Combinatorics in quantum computation, and vice versa

Ashley Montanaro

Department of Computer Science, University of Bristol, UK

11 December 2014

Based on joint work with Andris Ambainis and Tobias Osborne

University of Bristol

Introduction

In this talk, I will discuss three connections between combinatorics and quantum computation:

Introduction

In this talk, I will discuss three connections between combinatorics and quantum computation:

1. A quantum algorithm for pattern matching in strings which achieves a super-polynomial speedup over any possible classical algorithm, for most strings;

Introduction

In this talk, I will discuss three connections between combinatorics and quantum computation:

1. A quantum algorithm for pattern matching in strings which achieves a super-polynomial speedup over any possible classical algorithm, for most strings;
2. A quantum algorithm for a "search with wildcards" problem which achieves a square-root speedup in the worst case (with a cameo appearance from group testing);

Introduction

In this talk, I will discuss three connections between combinatorics and quantum computation:

1. A quantum algorithm for pattern matching in strings which achieves a super-polynomial speedup over any possible classical algorithm, for most strings;

2. A quantum algorithm for a "search with wildcards" problem which achieves a square-root speedup in the worst case (with a cameo appearance from group testing);

3. A conjectured quantum generalisation of the Kahn-Kalai-Linial (KKL) theorem that every boolean function has an influential variable.

The quantum query model in a nutshell

- Imagine we have access to some function $f : X \to Y$ as an oracle or black box:

$$x \mapsto f(x)$$

- We want to determine some property of f (with success probability, say, 2/3).

The quantum query model in a nutshell

- Imagine we have access to some function $f : X \to Y$ as an oracle or black box:

$$x \mapsto f(x)$$

- We want to determine some property of f (with success probability, say, 2/3).
- To do this, we can query f on inputs $x \in X$ to get outputs $y \in Y$. In fact, we assume we have access to a map

$$(x, y) \mapsto (x, y + f(x)).$$

The quantum query model in a nutshell

- Imagine we have access to some function $f : X \to Y$ as an oracle or black box:
$$x \mapsto f(x)$$
- We want to determine some property of f (with success probability, say, 2/3).
- To do this, we can query f on inputs $x \in X$ to get outputs $y \in Y$. In fact, we assume we have access to a map
$$(x, y) \mapsto (x, y + f(x)).$$
- On a quantum computer, we can query f in superposition:
$$\sum_{x \in X, y \in Y} \alpha_{xy} |x\rangle |y\rangle \mapsto \sum_{x \in X, y \in Y} \alpha_{xy} |x\rangle |y + f(x)\rangle.$$

Example: unstructured and structured search

Problem: unstructured search

We are given access to $f : [n] \to \{0, 1\}$. Our task is to output some x such that $f(x) = 1$, if such an x exists.

Example: unstructured and structured search

Problem: unstructured search

We are given access to $f : [n] \to \{0, 1\}$. Our task is to output some x such that $f(x) = 1$, if such an x exists.

- Grover's algorithm [Grover '97] solves this task using $O(\sqrt{n})$ quantum queries. Classically, $\Theta(n)$ queries are required.

Example: unstructured and structured search

Problem: unstructured search

We are given access to $f : [n] \to \{0, 1\}$. Our task is to output some x such that $f(x) = 1$, if such an x exists.

- Grover's algorithm [Grover '97] solves this task using $O(\sqrt{n})$ quantum queries. Classically, $\Theta(n)$ queries are required.

Problem: search of a sorted list

We are given access to $f : [n] \to \{0, 1\}$ such that $x \leqslant y \Rightarrow f(x) \leqslant f(y)$. Our task is to find the minimal x such that $f(x) = 1$, if such an x exists.

Example: unstructured and structured search

Problem: unstructured search

We are given access to $f : [n] \to \{0, 1\}$. Our task is to output some x such that $f(x) = 1$, if such an x exists.

- Grover's algorithm [Grover '97] solves this task using $O(\sqrt{n})$ quantum queries. Classically, $\Theta(n)$ queries are required.

Problem: search of a sorted list

We are given access to $f : [n] \to \{0, 1\}$ such that $x \leqslant y \Rightarrow f(x) \leqslant f(y)$. Our task is to find the minimal x such that $f(x) = 1$, if such an x exists.

- On a classical computer, binary search solves this problem using $\lfloor \log_2 n + 1 \rfloor$ queries.

Example: unstructured and structured search

Problem: unstructured search

We are given access to $f : [n] \to \{0, 1\}$. Our task is to output some x such that $f(x) = 1$, if such an x exists.

- Grover's algorithm [Grover '97] solves this task using $O(\sqrt{n})$ quantum queries. Classically, $\Theta(n)$ queries are required.

Problem: search of a sorted list

We are given access to $f : [n] \to \{0, 1\}$ such that $x \leqslant y \Rightarrow f(x) \leqslant f(y)$. Our task is to find the minimal x such that $f(x) = 1$, if such an x exists.

- On a classical computer, binary search solves this problem using $\lfloor \log_2 n + 1 \rfloor$ queries.
- The quantum query complexity is known to be $\Omega(\log n)$... but the precise constant factor is unknown!

Pattern matching

In the traditional pattern matching problem, we seek to find a pattern $P : [m] \to \Sigma$ within a text $T : [n] \to \Sigma$.

$T =$

Q	U	A	N	T	U	M

$P =$

A	N	T

Pattern matching

In the traditional pattern matching problem, we seek to find a pattern $P : [m] \to \Sigma$ within a text $T : [n] \to \Sigma$.

$T =$ | Q | U | **A** | **N** | **T** | U | M |

$P =$ | A | N | T |

Pattern matching

In the traditional pattern matching problem, we seek to find a pattern $P : [m] \to \Sigma$ within a text $T : [n] \to \Sigma$.

$T =$

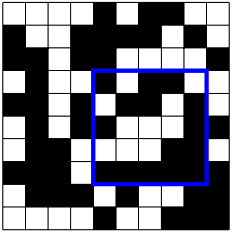
Q	U	A	N	T	U	M

$P =$

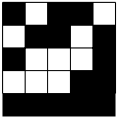
A	N	T

We can generalise this to higher dimensions d, where $P : [m]^d \to \Sigma$ and $T : [n]^d \to \Sigma$:

$T =$



$P =$



Pattern matching

Focusing on the 1-dimensional problem for now:

- Classically, it is known that this problem can be solved in worst-case time $O(n+m)$ [Knuth, Morris and Pratt '77].

Pattern matching

Focusing on the 1-dimensional problem for now:

- Classically, it is known that this problem can be solved in worst-case time $O(n+m)$ [Knuth, Morris and Pratt '77].
- There is a quantum algorithm which solves this problem (with bounded failure probability) in time $\widetilde{O}(\sqrt{n}+\sqrt{m})$ [Ramesh and Vinay '03].

Pattern matching

Focusing on the 1-dimensional problem for now:

- Classically, it is known that this problem can be solved in worst-case time $O(n+m)$ [Knuth, Morris and Pratt '77].
- There is a quantum algorithm which solves this problem (with bounded failure probability) in time $\widetilde{O}(\sqrt{n}+\sqrt{m})$ [Ramesh and Vinay '03].

Both these bounds are optimal in the worst case. But…what about the average case?

Pattern matching

Focusing on the 1-dimensional problem for now:

- Classically, it is known that this problem can be solved in worst-case time $O(n+m)$ [Knuth, Morris and Pratt '77].
- There is a quantum algorithm which solves this problem (with bounded failure probability) in time $\widetilde{O}(\sqrt{n}+\sqrt{m})$ [Ramesh and Vinay '03].

Both these bounds are optimal in the worst case. But... what about the average case?

Here we consider a simple model where each character of T is picked uniformly at random from Σ, and either:

- P is chosen to be an arbitrary substring of T; or
- P is uniformly random.

Could this be easier?

Pattern matching

- Classically, one can solve the average-case problem using $\widetilde{O}(n/m + \sqrt{n})$ queries, and this is optimal.

Pattern matching

- Classically, one can solve the average-case problem using $\widetilde{O}(n/m + \sqrt{n})$ queries, and this is optimal.
- But quantumly, we have the following result:

Theorem (modulo minor technicalities)

Let $T : [n] \to \Sigma$, $P : [m] \to \Sigma$ be picked as on the previous slide. Then there is a quantum algorithm which makes

$$\widetilde{O}(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$$

queries and determines whether P matches T.

Pattern matching

- Classically, one can solve the average-case problem using $\widetilde{O}(n/m + \sqrt{n})$ queries, and this is optimal.
- But quantumly, we have the following result:

Theorem (modulo minor technicalities)

Let $T : [n] \to \Sigma$, $P : [m] \to \Sigma$ be picked as on the previous slide. Then there is a quantum algorithm which makes

$$\widetilde{O}(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$$

queries and determines whether P matches T. If P does match T, the algorithm also outputs the position at which the match occurs.

Pattern matching

- Classically, one can solve the average-case problem using $\widetilde{O}(n/m + \sqrt{n})$ queries, and this is optimal.
- But quantumly, we have the following result:

Theorem (modulo minor technicalities)

Let $T : [n] \to \Sigma$, $P : [m] \to \Sigma$ be picked as on the previous slide. Then there is a quantum algorithm which makes

$$\widetilde{O}(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$$

queries and determines whether P matches T. If P does match T, the algorithm also outputs the position at which the match occurs. The algorithm fails with probability $O(1/n)$, taken over both the choice of T and P, and its internal randomness.

Pattern matching

- Classically, one can solve the average-case problem using $\widetilde{O}(n/m + \sqrt{n})$ queries, and this is optimal.
- But quantumly, we have the following result:

Theorem (modulo minor technicalities)

Let $T : [n] \to \Sigma$, $P : [m] \to \Sigma$ be picked as on the previous slide. Then there is a quantum algorithm which makes

$$\widetilde{O}(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$$

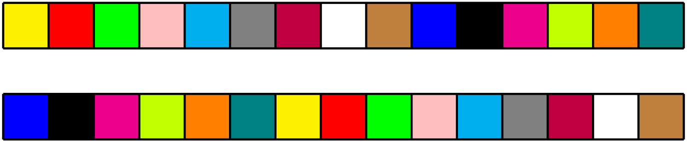
queries and determines whether P matches T. If P does match T, the algorithm also outputs the position at which the match occurs. The algorithm fails with probability $O(1/n)$, taken over both the choice of T and P, and its internal randomness.

This is a super-polynomial speedup for large m.

The dihedral hidden subgroup problem

The main quantum ingredient in the algorithm is an algorithm for the dihedral hidden subgroup problem:

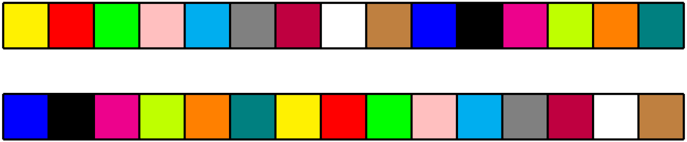
- Given two injective functions $f, g : \mathbb{Z}_N \to X$ such that $g(x) = f(x + s)$ for some $s \in \mathbb{Z}_N$, determine s.



The dihedral hidden subgroup problem

The main quantum ingredient in the algorithm is an algorithm for the dihedral hidden subgroup problem:

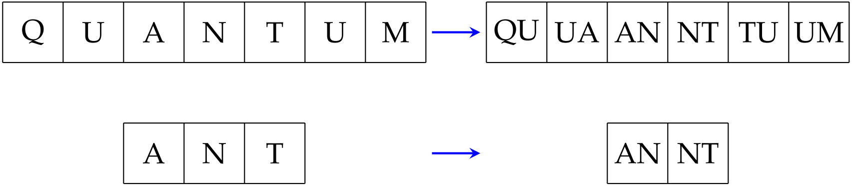
- Given two injective functions $f, g : \mathbb{Z}_N \to X$ such that $g(x) = f(x + s)$ for some $s \in \mathbb{Z}_N$, determine s.



- The best known quantum algorithm for the dihedral HSP uses $2^{O(\sqrt{\log N})} = o(N^{\epsilon})$ queries [Kuperberg '05].
- Classically, there is a lower bound of $\Omega(\sqrt{N})$ queries.

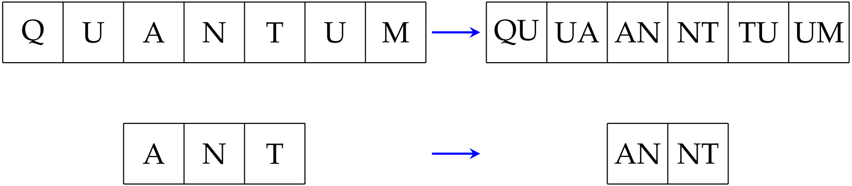
From the dihedral HSP to pattern matching

First, we make the pattern and text injective by concatenating characters (an idea used previously in some different contexts [Knuth '77, Gharibi '13]):



From the dihedral HSP to pattern matching

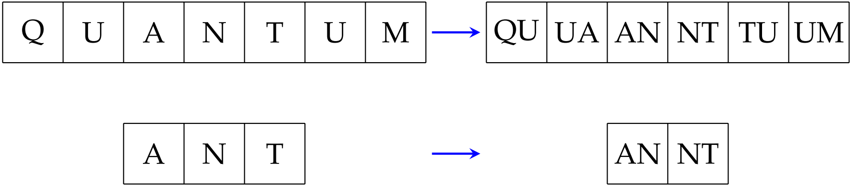
First, we make the pattern and text injective by concatenating characters (an idea used previously in some different contexts [Knuth '77, Gharibi '13]):



- Concatenation preserves the property of the pattern matching the text.

From the dihedral HSP to pattern matching

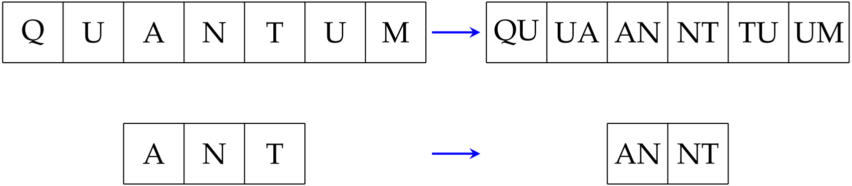
First, we make the pattern and text injective by concatenating characters (an idea used previously in some different contexts [Knuth '77, Gharibi '13]):



- Concatenation preserves the property of the pattern matching the text.
- If we produce a new alphabet whose symbols are strings of length k, a query to the new string can be simulated by k queries to the original string.

From the dihedral HSP to pattern matching

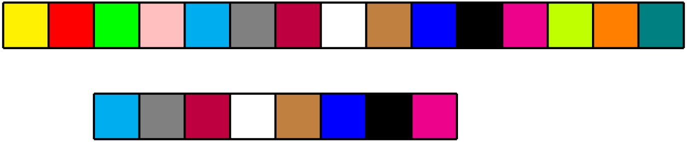
First, we make the pattern and text injective by concatenating characters (an idea used previously in some different contexts [Knuth '77, Gharibi '13]):



- Concatenation preserves the property of the pattern matching the text.
- If we produce a new alphabet whose symbols are strings of length k, a query to the new string can be simulated by k queries to the original string.
- For random strings, it suffices to take $k = O(\log n)$.

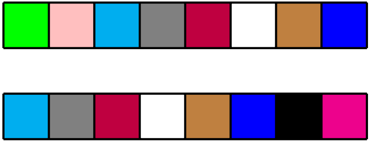
From the dihedral HSP to pattern matching

Second, we apply the dihedral HSP algorithm to the pattern and the text, at a randomly chosen offset.



From the dihedral HSP to pattern matching

Second, we apply the dihedral HSP algorithm to the pattern and the text, at a randomly chosen offset.



From the dihedral HSP to pattern matching

Second, we apply the dihedral HSP algorithm to the pattern and the text, at a randomly chosen offset.

Claim

If our guess for the start of the pattern is correct to within distance $m\,2^{-O(\sqrt{\log m})}$, the dihedral HSP algorithm outputs the correct position for the start of the pattern.

Completing the argument

- The probability of our guess being in this "good" range is $p = \Omega(m\,2^{-O(\sqrt{\log m})}/n)$.

Completing the argument

- The probability of our guess being in this "good" range is $p = \Omega(m\, 2^{-O(\sqrt{\log m})}/n)$.
- Using a variant of Grover's algorithm which can cope with bounded-error inputs, we can find a "good" position of this kind using $O(1/\sqrt{p}) = O(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$ queries.

Completing the argument

- The probability of our guess being in this "good" range is $p = \Omega(m\, 2^{-O(\sqrt{\log m})}/n)$.
- Using a variant of Grover's algorithm which can cope with bounded-error inputs, we can find a "good" position of this kind using $O(1/\sqrt{p}) = O(\sqrt{n/m}\, 2^{O(\sqrt{\log m})})$ queries.

For the quantum connoisseurs:

- To extend this to higher dimensions d we need to generalise Kuperberg's dihedral HSP algorithm.
- We also give a new variant of his algorithm with the equal best known complexity and a simpler correctness proof.

Search with wildcards

We are given oracle access to an unknown n-bit string x. Our task is to determine x using the minimum number of queries.

Search with wildcards

We are given oracle access to an unknown n-bit string x. Our task is to determine x using the minimum number of queries.

- The different possible queries are given by strings $s \in \{0, 1, *\}^n$. A query returns 1 if $x_i = s_i$ for all i such that $s_i \neq *$, and returns 0 otherwise.

Search with wildcards

We are given oracle access to an unknown n-bit string x. Our task is to determine x using the minimum number of queries.

- The different possible queries are given by strings $s \in \{0, 1, *\}^n$. A query returns 1 if $x_i = s_i$ for all i such that $s_i \neq *$, and returns 0 otherwise.
- A generalisation of the simple model where we are allowed to query individual bits of x.

Search with wildcards

We are given oracle access to an unknown n-bit string x. Our task is to determine x using the minimum number of queries.

- The different possible queries are given by strings $s \in \{0, 1, *\}^n$. A query returns 1 if $x_i = s_i$ for all i such that $s_i \neq *$, and returns 0 otherwise.
- A generalisation of the simple model where we are allowed to query individual bits of x.

Classically, we need n queries to determine x (each query gives one bit of information).

Search with wildcards

We are given oracle access to an unknown n-bit string x. Our task is to determine x using the minimum number of queries.

- The different possible queries are given by strings $s \in \{0, 1, *\}^n$. A query returns 1 if $x_i = s_i$ for all i such that $s_i \neq *$, and returns 0 otherwise.
- A generalisation of the simple model where we are allowed to query individual bits of x.

Classically, we need n queries to determine x (each query gives one bit of information).

Theorem

Search with wildcards can be solved with success probability 2/3 using $O(\sqrt{n})$ quantum queries.

Solving search with wildcards

The algorithm for search with wildcards is based on this claim:

Measurement Lemma

Fix $n \geqslant 1$ and, for any $0 \leqslant k \leqslant n$, set

$$|\psi_x^k\rangle := \frac{1}{\binom{n}{k}^{1/2}} \sum_{S \subseteq [n], |S|=k} |S\rangle |x_S\rangle,$$

where $|x_S\rangle := \bigotimes_{i \in S} |x_i\rangle$. Then, for any $k = n - O(\sqrt{n})$, there is a quantum measurement which, on input $|\psi_x^k\rangle$, outputs $\widetilde{x}$ such that the expected Hamming distance $d(x, \widetilde{x})$ is $O(1)$.

Solving search with wildcards

The algorithm for search with wildcards is based on this claim:

Measurement Lemma

Fix $n \geqslant 1$ and, for any $0 \leqslant k \leqslant n$, set

$$|\psi_x^k\rangle := \frac{1}{\binom{n}{k}^{1/2}} \sum_{S \subseteq [n], |S|=k} |S\rangle |x_S\rangle,$$

where $|x_S\rangle := \bigotimes_{i \in S} |x_i\rangle$. Then, for any $k = n - O(\sqrt{n})$, there is a quantum measurement which, on input $|\psi_x^k\rangle$, outputs $\widetilde{x}$ such that the expected Hamming distance $d(x, \widetilde{x})$ is $O(1)$.

- This is surprising because the equivalent classical statement is not true!

Solving search with wildcards

The algorithm for search with wildcards is based on this claim:

Measurement Lemma

Fix $n \geqslant 1$ and, for any $0 \leqslant k \leqslant n$, set

$$|\psi_x^k\rangle := \frac{1}{\binom{n}{k}^{1/2}} \sum_{S \subseteq [n], |S|=k} |S\rangle |x_S\rangle,$$

where $|x_S\rangle := \bigotimes_{i \in S} |x_i\rangle$. Then, for any $k = n - O(\sqrt{n})$, there is a quantum measurement which, on input $|\psi_x^k\rangle$, outputs $\widetilde{x}$ such that the expected Hamming distance $d(x, \widetilde{x})$ is $O(1)$.

- This is surprising because the equivalent classical statement is not true!
- The proof uses some basic Fourier analysis over $\mathbb{Z}_2^n$ and combinatorics.

Solving search with wildcards

Our algorithm for search with wildcards uses the Measurement Lemma to repeatedly learn $O(\sqrt{n})$ bits of x at a time in superposition.

Solving search with wildcards

Our algorithm for search with wildcards uses the Measurement Lemma to repeatedly learn $O(\sqrt{n})$ bits of x at a time in superposition.

- Imagine we have $|\psi_x^k\rangle$. For $k' > k$, this can be mapped to

$$\sum_{S':S\subseteq[n],|S'|=k'} |S'\rangle \left(\sum_{S:S\subseteq S',|S|=k} |S\rangle|x_S\rangle \right) = \sum_{S:S\subseteq[n],|S|=k'} |S\rangle|\psi_{x_S}^k\rangle,$$

so if we can map $|\psi_{x_S}^k\rangle \mapsto |x_S\rangle$, we've made $|\psi_x^{k'}\rangle$.

Solving search with wildcards

Our algorithm for search with wildcards uses the Measurement Lemma to repeatedly learn $O(\sqrt{n})$ bits of x at a time in superposition.

- Imagine we have $|\psi_x^k\rangle$. For $k' > k$, this can be mapped to

$$\sum_{S':S\subseteq[n],|S'|=k'} |S'\rangle \left(\sum_{S:S\subseteq S',|S|=k} |S\rangle|x_S\rangle \right) = \sum_{S:S\subseteq[n],|S|=k'} |S\rangle|\psi_{x_S}^k\rangle,$$

  so if we can map $|\psi_{x_S}^k\rangle \mapsto |x_S\rangle$, we've made $|\psi_x^{k'}\rangle$.

- By the lemma, we can do this when $k = k' - O(\sqrt{k'})$.

Solving search with wildcards

Our algorithm for search with wildcards uses the Measurement Lemma to repeatedly learn $O(\sqrt{n})$ bits of x at a time in superposition.

- Imagine we have $|\psi_x^k\rangle$. For $k' > k$, this can be mapped to

  $$\sum_{S':S\subseteq[n],|S'|=k'} |S'\rangle \left(\sum_{S:S\subseteq S',|S|=k} |S\rangle|x_S\rangle \right) = \sum_{S:S\subseteq[n],|S|=k'} |S\rangle|\psi_{x_S}^k\rangle,$$

  so if we can map $|\psi_{x_S}^k\rangle \mapsto |x_S\rangle$, we've made $|\psi_x^{k'}\rangle$.
- By the lemma, we can do this when $k = k' - O(\sqrt{k'})$.
- After each measurement, an expected $O(1)$ bits are incorrect.

Solving search with wildcards

Our algorithm for search with wildcards uses the Measurement Lemma to repeatedly learn $O(\sqrt{n})$ bits of x at a time in superposition.

- Imagine we have $|\psi_x^k\rangle$. For $k' > k$, this can be mapped to

$$\sum_{S':S\subseteq[n],|S'|=k'} |S'\rangle \left(\sum_{S:S\subseteq S',|S|=k} |S\rangle|x_S\rangle \right) = \sum_{S:S\subseteq[n],|S|=k'} |S\rangle|\psi_{x_S}^k\rangle,$$

  so if we can map $|\psi_{x_S}^k\rangle \mapsto |x_S\rangle$, we've made $|\psi_x^{k'}\rangle$.

- By the lemma, we can do this when $k = k' - O(\sqrt{k'})$.
- After each measurement, an expected $O(1)$ bits are incorrect.
- How to fix these?

Combinatorial group testing (CGT)

Proposed by [Dorfman '43] as a means of "weeding out all syphilitic men called up for induction".

Combinatorial group testing (CGT)

Proposed by [Dorfman '43] as a means of "weeding out all syphilitic men called up for induction".

The abstract problem is:

- We have a set of n items $x_1, \ldots, x_n \in \{0, 1\}$.

Combinatorial group testing (CGT)

Proposed by [Dorfman '43] as a means of "weeding out all syphilitic men called up for induction".

The abstract problem is:

- We have a set of n items $x_1, \ldots, x_n \in \{0, 1\}$.
- At most $k \ll n$ items x_i are special and have $x_i = 1$.

Combinatorial group testing (CGT)

Proposed by [Dorfman '43] as a means of "weeding out all syphilitic men called up for induction".

The abstract problem is:

- We have a set of n items $x_1, \ldots, x_n \in \{0, 1\}$.
- At most $k \ll n$ items x_i are special and have $x_i = 1$.
- We are allowed to query any subset $S \subseteq [n] := \{1, \ldots, n\}$. A query returns 1 if any items in S are special.

Combinatorial group testing (CGT)

Proposed by [Dorfman '43] as a means of "weeding out all syphilitic men called up for induction".

The abstract problem is:

- We have a set of n items $x_1, \ldots, x_n \in \{0, 1\}$.
- At most $k \ll n$ items x_i are special and have $x_i = 1$.
- We are allowed to query any subset $S \subseteq [n] := \{1, \ldots, n\}$. A query returns 1 if any items in S are special.
- We want to output the identities of all of the special items using the minimal number of queries.

Combinatorial group testing (CGT)

Proposed by [Dorfman '43] as a means of "weeding out all syphilitic men called up for induction".

The abstract problem is:

- We have a set of n items $x_1, \dots, x_n \in \{0, 1\}$.
- At most $k \ll n$ items x_i are special and have $x_i = 1$.
- We are allowed to query any subset $S \subseteq [n] := \{1, \dots, n\}$. A query returns 1 if any items in S are special.
- We want to output the identities of all of the special items using the minimal number of queries.

In particular, we would like to minimise the dependence on n.

Combinatorial group testing

- The number of classical queries required to solve CGT is

$$\Theta\left(\log\binom{n}{k}\right) = \Theta(k\log(n/k)).$$

- Many applications known: molecular biology, data streaming algorithms, compressed sensing, pattern matching in strings, ...

Combinatorial group testing

- The number of classical queries required to solve CGT is

$$\Theta\left(\log\binom{n}{k}\right) = \Theta(k\log(n/k)).$$

- Many applications known: molecular biology, data streaming algorithms, compressed sensing, pattern matching in strings, ...

Theorem

CGT can be solved using $O(k)$ quantum queries.

Combinatorial group testing

- The number of classical queries required to solve CGT is

$$\Theta\left(\log\binom{n}{k}\right) = \Theta(k \log(n/k)).$$

- Many applications known: molecular biology, data streaming algorithms, compressed sensing, pattern matching in strings, …

Theorem

CGT can be solved using $O(k)$ quantum queries.

This has subsequently been improved to $O(\sqrt{k})$ queries [Belovs '13], which is optimal.

Back to search with wildcards

- When we measure $|\psi_x^k\rangle$, we get an outcome $\widetilde{x}$ such that $d(\widetilde{x}, x) = O(1)$.
- We want to determine x, which is equivalent to determining $\widetilde{x} \oplus x$, a string of Hamming weight $O(1)$.
- A wildcard query corresponding to $S \subseteq [n]$ and $\widetilde{x}_S \oplus y$, $y \in \{0,1\}^{|S|}$, returns 1 iff all bits of $\widetilde{x}_S$ are correct.
- So we can use the algorithm for CGT to find, and correct, all incorrect bits using $O(1)$ queries.

Boolean functions and influential variables

- For the purposes of the rest of this talk, a boolean function is a function of the form

$$f : \{0,1\}^n \to \{\pm 1\}.$$

- Define the influence of the j'th variable as

$$I_j(f) = \Pr_x[f(x) \neq f(x^j)],$$

  where x^j is the bit-string formed by starting with x and flipping the j'th bit.

- For example, if $f : \{0,1\}^2 \to \{\pm 1\}$ is defined by $f(x) = x_1$,

$$I_1(f) = 1, \quad I_2(f) = 0.$$

Boolean functions and influential variables

The Kahn-Kalai-Linial (KKL) theorem states that every (balanced) boolean function has an influential variable:

Theorem **[Kahn, Kalai and Linial '88]**

Let $f : \{0,1\}^n \to \{\pm 1\}$ satisfy $\mathbb{E}[f] = 0$. Then there exists j such that

$$I_j(f) = \Omega((\log n)/n).$$

Boolean functions and influential variables

The Kahn-Kalai-Linial (KKL) theorem states that every (balanced) boolean function has an influential variable:

Theorem [Kahn, Kalai and Linial '88]

Let $f : \{0,1\}^n \to \{\pm 1\}$ satisfy $\mathbb{E}[f] = 0$. Then there exists j such that

$$I_j(f) = \Omega((\log n)/n).$$

Corollary

In any balanced voting scheme on n parties, there is a coalition of $O(n/\log n)$ voters who control the outcome of the election with probability 99%.

Boolean functions and influential variables

The Kahn-Kalai-Linial (KKL) theorem states that every (balanced) boolean function has an influential variable:

Theorem [Kahn, Kalai and Linial '88]

Let $f : \{0,1\}^n \to \{\pm 1\}$ satisfy $\mathbb{E}[f] = 0$. Then there exists j such that

$$I_j(f) = \Omega((\log n)/n).$$

Corollary

In any balanced voting scheme on n parties, there is a coalition of $O(n/\log n)$ voters who control the outcome of the election with probability 99%.

We would like to generalise this to the quantum setting...

Quantum boolean functions

A natural quantum (aka noncommutative) generalisation of the concept of a boolean function:

- A square 2^n-dimensional matrix F (i.e. a matrix acting on n qubits) whose eigenvalues are all ± 1.
- Then a classical boolean function corresponds to a diagonal matrix.

Quantum boolean functions

A natural quantum (aka noncommutative) generalisation of the concept of a boolean function:

- A square 2^n-dimensional matrix F (i.e. a matrix acting on n qubits) whose eigenvalues are all ± 1.
- Then a classical boolean function corresponds to a diagonal matrix.

Many of the concepts from classical analysis of boolean functions carry across to the quantum setting.

Quantum boolean functions

A natural quantum (aka noncommutative) generalisation of the concept of a boolean function:

- A square 2^n-dimensional matrix F (i.e. a matrix acting on n qubits) whose eigenvalues are all ± 1.
- Then a classical boolean function corresponds to a diagonal matrix.

Many of the concepts from classical analysis of boolean functions carry across to the quantum setting.

In particular, a natural generalisation of Fourier expansion of functions (in terms of the characters of the group $\mathbb{Z}_2^n$) is expansion in terms of tensor products of the Pauli matrices

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \; X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \; Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \; Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Influence and quantum boolean functions

Define the derivative of F in the j'th direction as

$$\Delta_j(F) := F - (\operatorname{tr}_j F) \otimes \frac{I_j}{2}.$$

- The second term traces out (throws away) the j'th qubit of F and replaces it with the (normalised) identity matrix.
- For example: $\Delta_1(X \otimes I) = X \otimes I$, $\quad \Delta_2(X \otimes I) = 0$.

Influence and quantum boolean functions

Define the derivative of F in the j'th direction as

$$\Delta_j(F) := F - (\mathrm{tr}_j\, F) \otimes \frac{I_j}{2}.$$

- The second term traces out (throws away) the j'th qubit of F and replaces it with the (normalised) identity matrix.
- For example: $\Delta_1(X \otimes I) = X \otimes I$, $\quad \Delta_2(X \otimes I) = 0$.

Define the influence of the j'th qubit as

$$\|\Delta_j\|_2^2 := \frac{\mathrm{tr}(\Delta_j^2)}{2^n}.$$

- For example: $I_1(X \otimes I) = 1$, $\quad I_2(X \otimes I) = 0$.

Does every quantum boolean function have an influential qubit?

Conjecture [AM and Osborne '08]

For every quantum boolean function F on n qubits such that $\operatorname{tr} F = 0$, there is a qubit j such that $I_j(F) = \Omega((\log n)/n)$.

Does every quantum boolean function have an influential qubit?

Conjecture [AM and Osborne '08]

For every quantum boolean function F on n qubits such that $\operatorname{tr} F = 0$, there is a qubit j such that $I_j(F) = \Omega((\log n)/n)$.

- We can easily prove a weaker lower bound of $\Omega(1/n)$.

Does every quantum boolean function have an influential qubit?

Conjecture [AM and Osborne '08]

For every quantum boolean function F on n qubits such that $\operatorname{tr} F = 0$, there is a qubit j such that $I_j(F) = \Omega((\log n)/n)$.

- We can easily prove a weaker lower bound of $\Omega(1/n)$.
- We can also prove the conjecture in a few special cases (for example, when F can be diagonalised by local unitaries, or can be expressed as a sum of anticommuting terms).

Does every quantum boolean function have an influential qubit?

Conjecture [AM and Osborne '08]

For every quantum boolean function F on n qubits such that $\operatorname{tr} F = 0$, there is a qubit j such that $I_j(F) = \Omega((\log n)/n)$.

- We can easily prove a weaker lower bound of $\Omega(1/n)$.
- We can also prove the conjecture in a few special cases (for example, when F can be diagonalised by local unitaries, or can be expressed as a sum of anticommuting terms).
- The conjecture might also be true for unitary operators in general.

A step on the path to this conjecture?

- A key ingredient in the proof of the KKL Theorem is the hypercontractive (Bonami-Gross-Beckner) inequality for noise applied to functions $f : \{0,1\}^n \to \mathbb{R}$.

A step on the path to this conjecture?

- A key ingredient in the proof of the KKL Theorem is the hypercontractive (Bonami-Gross-Beckner) inequality for noise applied to functions $f : \{0, 1\}^n \to \mathbb{R}$.
- We can prove a suitable quantum generalisation of this result to hypercontractivity of the qubit depolarising channel.

A step on the path to this conjecture?

- A key ingredient in the proof of the KKL Theorem is the hypercontractive (Bonami-Gross-Beckner) inequality for noise applied to functions $f : \{0,1\}^n \to \mathbb{R}$.
- We can prove a suitable quantum generalisation of this result to hypercontractivity of the qubit depolarising channel.
- This has the following consequence:

A quantum generalisation of a lemma of Talagrand

Let F be a traceless Hermitian operator on n qubits. Then

$$\|F\|_2^2 \leqslant \sum_{j=1}^{n} \frac{10\|\Delta_j(F)\|_2^2}{(2/3)\log(\|\Delta_j(F)\|_2/\|\Delta_j(F)\|_1) + 1}.$$

A step on the path to this conjecture?

A quantum generalisation of a lemma of Talagrand

Let F be a traceless Hermitian operator on n qubits. Then

$$\|F\|_2^2 \leqslant \sum_{j=1}^{n} \frac{10\|\Delta_j(F)\|_2^2}{(2/3)\log(\|\Delta_j(F)\|_2/\|\Delta_j(F)\|_1)+1}.$$

- Classically, the KKL Theorem follows immediately from this lemma, using the fact that $\Delta_j(f)$ only takes values in $\{0, 1, -1\}$, allowing us to control the denominator.
- The analogue does not hold in the quantum setting!
- It seems we need to find a "non-combinatorial" argument. . .

Summary

We have seen that quantum algorithms:

- … can provide a substantial speedup for pattern matching problems on average-case inputs;
- … can achieve a square-root speedup for search with wildcards.

Summary

We have seen that quantum algorithms:

- … can provide a substantial speedup for pattern matching problems on average-case inputs;
- … can achieve a square-root speedup for search with wildcards.

There are a number of results in the classical theory of boolean functions for which it would be very nice to have quantum analogues: one particularly annoying example is the KKL Theorem.

Summary

We have seen that quantum algorithms:

- … can provide a substantial speedup for pattern matching problems on average-case inputs;
- … can achieve a square-root speedup for search with wildcards.

There are a number of results in the classical theory of boolean functions for which it would be very nice to have quantum analogues: one particularly annoying example is the KKL Theorem.

Another open problem: what is the quantum query complexity of the dihedral hidden subgroup problem?

Thanks!

Some further reading:

- Quantum pattern matching fast on average
  `arXiv:1408.1816`
- Quantum algorithms for search with wildcards and combinatorial group testing (with Andris Ambainis)
  Quantum Information & Computation, vol. 14 no. 5&6, pp. 439–453, 2014; `arXiv:1210.1148`
- Quantum boolean functions (with Tobias Osborne)
  Chicago Journal of Theoretical Computer Science 2010; `arXiv:0810.2435`

Proving the measurement lemma

We need to prove we can distinguish the $|\psi_x^k\rangle$ states. We use the pretty good measurement (PGM).

Proving the measurement lemma

We need to prove we can distinguish the $|\psi_x^k\rangle$ states. We use the pretty good measurement (PGM).

Lemma

The probability that the PGM outputs y on input $|\psi_x^k\rangle$ is precisely $(\sqrt{G})_{xy}^2$, where

$$G_{xy} = \langle\psi_x^k|\psi_y^k\rangle = \frac{1}{\binom{n}{k}} \sum_{S\subseteq[n],|S|=k} [x_S = y_S] = \frac{\binom{n-d(x,y)}{k}}{\binom{n}{k}}.$$

Proving the measurement lemma

We need to prove we can distinguish the $|\psi_x^k\rangle$ states. We use the pretty good measurement (PGM).

Lemma

The probability that the PGM outputs y on input $|\psi_x^k\rangle$ is precisely $(\sqrt{G})_{xy}^2$, where

$$G_{xy} = \langle \psi_x^k | \psi_y^k \rangle = \frac{1}{\binom{n}{k}} \sum_{S \subseteq [n], |S|=k} [x_S = y_S] = \frac{\binom{n-d(x,y)}{k}}{\binom{n}{k}}.$$

- We want to bound $D_k := \sum_{y \in \{0,1\}^n} d(x,y) (\sqrt{G}_{xy})^2$.

Proving the measurement lemma

We need to prove we can distinguish the $|\psi_x^k\rangle$ states. We use the pretty good measurement (PGM).

Lemma

The probability that the PGM outputs y on input $|\psi_x^k\rangle$ is precisely $(\sqrt{G})^2_{xy}$, where

$$G_{xy} = \langle \psi_x^k | \psi_y^k \rangle = \frac{1}{\binom{n}{k}} \sum_{S \subseteq [n], |S|=k} [x_S = y_S] = \frac{\binom{n-d(x,y)}{k}}{\binom{n}{k}}.$$

- We want to bound $D_k := \sum_{y \in \{0,1\}^n} d(x,y)(\sqrt{G}_{xy})^2$.
- G_{xy} depends only on $x \oplus y$, so G is diagonalised by the Fourier transform over $\mathbb{Z}_2^n$ and D_k does not depend on x.

Proving the measurement lemma

We need to prove we can distinguish the $|\psi_x^k\rangle$ states. We use the pretty good measurement (PGM).

Lemma

The probability that the PGM outputs y on input $|\psi_x^k\rangle$ is precisely $(\sqrt{G})^2_{xy}$, where

$$G_{xy} = \langle \psi_x^k | \psi_y^k \rangle = \frac{1}{\binom{n}{k}} \sum_{S \subseteq [n], |S|=k} [x_S = y_S] = \frac{\binom{n-d(x,y)}{k}}{\binom{n}{k}}.$$

- We want to bound $D_k := \sum_{y \in \{0,1\}^n} d(x,y) (\sqrt{G}_{xy})^2$.
- G_{xy} depends only on $x \oplus y$, so G is diagonalised by the Fourier transform over $\mathbb{Z}_2^n$ and D_k does not depend on x.
- D_k can be upper bounded using Fourier duality and some combinatorics.

Combinatorial group testing

The $k = 1$ case

If $k = 1$, CGT can be solved exactly using one quantum query.

Combinatorial group testing

The $k = 1$ case

If $k = 1$, CGT can be solved exactly using one quantum query.

1. Create the state $\frac{1}{\sqrt{2^{n+1}}} \sum_{s \in \{0,1\}^n} |s\rangle(|0\rangle - |1\rangle)$.

Combinatorial group testing

The $k = 1$ case

If $k = 1$, CGT can be solved exactly using one quantum query.

1. Create the state $\frac{1}{\sqrt{2^{n+1}}} \sum_{s \in \{0,1\}^n} |s\rangle(|0\rangle - |1\rangle)$.
2. Apply the oracle to create the state

$$\begin{aligned} & \frac{1}{\sqrt{2^{n+1}}} \sum_{s \in \{0,1\}^n} (-1)^{\bigvee_i s_i x_i} |s\rangle(|0\rangle - |1\rangle) \\ = \quad & \frac{1}{\sqrt{2^{n+1}}} \sum_{s \in \{0,1\}^n} (-1)^{s \cdot x} |s\rangle(|0\rangle - |1\rangle). \end{aligned}$$

Combinatorial group testing

The $k = 1$ case

If $k = 1$, CGT can be solved exactly using one quantum query.

1. Create the state $\frac{1}{\sqrt{2^{n+1}}} \sum_{s \in \{0,1\}^n} |s\rangle(|0\rangle - |1\rangle)$.
2. Apply the oracle to create the state

$$\begin{aligned} & \frac{1}{\sqrt{2^{n+1}}} \sum_{s \in \{0,1\}^n} (-1)^{\bigvee_i s_i x_i} |s\rangle(|0\rangle - |1\rangle) \\ = \; & \frac{1}{\sqrt{2^{n+1}}} \sum_{s \in \{0,1\}^n} (-1)^{s \cdot x} |s\rangle(|0\rangle - |1\rangle). \end{aligned}$$

3. Apply Hadamard gates to each qubit of the first register and measure to obtain x.

Generalising this idea to arbitrary k

Claim

CGT can be solved using $O(k)$ quantum queries on average.

Generalising this idea to arbitrary k

Claim

CGT can be solved using $O(k)$ quantum queries on average.

- Construct $S \subseteq [n]$ by including each $i \in [n]$ with prob. $1/k$.

Generalising this idea to arbitrary k

Claim

CGT can be solved using $O(k)$ quantum queries on average.

- Construct $S \subseteq [n]$ by including each $i \in [n]$ with prob. $1/k$.
- Run the $k = 1$ algorithm on the subset of bits in S.

Generalising this idea to arbitrary k

Claim

CGT can be solved using $O(k)$ quantum queries on average.

- Construct $S \subseteq [n]$ by including each $i \in [n]$ with prob. $1/k$.
- Run the $k = 1$ algorithm on the subset of bits in S.
- If S contains exactly one 1 bit at position i, which will occur with probability at least $(1 - 1/k)^{k-1} \geqslant 1/e$, we are guaranteed to learn i.

Generalising this idea to arbitrary k

Claim

CGT can be solved using $O(k)$ quantum queries on average.

- Construct $S \subseteq [n]$ by including each $i \in [n]$ with prob. $1/k$.
- Run the $k = 1$ algorithm on the subset of bits in S.
- If S contains exactly one 1 bit at position i, which will occur with probability at least $(1 - 1/k)^{k-1} \geqslant 1/e$, we are guaranteed to learn i.
- We can check whether the index $\tilde{i}$ we received really is a 1 by making one more query to index $\tilde{i}$.

Generalising this idea to arbitrary k

Claim

CGT can be solved using $O(k)$ quantum queries on average.

- Construct $S \subseteq [n]$ by including each $i \in [n]$ with prob. $1/k$.
- Run the $k = 1$ algorithm on the subset of bits in S.
- If S contains exactly one 1 bit at position i, which will occur with probability at least $(1 - 1/k)^{k-1} \geqslant 1/e$, we are guaranteed to learn i.
- We can check whether the index $\tilde{i}$ we received really is a 1 by making one more query to index $\tilde{i}$.
- Following each successful query, we reduce k by 1 and exclude the bit that we just learned from future queries.

Generalising this idea to arbitrary k

Claim

CGT can be solved using $O(k)$ quantum queries on average.

- Construct $S \subseteq [n]$ by including each $i \in [n]$ with prob. $1/k$.
- Run the $k = 1$ algorithm on the subset of bits in S.
- If S contains exactly one 1 bit at position i, which will occur with probability at least $(1 - 1/k)^{k-1} \geqslant 1/e$, we are guaranteed to learn i.
- We can check whether the index $\tilde{i}$ we received really is a 1 by making one more query to index $\tilde{i}$.
- Following each successful query, we reduce k by 1 and exclude the bit that we just learned from future queries.
- In order to learn x completely, the expected overall number of queries used is $O(k)$.

A quantum hypercontractive inequality

Let $\mathcal{D}_\epsilon$ be the qubit depolarising channel with noise rate ϵ, i.e.

$$\mathcal{D}_\epsilon(\rho) = \frac{(1-\epsilon)}{2}\,\mathrm{tr}(\rho) I + \epsilon\rho.$$

Theorem [AM and Osborne '08, King '12]

Let M be a Hermitian operator on n qubits and fix $q \geqslant p \geqslant 1$. Then, provided that

$$\epsilon \leqslant \sqrt{\frac{p-1}{q-1}},$$

we have

$$\|\mathcal{D}_\epsilon^{\otimes n}(M)\|_q \leqslant \|f\|_p.$$

Here $\|\cdot\|_p$ is the normalised Schatten p-norm:

$$\|M\|_p = \left(\frac{\mathrm{tr}\,|M|^p}{2^n}\right)^{1/p}.$$