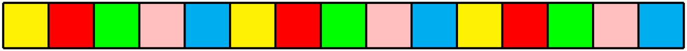
Let G be a group. Given oracle access to a function $f : G \to X$ such that f is constant on the cosets of some subgroup $H \leqslant G$, and distinct on each coset, identify H.

Hidden subgroup problems

Hidden subgroup problem (e.g. [Boneh and Lipton '95])

Let G be a group. Given oracle access to a function $f : G \to X$ such that f is constant on the cosets of some subgroup $H \leqslant G$, and distinct on each coset, identify H.

Integer factorisation reduces to the case $G = \mathbb{Z}_M$ for some integer M. This is the problem of determining the period of a periodic function:

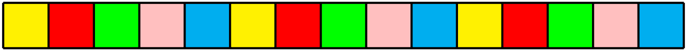


Hidden subgroup problems

Hidden subgroup problem (e.g. [Boneh and Lipton '95])

Let G be a group. Given oracle access to a function $f : G \to X$ such that f is constant on the cosets of some subgroup $H \leqslant G$, and distinct on each coset, identify H.

Integer factorisation reduces to the case $G = \mathbb{Z}_M$ for some integer M. This is the problem of determining the period of a periodic function:



On a quantum computer, the HSP can be solved using $O(\log |G|)$ queries to f for all groups G [Ettinger et al. '04]. Classically, some groups require $\Omega(\sqrt{|G|})$ queries [Simon '97].

Hidden subgroup problems

Open problem

For which groups G can the HSP be solved efficiently?

Hidden subgroup problems

Open problem

For which groups G can the HSP be solved efficiently?

The HSP is related to many other problems and cryptosystems:

Problem	**Group**	**Complexity**	**Cryptosystem**
Factorisation	$\mathbb{Z}_N$	Polynomial[1]	RSA
Discrete log	$\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$	Polynomial[1]	Diffie-Hellman, DSA, . . .
Elliptic curve d. log	Elliptic curve	Polynomial[2]	ECDH, ECDSA, . . .
Principal ideal	$\mathbb{R}$	Polynomial[3]	Buchmann-Williams
Shortest lattice vector	Dihedral grp	Subexp.[4]	NTRU, Ajtai-Dwork, . . .
Graph isomorphism	Symmetric grp	Exponential	–

[1] Shor ’97, [2] Proos et al. ’03, [3] Hallgren ’07, [4] Kuperberg ’05, Regev ’04

Hidden subgroup problems

Open problem

For which groups G can the HSP be solved efficiently?

The HSP is related to many other problems and cryptosystems:

Problem	**Group**	**Complexity**	**Cryptosystem**
Factorisation	$\mathbb{Z}_N$	Polynomial[1]	RSA
Discrete log	$\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$	Polynomial[1]	Diffie-Hellman, DSA, ...
Elliptic curve d. log	Elliptic curve	Polynomial[2]	ECDH, ECDSA, ...
Principal ideal	$\mathbb{R}$	Polynomial[3]	Buchmann-Williams
Shortest lattice vector	Dihedral grp	Subexp.[4]	NTRU, Ajtai-Dwork, ...
Graph isomorphism	Symmetric grp	Exponential	—

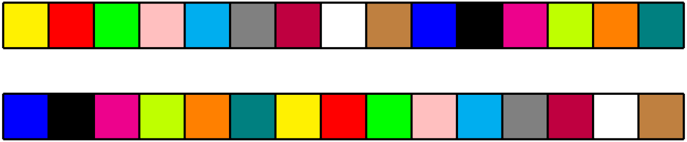
[1]Shor '97, [2]Proos et al. '03, [3]Hallgren '07, [4]Kuperberg '05, Regev '04

A significant amount of other work on the HSP has resolved its complexity for many other groups.

The dihedral hidden subgroup problem

The dihedral HSP turns out to be equivalent to a hidden shift problem:

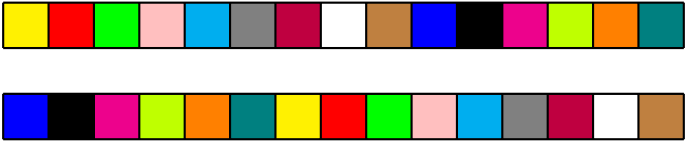
- Given two injective functions $f, g : \mathbb{Z}_N \to X$ such that $g(x) = f(x+s)$ for some $s \in \mathbb{Z}_N$, determine s.



The dihedral hidden subgroup problem

The dihedral HSP turns out to be equivalent to a hidden shift problem:

- Given two injective functions $f, g : \mathbb{Z}_N \to X$ such that $g(x) = f(x+s)$ for some $s \in \mathbb{Z}_N$, determine s.

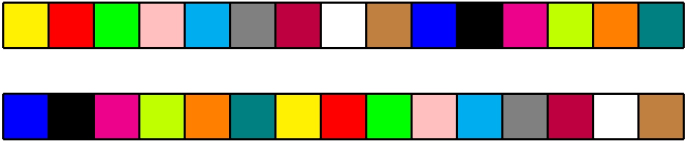


Implies applications to pattern matching problems in strings.

The dihedral hidden subgroup problem

The dihedral HSP turns out to be equivalent to a hidden shift problem:

- Given two injective functions $f, g : \mathbb{Z}_N \to X$ such that $g(x) = f(x+s)$ for some $s \in \mathbb{Z}_N$, determine s.



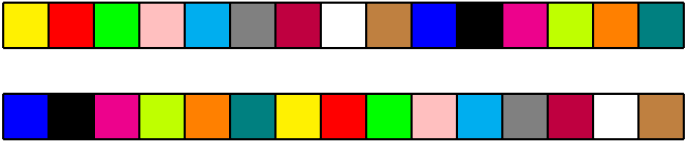
Implies applications to pattern matching problems in strings.

- The best known algorithm for the dihedral HSP uses time $2^{O(\sqrt{\log N})}$ [Kuperberg '05] …can this be improved?

The dihedral hidden subgroup problem

The dihedral HSP turns out to be equivalent to a hidden shift problem:

- Given two injective functions $f, g : \mathbb{Z}_N \to X$ such that $g(x) = f(x+s)$ for some $s \in \mathbb{Z}_N$, determine s.



Implies applications to pattern matching problems in strings.

- The best known algorithm for the dihedral HSP uses time $2^{O(\sqrt{\log N})}$ [Kuperberg '05] …can this be improved?
- A $\mathrm{poly}(\log N)$-time algorithm would give an efficient quantum algorithm for the shortest vector problem in lattices [Regev '04].

Solving the HSP via the Kuperberg sieve

- One approach to solving the dihedral HSP starts by producing many quantum states of the form

  $$|\psi_x\rangle := |0\rangle + e^{2\pi i s x/N}|1\rangle,$$

  where $x \in \{0, \ldots, N-1\}$ is uniformly random.

Solving the HSP via the Kuperberg sieve

- One approach to solving the dihedral HSP starts by producing many quantum states of the form

  $$|\psi_x\rangle := |0\rangle + e^{2\pi i s x/N}|1\rangle,$$

  where $x \in \{0, \ldots, N-1\}$ is uniformly random.

- We would like to make the state $\left|\psi_{N/2}\right\rangle = |0\rangle + (-1)^s|1\rangle$, which is sufficient to determine one bit of s.

Solving the HSP via the Kuperberg sieve

- One approach to solving the dihedral HSP starts by producing many quantum states of the form

$$|\psi_x\rangle := |0\rangle + e^{2\pi i s x/N}|1\rangle,$$

  where $x \in \{0, \ldots, N-1\}$ is uniformly random.

- We would like to make the state $|\psi_{N/2}\rangle = |0\rangle + (-1)^s|1\rangle$, which is sufficient to determine one bit of s.

- One way to do this is via the following combination operation:

$$C(|\psi_x\rangle, |\psi_y\rangle) = \begin{cases} |\psi_{x+y}\rangle & \text{with prob. } 1/2 \\ |\psi_{x-y}\rangle & \text{with prob. } 1/2 \end{cases}$$

Solving the HSP via the Kuperberg sieve

Theorem **[Kuperberg '05, AM '14]**

It is sufficient to start with $2^{1.781\ldots\sqrt{\log_2 N}}\,\mathrm{poly}(\log N)$ random states $|\psi_x\rangle$ to be able to produce a state of the form $\left|\psi_{N/2}\right\rangle$ with high probability using combination operations.

- This gives us a subexponential-time, but superpolynomial-time, algorithm.

Solving the HSP via the Kuperberg sieve

Theorem **[Kuperberg ’05, AM ’14]**

It is sufficient to start with $2^{1.781\ldots\sqrt{\log_2 N}}\,\mathrm{poly}(\log N)$ random states $|\psi_x\rangle$ to be able to produce a state of the form $|\psi_{N/2}\rangle$ with high probability using combination operations.

- This gives us a subexponential-time, but superpolynomial-time, algorithm.

Open problem

Can this be improved?

- Solving certain average-case subset sum problems efficiently would also give us an efficient solution to this problem [Regev ’04].

Quantum query complexity

- The query complexity of a boolean function $f : \{0,1\}^n \to \{0,1\}$ is the number of queries to its input bits that are required to compute it (e.g. with bounded error).

Quantum query complexity

- The query complexity of a boolean function $f : \{0,1\}^n \to \{0,1\}$ is the number of queries to its input bits that are required to compute it (e.g. with bounded error).
- For example, the OR_n function ($f(x) = 0 \Leftrightarrow x = 0^n$) has classical query complexity $\Theta(n)$, and quantum query complexity $\Theta(\sqrt{n})$ [Grover '97].

Quantum query complexity

- The query complexity of a boolean function $f : \{0,1\}^n \to \{0,1\}$ is the number of queries to its input bits that are required to compute it (e.g. with bounded error).
- For example, the OR_n function ($f(x) = 0 \Leftrightarrow x = 0^n$) has classical query complexity $\Theta(n)$, and quantum query complexity $\Theta(\sqrt{n})$ [Grover '97].
- For problems where there is a promise on the input (like the HSP), the separation between quantum and classical query complexity can be exponential.

Quantum query complexity

- The query complexity of a boolean function $f : \{0,1\}^n \to \{0,1\}$ is the number of queries to its input bits that are required to compute it (e.g. with bounded error).
- For example, the OR_n function ($f(x) = 0 \Leftrightarrow x = 0^n$) has classical query complexity $\Theta(n)$, and quantum query complexity $\Theta(\sqrt{n})$ [Grover '97].
- For problems where there is a promise on the input (like the HSP), the separation between quantum and classical query complexity can be exponential.

Open problem

What is the largest possible separation between quantum and classical query complexity for a total function?

Quantum query complexity

- If $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is a total function, the separation can be at most a 6th power [Beals '01].

Quantum query complexity

- If $f : \{0,1\}^n \rightarrow \{0,1\}$ is a total function, the separation can be at most a 6th power [Beals '01]. Conjecture: The 6 can be replaced with a 2.

Quantum query complexity

- If $f : \{0,1\}^n \to \{0,1\}$ is a total function, the separation can be at most a 6th power [Beals '01]. Conjecture: The 6 can be replaced with a 2.
- To improve it to a 4, it would suffice to prove that

$$\deg(f) = O(\widetilde{\deg}(f))^2.$$

Quantum query complexity

- If $f : \{0,1\}^n \to \{0,1\}$ is a total function, the separation can be at most a 6th power [Beals '01]. Conjecture: The 6 can be replaced with a 2.
- To improve it to a 4, it would suffice to prove that

$$\deg(f) = O(\widetilde{\deg}(f))^2.$$

What are these quantities?

- $\deg(f)$ is the degree of f as an n-variate polynomial over $\mathbb{R}$.
- $\widetilde{\deg}(f)$ is the approximate degree: i.e. the smallest degree of any polynomial $\widetilde{f}$ such that $|\widetilde{f}(x) - f(x)| \leqslant 1/3$ for all x.

Quantum query complexity

- If $f : \{0,1\}^n \to \{0,1\}$ is a total function, the separation can be at most a 6th power [Beals '01]. Conjecture: The 6 can be replaced with a 2.
- To improve it to a 4, it would suffice to prove that

$$\deg(f) = O(\widetilde{\deg}(f))^2.$$

What are these quantities?

- $\deg(f)$ is the degree of f as an n-variate polynomial over $\mathbb{R}$.
- $\widetilde{\deg}(f)$ is the approximate degree: i.e. the smallest degree of any polynomial $\widetilde{f}$ such that $|\widetilde{f}(x) - f(x)| \leqslant 1/3$ for all x.

For example:

$$\deg(\mathrm{OR}_2) = 2 \qquad \mathrm{OR}_2(x) = x_1 + x_2 - x_1 x_2$$

$$\widetilde{\deg}(\mathrm{OR}_2) = 1 \qquad \text{e.g. } \widetilde{\mathrm{OR}_2}(x) = (x_1 + x_2)/3$$

Degree and other complexity measures

How does this imply the claim about quantum query complexity?

Degree and other complexity measures

How does this imply the claim about quantum query complexity?

Let $Q(f)$, $D(f)$ denote the quantum and classical query complexities of computing f. Then it's known that:

- $Q(f) = \Omega(\widetilde{\deg}(f))$ [Beals et al. '01];
- $Q(f) = \Omega(\sqrt{\mathrm{bs}(f)})$ [Bennett et al. '97];
- $D(f) = O(\deg(f)\,\mathrm{bs}(f))$ [Midrijānis '05].

In the above $\mathrm{bs}(f)$ is the block sensitivity of f.

Degree and other complexity measures

How does this imply the claim about quantum query complexity?

Let $Q(f)$, $D(f)$ denote the quantum and classical query complexities of computing f. Then it's known that:

- $Q(f) = \Omega(\widetilde{\deg}(f))$ [Beals et al. '01];
- $Q(f) = \Omega(\sqrt{\mathrm{bs}(f)})$ [Bennett et al. '97];
- $D(f) = O(\deg(f)\,\mathrm{bs}(f))$ [Midrijānis '05].

In the above $\mathrm{bs}(f)$ is the block sensitivity of f.

Hence we would have $D(f) \overset{?}{=} O(\widetilde{\deg}(f)^2\,\mathrm{bs}(f)) = O(Q(f)^4)$.

Quantum property testing

The field of property testing works in a setting where we would like to determine whether some very large object has a property, or is far away from having that property, by making very few queries to the object.

Quantum property testing

The field of property testing works in a setting where we would like to determine whether some very large object has a property, or is far away from having that property, by making very few queries to the object.

Property testing

- Let X be a set of objects and $d : X \times X \to [0, 1]$ be a distance measure on X.
- A subset $\mathcal{P} \subseteq X$ is called a property.
- An object $x \in X$ is ϵ-far from $\mathcal{P}$ if $d(x, y) \geqslant \epsilon$ for all $y \in \mathcal{P}$;
- x is ϵ-close to $\mathcal{P}$ if there is a $y \in \mathcal{P}$ such that $d(x, y) \leqslant \epsilon$.

Quantum property testing

The field of property testing works in a setting where we would like to determine whether some very large object has a property, or is far away from having that property, by making very few queries to the object.

Property testing

- Let X be a set of objects and $d : X \times X \rightarrow [0, 1]$ be a distance measure on X.
- A subset $\mathcal{P} \subseteq X$ is called a property.
- An object $x \in X$ is ϵ-far from $\mathcal{P}$ if $d(x, y) \geqslant \epsilon$ for all $y \in \mathcal{P}$;
- x is ϵ-close to $\mathcal{P}$ if there is a $y \in \mathcal{P}$ such that $d(x, y) \leqslant \epsilon$.

An ϵ-property tester for $\mathcal{P}$ is an algorithm that receives as input either an $x \in \mathcal{P}$ or an x that is ϵ-far from $\mathcal{P}$, and that distinguishes these two cases with success probability at least $2/3$.

Quantum property testing

In some cases, quantum property testers can significantly outperform their classical counterparts. For example:

- An exponential speedup for testing whether a sequence of N integers is periodic (i.e. $\mathrm{poly}(\log N)$ vs. $\Omega(N^{1/4})$ queries) [Chakraborty et al. '10];
- Polynomial speedups for testing some properties of graphs: e.g. bipartiteness, expansion ($\widetilde{O}(N^{1/3})$ vs. $\Omega(N^{1/2})$ queries in both cases) [Ambainis et al. '11];
- …

Quantum property testing

In some cases, quantum property testers can significantly outperform their classical counterparts. For example:

- An exponential speedup for testing whether a sequence of N integers is periodic (i.e. $\mathrm{poly}(\log N)$ vs. $\Omega(N^{1/4})$ queries) [Chakraborty et al. '10];
- Polynomial speedups for testing some properties of graphs: e.g. bipartiteness, expansion ($\widetilde{O}(N^{1/3})$ vs. $\Omega(N^{1/2})$ queries in both cases) [Ambainis et al. '11];
- …

However, most known quantum property-testing algorithms are based around taking an existing quantum algorithm and adapting it for the property-testing setting.

Quantum property testing

Open problem

Could there be an exponential quantum speedup for testing a graph property?

- A graph property is simply a subset of the set of all adjacency matrices which is invariant under relabelling the graph vertices.
- Examples include bipartiteness, planarity, 3-colourability, connectivity …
- No super-polynomial speedup is currently known.

Quantum property testing

Why is this interesting?

Quantum property testing

Why is this interesting?

- It is known that there can be no exponential quantum speedup for computing functions that are "too symmetric" [Aaronson and Ambainis '11], i.e. that are invariant under any permutation of their inputs.

Quantum property testing

Why is this interesting?

- It is known that there can be no exponential quantum speedup for computing functions that are "too symmetric" [Aaronson and Ambainis '11], i.e. that are invariant under any permutation of their inputs.
- Graph properties possess an intermediate level of symmetry.
- So it seems that proving that an exponential speedup can, or cannot, exist would throw light on the role of symmetry in quantum algorithms.

Quantum property testing

Why is this interesting?

- It is known that there can be no exponential quantum speedup for computing functions that are “too symmetric” [Aaronson and Ambainis ’11], i.e. that are invariant under any permutation of their inputs.
- Graph properties possess an intermediate level of symmetry.
- So it seems that proving that an exponential speedup can, or cannot, exist would throw light on the role of symmetry in quantum algorithms.
- Also, classically the graph properties that are efficiently testable have been completely characterised [Alon et al. ’09]. Can we use this characterisation quantumly?

Summary and further reading

Although there is much known about quantum algorithms and quantum computational complexity, there are still many tantalising open problems.

Summary and further reading

Although there is much known about quantum algorithms and quantum computational complexity, there are still many tantalising open problems.

Some of these problems require very little background in quantum computing itself to solve.

Summary and further reading

Although there is much known about quantum algorithms and quantum computational complexity, there are still many tantalising open problems.

Some of these problems require very little background in quantum computing itself to solve.

Some further reading:

- "Quantum algorithms for algebraic problems" [Childs and van Dam '08]
- "Quantum algorithms" [Mosca '08]
- "New developments in quantum algorithms" [Ambainis '10]
- "A survey of quantum property testing" [AM and de Wolf '13]

Summary and further reading

Although there is much known about quantum algorithms and quantum computational complexity, there are still many tantalising open problems.

Some of these problems require very little background in quantum computing itself to solve.

Some further reading:

- “Quantum algorithms for algebraic problems” [Childs and van Dam ’08]
- “Quantum algorithms” [Mosca ’08]
- “New developments in quantum algorithms” [Ambainis ’10]
- “A survey of quantum property testing” [AM and de Wolf ’13]

Thanks!

Average-case simulation of quantum algorithms

Conjecture A **[Aaronson and Ambainis '09, slightly modified]**

Let Q be a quantum algorithm which makes T queries to x. Then, for any $\epsilon > 0$, there is a classical algorithm which makes $\mathrm{poly}(T, 1/\epsilon, 1/\delta)$ queries to x, and approximates Q's success probability to within $\pm\epsilon$ on a $1-\delta$ fraction of inputs.

Average-case simulation of quantum algorithms

Conjecture A **[Aaronson and Ambainis '09, slightly modified]**

Let Q be a quantum algorithm which makes T queries to x. Then, for any $\epsilon > 0$, there is a classical algorithm which makes $\mathrm{poly}(T, 1/\epsilon, 1/\delta)$ queries to x, and approximates Q's success probability to within $\pm\epsilon$ on a $1 - \delta$ fraction of inputs.

- Given known results, essentially the strongest conjecture one could make about classical simulation of quantum query algorithms.

Average-case simulation of quantum algorithms

Conjecture A **[Aaronson and Ambainis '09, slightly modified]**

Let Q be a quantum algorithm which makes T queries to x. Then, for any $\epsilon > 0$, there is a classical algorithm which makes $\text{poly}(T, 1/\epsilon, 1/\delta)$ queries to x, and approximates Q's success probability to within $\pm\epsilon$ on a $1-\delta$ fraction of inputs.

- Given known results, essentially the strongest conjecture one could make about classical simulation of quantum query algorithms.
- Aaronson and Ambainis show that Conjecture A follows from the following, more mathematical conjecture...

Influences of variables on low-degree polynomials

Conjecture B **[Aaronson and Ambainis '09]**

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a degree d multivariate polynomial such that $0 \leqslant f(x) \leqslant 1$ for all $x \in \{\pm 1\}^n$ and $\mathsf{Var}(f) \geqslant \epsilon$. Then there exists $j \in \{1, \ldots, n\}$ such that

$$\mathrm{Inf}_j(f) \geqslant \mathrm{poly}(\epsilon/d).$$

Influences of variables on low-degree polynomials

Conjecture B **[Aaronson and Ambainis '09]**

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a degree d multivariate polynomial such that $0 \leqslant f(x) \leqslant 1$ for all $x \in \{\pm 1\}^n$ and $\mathsf{Var}(f) \geqslant \epsilon$. Then there exists $j \in \{1, \ldots, n\}$ such that

$$\mathrm{Inf}_j(f) \geqslant \mathrm{poly}(\epsilon/d).$$

In this conjecture:

$$\begin{aligned}
\mathsf{Var}(f) &= \mathbb{E}[(f(x) - \mathbb{E}[f])^2] = \frac{1}{2^n} \sum_{x \in \{\pm 1\}^n} \left(f(x) - \frac{1}{2^n} \sum_{y \in \{\pm 1\}^n} f(x) \right)^2 \\
\mathrm{Inf}_j(f) &= \frac{1}{2^{n+2}} \sum_{x \in \{\pm 1\}^n} (f(x) - f(x^j))^2
\end{aligned}$$

Influences of variables on low-degree polynomials

This conjecture has been proven in a couple of special cases:

- If f is symmetric under permutations of the input bits [Bačkurs '12];
- If f is a multilinear form whose coefficients are equal in absolute value [AM '12].

The general case remains open.