Quantum Algorithms for Element Distinctness

Harry Buhrman[*] Christoph Dürr[‡] Mark Heiligman[§] Peter Høyer[¶]
Frédéric Magniez[||] Miklos Santha[**] Ronald de Wolf[††]

Abstract

We present several applications of quantum amplitude amplification to finding claws and collisions in ordered or unordered functions. Our algorithms generalize those of Brassard, Høyer, and Tapp, and imply an $O(N^{3/4}\log N)$ quantum upper bound for the element distinctness problem in the comparison complexity model. This contrasts with $\Theta(N\log N)$ classical complexity. We also prove a lower bound of $\Omega(\sqrt{N})$ comparisons for this problem and derive bounds for a number of related problems.

1 Introduction

In the last decade, quantum computing has become a prominent and promising area of theoretical computer science. Realizing this promise requires two things: (1) actually building a quantum computer and (2) discovering tasks where a quantum computer is significantly faster than a classical computer. Here we are concerned with the second issue. Few good quantum algorithms are known to date, the two main examples being Shor's algorithm for factoring [20] and Grover's algorithm for searching [14]. Whereas the first so far has remained a seminal but somewhat isolated result, the second has been applied as a building block in quite a few other quantum algorithms [6, 8, 10, 11, 18, 7]. For a general introduction to quantum computing we refer to [19].

One of the earliest applications of Grover's algorithm was the algorithm of Brassard, Høyer, and Tapp [8] for finding a *collision* in a 2-to-1 function f. A collision is a pair of distinct elements x, y such that $f(x) = f(y)$. Suppose the size of f's domain is N. For a classical randomized algorithm, $\Theta(N^{1/2})$ evaluations of the function are necessary and sufficient to find a collision. The quantum algorithm of [8] finds a collision using $O(N^{1/3})$ evaluations of f. No non trivial quantum lower bound is known for this problem. A notion related to collisions is that of a *claw*. A claw in functions f and g is a pair (x, y) such that $f(x) = g(y)$. If f and g are permutations on $[N] = \{1, \ldots, N\}$, then the function on $[2N]$ that maps the first half of the domain according to f and the second half according to g, is a 2-to-1 function. Thus the algorithm of Brassard, Høyer, and Tapp can also find a claw in such f and g using $O(N^{1/3})$ evaluations of f and g.

In this paper we consider the quantum complexity of collision-finding or claw-finding with and without restrictions on the functions f and g. In Section 3 we consider the situation where $f : [N] \to Z$ and $g : [M] \to Z$ are arbitrary. Our aim is to find a claw between f and g, if one exists. For now, let us assume $N = M$ (in the body of the paper we treat the general case). The complexity measure we use is the number of *comparisons* between elements. That is, we assume a total order on Z and our only way to access f and g is by comparing $f(x)$ with $f(y)$, $g(x)$ with $g(y)$, or $f(x)$ with $g(y)$, according to this total order. The ability to make such comparisons is weaker than the ability to evaluate and actually obtain the function values $f(x)$ and $g(y)$, because if we can obtain the values $f(x)$ and $g(y)$, we can of course also compare those two values. Accordingly, the existence of a quantum algorithm that finds a claw using T comparisons implies the existence of a quantum algorithm that finds a claw using $O(T)$ function-evaluations. However, also our *lower* bounds on the complexity of claw-finding remain essentially the same if we were to count the number of function-evaluations instead

Research partially supported by the EU 5th framework programs QAIP IST-1999-11234, and RAND-APX IST-1999-14036.

[*] CWI, P.O.Box 94079, Amsterdam, the Netherlands. Also affiliated with the University of Amsterdam. Email: buhrman@cwi.nl.

[‡] Université Paris-Sud, LRI, 91405 Orsay, France. Email: durr@lri.fr.

[§] NSA, Suite 6111, Fort George G. Meade, MD 20755, USA. Email: mheilig@zombie.ncsc.mil.

[¶] Dept. of Comp. Sci., University of Calgary, Alberta, Canada T2N 1N4. email: hoyer@cpsc.ucalgary.ca. Research conducted while at BRICS, University of Aarhus, 8000 Aarhus C, Denmark.

[||] CNRS–LRI, UMR 8623 Université Paris–Sud, 91405 Orsay, France. Email: magniez@lri.fr.

[**] CNRS–LRI, UMR 8623 Université Paris–Sud, 91405 Orsay, France. Email: santha@lri.fr.

[††] CWI, P.O.Box 94079, Amsterdam, The Netherlands. Also affiliated with the University of Amsterdam. Email: rdewolf@cwi.nl.

of comparisons. This shows that it does not matter much for our results whether we count comparisons or function-evaluations.

A simple yet essentially optimal *classical* algorithm for this general claw-finding problem is the following. Viewing f as a list of N items, we can sort it using $N \log N + O(N)$ comparisons. Once f is sorted, we can for a given $y \in [N]$ find an x such that $f(x) = g(y)$ provided such an x exists, using $\log N$ comparisons (by utilizing binary search on f). Thus exhaustive search on all y yields an $O(N \log N)$ algorithm for finding a claw with certainty, provided one exists. This $N \log N$ is optimal up to constant factors even for bounded-error classical algorithms, as follows from the classical $\Omega(N \log N)$ bounds for the *element distinctness* problem, explained below. In this paper we show that a quantum computer can do better: we exhibit a quantum algorithm that finds a claw with high probability using $O(N^{3/4} \log N)$ comparisons. We also prove a lower bound for this problem of $\Omega(N^{1/2})$ comparisons for bounded-error quantum algorithms and $\Omega(N)$ for exact quantum algorithms.

Our algorithm for claw-finding also yields an $O(N^{3/4} \log N)$ bounded-error quantum algorithm for finding a collision for arbitrary functions. Note that *deciding* if a collision occurs in f is equivalent to deciding whether f maps all x to distinct elements. This is known as the *element distinctness* problem and has been well studied classically, see e.g. [21, 16, 13, 4]. Element distinctness is particularly interesting because its classical complexity is related to that of sorting, which is well known to require $N \log N + \Theta(N)$ comparisons. If we sort f, we can decide element distinctness by going through the sorted list once, which gives a classical upper bound of $N \log N + O(N)$ comparisons. Conversely, element distinctness requires $\Omega(N \log N)$ comparisons in case of classical bounded-error algorithms (even in a much stronger model, see [13]), so sorting and element distinctness are equally hard for classical computers. On a quantum computer, the best known upper bound for sorting is roughly $0.53\ N \log N$ comparisons [12], and it was recently shown that such a linear speed-up is the best possible: quantum sorting requires $\Omega(N \log N)$ comparisons, even if one allows a small probability of error [15]. Accordingly, our $O(N^{3/4} \log N)$ quantum upper bound shows that element distinctness is significantly easier than sorting for a quantum computer, in contrast to the classical case.

In Section 4, we consider the case where f is ordered (monotone non-decreasing): $f(1) \leq f(2) \leq \cdots \leq f(N)$. In this case, the quantum complexity of claw-finding and collision finding drops from $O(N^{3/4} \log N)$ to $O(N^{1/2} \log N)$. In Section 5 we show how to remove the $\log N$ factor (replacing it by a near-constant function) if both f and g are ordered. The lower bound for this restricted case remains $\Omega(N^{1/2})$.

In Section 6 we give some problems related to the element distinctness problem for which quantum computers cannot help. We then, in Section 7, give bounds for the number of edges a quantum computer needs to query in order to find a *triangle* in a given graph (which, informally, can be viewed as a collision between three nodes). Finally, we end with some concluding remarks in Section 8.

2 Preliminaries

We consider the following problems:

Claw-finding problem
Given two functions $f : X \to Z$ and $g : Y \to Z$, find a pair $(x, y) \in X \times Y$ such that $f(x) = g(y)$.

Collision-finding problem
Given a function $f : X \to Z$, find two distinct elements $x, y \in X$ such that $f(x) = f(y)$.

We assume that $X = [N] = \{1, \ldots, N\}$ and $Y = [M] = \{1, \ldots, M\}$ with $N \leq M$.

For general details about quantum computing we refer to [19]. We formalize a comparison between $f(x)$ and $f(y)$ as an application of the following unitary transformation:

$$|x, y, b\rangle \longmapsto |x, y, b \oplus [f(x) \leq f(y)]\rangle,$$

where $b \in \{0, 1\}$ and $[f(x) \leq f(y)]$ denotes the truth-value of the statement "$f(x) \leq f(y)$". We formalize comparisons between $f(x)$ and $g(y)$ similarly.

We are interested in the number of comparisons required for claw-finding or collision-finding. We will consider the complexity of *exact* algorithms, which are required to solve the problem with certainty, as well as *bounded-error* algorithms, which are required to solve the problem with probability at least $2/3$, for every input. We use $Q_E(P)$ and $Q_2(P)$ for the worst-case number of comparisons required for solving problem P by exact and bounded-error quantum algorithms, respectively. (The subscripts 'E' and '2' refer to exact computation and 2-sided bounded-error computation, respectively.) In our algorithms we make abundant use of quantum *amplitude amplification* [7], which generalizes quantum search [14]. The essence of amplitude amplification can be summarized by the following theorem.

Theorem 1 (Amplitude amplification)
There exists a quantum algorithm **QSearch** *with the following property. Let $\mathcal{A}$ be any quantum algorithm that uses no measurements, and let $\chi : \mathbb{Z} \to \{0, 1\}$ be any Boolean function. Let p denote the initial success probability of $\mathcal{A}$ of finding a solution (i.e., the probability of outputting z s.t. $\chi(z) = 1$). Algorithm* **QSearch** *finds a solution using an expected number of $O(1/\sqrt{p})$ applications of $\mathcal{A}$ and $\mathcal{A}^{-1}$ if $p > 0$, and otherwise runs forever.*

Very briefly, **QSearch** works by iterating the unitary transformation $Q = -\mathcal{A}S_0\mathcal{A}^{-1}S_\chi$ a number of times, starting with initial state $\mathcal{A}|0\rangle$. Here $S_\chi|z\rangle = (-1)^{\chi(z)}|z\rangle$, and $S_0|0\rangle = -|0\rangle$ and $S_0|z\rangle = |z\rangle$ for all $z \neq 0$. The analysis of [7] shows that doing a measurement after $O(1/\sqrt{p})$ iterations of Q will yield a solution with probability close to 1. The algorithm **QSearch** does not need to know the value of p in advance, but if p is known, then a slightly modified **QSearch** can find a solution *with certainty* using $O(1/\sqrt{p})$ applications of $\mathcal{A}$ and $\mathcal{A}^{-1}$.

Grover's algorithm for searching a space of N items is a special case of amplitude amplification, where $\mathcal{A}$ is the Hadamard transform on each qubit. This $\mathcal{A}$ has probability $p \geq 1/N$ of finding a solution (if there is one), so amplitude amplification implies an $O(N^{1/2})$ quantum algorithm for searching the space. We sometimes refer to this process as "quantum searching".

3 Finding claws if f and g are not ordered

First we consider the most general case, where f and g are arbitrary, possibly unordered functions. Our claw-finding algorithms are instances of the following generic algorithm, which is parameterized by an integer $\ell \leq \min\{N, \sqrt{M}\}$:

Algorithm: Generic claw-finder

1. Select a random subset $A \subseteq [N]$ of size ℓ
2. Select a random subset $B \subseteq [M]$ of size ℓ^2
3. Sort the elements in A according to their f-value
4. For a specific $b \in B$, we can check if there is an $a \in A$ such that (a, b) is a claw using classical binary search on the sorted version of A. Combine this with quantum search on the B-elements to search for a claw in $A \times B$.
5. Apply amplitude amplification on steps 1–4

We analyze the comparison-complexity of this algorithm. Step 3 just employs classical sorting and hence takes $\ell \log \ell + O(\ell)$ comparisons. Step 4 takes $O(\sqrt{|B|}\log|A|) = O(\ell \log \ell)$ comparisons, since testing if there is an A-element colliding with a given $b \in B$ takes $O(\log A)$ comparisons (via binary search on the sorted A) and the quantum search needs $O(\sqrt{|B|})$ such tests to find a B-element that collides with an element occurring in A (if there is one). In total, steps 1–4 take $O(\ell \log \ell)$ comparisons.

If no claws between f and g exist, then this algorithm does not terminate. Now suppose there is a claw $(x, y) \in X \times Y$. Then $(x, y) \in A \times B$ with probability $(\ell/N) \cdot (\ell^2/M)$, and if indeed $(x, y) \in A \times B$, then step 4 will find this (or some other) collision with probability at least $1/2$ in at most $O(\ell \log \ell)$ comparisons. Hence the overall success probability of steps 1–4 is at least $p = \ell^3/2NM$, and the amplitude amplification of step 5 requires an expected number of $O(\sqrt{NM/\ell^3})$ iterations of steps 1–4. Accordingly, the total expected number of comparisons to find a claw is $O(\sqrt{\frac{NM}{\ell}}\log \ell)$, provided there is one. In order to minimize the number of comparisons, we maximize ℓ, subject to the constraint $\ell \leq \min\{N, \sqrt{M}\}$. This gives upper bounds of $O(N^{1/2}M^{1/4}\log N)$ comparisons if $N \leq M \leq N^2$, and $O(M^{1/2}\log N)$ if $M > N^2$.

What about *lower* bounds for the claw-finding problem? We can reduce the **OR**-problem to claw-finding as follows. Given a function $g : [M] \to \{0, 1\}$, by definition $\mathbf{OR}(g)$ is 1 if there is an i such that $g(i) = 1$. To determine this value, we set $N = 1$ and define $f(1) = 1$. Then there is a claw between f and g if and only if $\mathbf{OR}(g) = 1$. Thus if we can find a claw using c comparisons, we can decide **OR** using $2c$ queries to g (two g-queries suffice to implement a comparison). Using known lower bounds for the **OR**-function [5, 3], this gives an $\Omega(M)$ bound for exact quantum and an $\Omega(\sqrt{M})$ bound for bounded-error quantum algorithms. The next theorem follows.

Theorem 2 *The comparison-complexity of the claw-finding problem is*

- $$\Omega(M^{1/2}) \leq Q_2(\mathbf{Claw}) \leq \begin{cases} O(N^{1/2}M^{1/4}\log N) \text{ if } N \leq M \leq N^2 \\ O(M^{1/2}\log N) \text{ if } M > N^2 \end{cases}$$
- $\Omega(M) \leq Q_E(\mathbf{Claw}) \leq O(M \log N)$.

The bounds for the case $M > N^2$ and the case of exact computation are tight up to the $\log N$ term, but the case $M \leq N^2$ is nowhere near tight. In particular, for $N = M$ the complexity lies somewhere between $N^{1/2}$ and $N^{3/4}\log N$.

Now consider the problem of finding a *collision* for an arbitrary function $f : [N] \to Z$, i.e., to find distinct $x, y \in [N]$ such that $f(x) = f(y)$. A simple modification of the above algorithm for claw-finding works fine to find such (x, y)-pairs if they exist (put $g = f$ and avoid claws of the form (x, x)), and gives a bounded-error algorithm that finds a collision using $O(N^{3/4}\log N)$ comparisons. This algorithm may be viewed as a modification of the Generic claw finder with $|A| = \ell = O(\sqrt{N})$ and $B = [N]\backslash A$. Note that now the choice of A determines B, so our algorithm only has to store A and sort it, which means that the *space* requirement of steps 1–3 is $O(\sqrt{N}\log N)$ qubits. The amplitude amplification of step 4 requires not more space than the algorithm that is being amplified, so the total space complexity of our algorithm is $O(\sqrt{N}\log N)$ as well.

As mentioned in the introduction, the problem of deciding if there is a collision is equivalent to the element distinctness (ED) problem. The best known lower bounds follow again via reductions from the **OR**-problem: given $X \in \{0,1\}^N$, we define $f : [N+1] \rightarrow \{0,\dots,N\}$ as $f(i) = i(1-x_i)$ and $f(N+1) = 0$. Now $\mathbf{OR}(X) = 1$ if and only if f contains a collision. Thus we obtain:

Theorem 3 *The comparison-complexity of the element distinctness problem is*

- $\Omega(N^{1/2}) \leq Q_2(\mathbf{ED}) \leq O(N^{3/4}\log N)$
- $\Omega(N) \leq Q_E(\mathbf{ED}) \leq O(N\log N)$.

In contrast, for classical (exact or bounded-error) algorithms, element distinctness is as hard as sorting and requires $\Theta(N\log N)$ comparisons.

Collision-finding requires fewer comparisons if we know that some value $z \in Z$ occurs at least k times. If we pick a random subset S of $10N/k$ of the domain, then with high probability at least two pre-images of z will be contained in S. Thus running our algorithm on S will find a collision with high probability, resulting in complexity $O((N/k)^{3/4}\log(N/k))$. Also, if f is a 2-to-1 function, we can rederive the $O(N^{1/3}\log N)$ bound of Brassard, Høyer, and Tapp [8] by taking $\ell = N^{1/3}$. This yields constant success probability after steps 1–4 in the generic algorithm, and hence no further rounds of amplitude amplification are required. As in the case of [8], this algorithm can be made exact by using the exact form of amplitude amplification (the success probability can be exactly computed in this case, so exact amplitude amplification is applicable).

In another direction, we may extend our results by considering the problem of finding a claw between *three* unordered functions f, g, h with domains of size N. That is, we want to find x, y, z such that $f(x) = h(y) = g(z)$. Classically, the best algorithm requires $\Theta(N\log N)$ comparisons. The following quantum algorithm solves the problem in $O(N^{7/8}\log N)$ comparisons and bounded error:

Algorithm: Claw-finder for 3 functions

1. Select a random subset A of $N^{3/4}$ elements from the domain of f and sort these according to their f-value
2. Select a random subset B of size $N^{1/2}$ from the domain of g and sort these according to their g-value
3. Search the domain of h for an element that forms a claw with a pair in $A \times B$
4. Apply amplitude amplification on steps 2–3
5. Apply amplitude amplification on steps 1–4

Step 1 takes $O(N^{3/4}\log N)$ comparisons, steps 2 and 3 each take $O(N^{1/2}\log N)$ comparisons. If there is a claw (x, y, z) with $x \in A$, then $y \in B$ with probability $1/\sqrt{N}$, so step 4 requires $O(N^{1/4})$ rounds of amplitude amplification, hence steps 1–4 together take $O(N^{3/4}\log N)$ comparisons. Steps 1–4 have probability $\approx 1/N^{1/4}$ of finding a claw, so step 5 in turn requires $O(N^{1/8})$ rounds of amplitude amplification. In total, the algorithm thus takes $O(N^{7/8}\log N)$ comparisons. More generally, finding a claw among k functions can be done in $O(N^{1-\frac{1}{2^k}}\log N)$ comparisons (for fixed k).

4 Finding claws if *f* is ordered

Now suppose that function f is ordered: $f(1) \leq f(2) \leq \dots \leq f(N)$, and that function $g : [M] \rightarrow Z$ is not necessarily ordered. In this case, given some $y \in [M]$, we can find an $x \in [N]$ such that (x, y) is a claw using binary search on f. Thus, combining this with a quantum search on all $y \in [M]$, we obtain the upper bound of $O(\sqrt{M}\log N)$ for finding a claw in f and g. The lower bounds of the last section via the **OR**-reduction still apply, and hence we obtain the following theorem.

Theorem 4 *The comparison-complexity of the claw-finding problem with ordered f is*

- $\Omega(M^{1/2}) \leq Q_2(\mathbf{Claw}) \leq O(M^{1/2}\log N)$
- $\Omega(M) \leq Q_E(\mathbf{Claw}) \leq O(M\log N)$.

Note that collision-finding for an ordered $f : [N] \rightarrow Z$ is equivalent to searching a space of $N-1$ items (namely all consecutive pairs in the domain of f) and hence requires $\Theta(\sqrt{N})$ comparisons.

5 Finding claws if both *f* and *g* are ordered

Now consider the case where both f and g are ordered. Assume for simplicity that $N = M$. Again we get an $\Omega(\sqrt{N})$ lower bound via a reduction from the **OR**-problem: given an **OR**-instance $X \in \{0,1\}^N$, we define $f, g : [N] \rightarrow Z$ by $f(i) = 2i+1$ and $g(i) = 2i + x_i$ for all $i \in [N]$. Then f and g are ordered, and $\mathbf{OR}(X) = 1$ if and only if there is a claw between f and g. The lower bound follows.

We give a quantum algorithm that solves the problem using $O\big(\sqrt{N}c^{\log^\star(N)}\big)$ comparisons for some constant $c > 0$. The function $\log^\star(N)$ is defined as the minimum number of iterated applications of the logarithm function necessary to obtain a number less than or equal to 1: $\log^\star(N) = \min\{i \geq 0 \mid \log^{(i)}(N) \leq 1\}$, where $\log^{(i)} = \log \circ \log^{(i-1)}$ denotes the ith iterated application of log, and $\log^{(0)}$ is the identity function. Even though $c^{\log^\star(N)}$ is exponential in $\log^\star(N)$, it is still very small in N, in particular

$c^{\log^\star(N)} \in o(\log^{(i)}(N))$ for any constant $i \geq 1$. Thus we replace the $\log N$ in the upper bound of the previous section by a near-constant function. Our algorithm defines a set of subproblems such that the original problem (f, g) contains a claw if and only if at least one of the subproblems contains a claw. We then solve the original problem by running the subproblems in quantum parallel and applying amplitude amplification.

Let $r > 0$ be an integer. We define $2\left\lceil \frac{N}{r} \right\rceil$ subproblems as follows.

Definition 5 *Let $r > 0$ be an integer and $f, g : [N] \to Z$. For each $0 \leq i \leq \lceil N/r \rceil - 1$, define the subproblem (f_i, g'_i) by letting f_i denote the restriction of f to subdomain $[ir+1, (i+1)r]$ and g'_i the restriction of g to $[j, j+r-1]$ where j is the minimum $j' \in [N]$ such that $g(j') \geq f(ir+1)$.*

Similarly, for each $0 \leq j \leq \lceil N/r \rceil - 1$, define the subproblem (f'_j, g_j) by letting g_j denote the restriction of g to $[jr+1, (j+1)r]$, and f'_j the restriction of f to $[i, i+r-1]$ where i is the minimum $i' \in [N]$ such that $f(i') \geq g(jr+1)$.

It is not hard to check that these subproblems all together provide a solution to the original problem.

Lemma 6 *Let $r > 0$ be an integer and $f, g : [N] \to Z$. Then (f, g) contains a claw if and only if for some i or j in $[0, \lceil N/r \rceil - 1]$ the subproblem (f_i, g'_i) or (f'_j, g_j) contains a claw.*

Each of these $2\left\lceil \frac{N}{r} \right\rceil$ subproblems is itself an instance of the claw-finding problem of size r. By running them all together in quantum parallel and then applying amplitude amplification, we obtain our main result.

Theorem 7 *There exists a quantum algorithm that outputs a claw between f and g with probability at least $\frac{2}{3}$ provided one exists, using $O\big(\sqrt{N} c^{\log^\star(N)}\big)$ comparisons, for some constant c.*

Proof Let $T(N)$ denote the worst-case number of comparisons required if f and g have domain of size N. We show that

$$T(N) \leq c' \sqrt{\frac{N}{r}} \Big(\lceil \log(N+1) \rceil + T(r) \Big), \tag{1}$$

for some (small) constant c'. Let $0 \leq i \leq \lceil N/r \rceil - 1$ and consider the subproblem (f_i, g'_i). Using at most $\lceil \log(N+1) \rceil + T(r)$ comparisons, we can find a claw in (f_i, g'_i) with probability at least $\frac{2}{3}$, provided there is one. We do that by using binary search to find the minimum j for which $g(j) \geq f(ir+1)$, at the cost of $\lceil \log(N+1) \rceil$ comparisons, and then recursively determining if the subproblem (f_i, g'_i) contains a claw at the cost of at most $T(r)$ additional comparisons. There are $2\left\lceil \frac{N}{r} \right\rceil$ subproblems, so by applying amplitude amplification we can find a claw among any one of them with probability at least $\frac{2}{3}$, provided there is one, in the number of comparisons given in equation (1).

We pick $r = \lceil \log^2(N) \rceil$. Since $T(r) \geq \Omega(\sqrt{r}) = \Omega(\log N)$, equation (1) implies

$$T(N) \leq c'' \sqrt{\frac{N}{r}} T(r), \tag{2}$$

for some constant c''. Furthermore, our choice of r implies that the depth of the recursion defined by equation (2) is on the order of $\log^\star(N)$, so unfolding the recursion gives the theorem. □

6 Hard problems related to distinctness

In this section, we consider some related problems for which quantum computers cannot improve upon classical (probabilistic) complexity.

Parity-collision problem

Given function $f : X \to Z$, find the parity of the cardinality of the set $C_f = \{(x, y) \in X \times X : x < y \text{ and } f(x) = f(y)\}$.

No-collision problem

Given function $f : X \to Z$, find an element $x \in X$ that is not involved in a collision (i.e., $f^{-1}(f(x)) = \{x\}$).

No-range problem

Given function $f : X \to Z$, find $z \in Z$ such that $z \notin f(X)$.

We assume that $X = Z = [N]$, and show that these problems are hard even for the function-evaluation model.

Theorem 8 *The evaluation-complexities of the parity-collision problem, the no-collision problem and the no-range problem are lower bounded by $\Omega(N)$.*

Note that the hardness of the parity-collision problem implies the hardness of exactly *counting* the number of collisions. Our proofs use the powerful lower bound method developed by Ambainis [2]. Let us state here exactly the result that we require.

Theorem 9 ([2]) *Let $\mathcal{F} = \{f : [N] \to [N]\}$ be the set of all possible input-functions, and $\Phi : \mathcal{F} \to Z$ be a function (which we want to compute). Let A, B be two subsets of $\mathcal{F}$ such that $\Phi(f) \neq \Phi(g)$ if $f \in A$ and $g \in B$, and $R \subseteq A \times B$ be a relation such that*

1. *For every $f \in A$, there exist at least m different $g \in B$ such that $(f, g) \in R$.*

2. *For every $g \in B$, there exist at least m' different $f \in A$ such that $(f, g) \in R$.*

3. *For every $f \in A$ and $x \in [N]$, there exist at most l different $g \in B$ such that $(f, g) \in R$ and $f(x) \neq g(x)$.*

4. *For every $g \in B$ and $x \in [N]$, there exist at most l' different $f \in A$ such that $(f, g) \in R$ and $f(x) \neq g(x)$.*

Then any quantum algorithm computing Φ with probability at least $2/3$ requires $\Omega(\sqrt{\frac{mm'}{ll'}})$ evaluation-queries.

We now give our proof of Theorem 8.

Proof To apply Theorem 9, we will describe a relation R for each of our problems. For functions $f : [N] \to [N]$ and $g : [N] \to [N]$, we denote by $d(f, g)$ the cardinality of the set $\{x \in [N] \mid f(x) \neq g(x)\}$. For each problem R will be defined by

$$R = \{(f, g) \in A \times B \mid d(f, g) = 1\},$$

for some appropriate sets A and B.

Parity-collision problem Here we suppose that 4 divides N. Let A be the set of functions $f : [N] \to [N]$ such that $|C_f| = N/4$ and $\left|f^{-1}(z)\right| \leq 2$ for all $z \in [N]$. Let B be the set of functions $g : [N] \to [N]$ such that $|C_g| = N/4 + 1$ and $\left|g^{-1}(z)\right| \leq 2$ for all $z \in [N]$. Then a simple computation gives that the relation R satisfies $m = \Theta(N^2)$, $m' = \Theta(N^2)$, $l = \Theta(N)$, and $l' = \Theta(N)$.

No-collision problem Now we suppose that N is odd. Let $A = B$ be the set of functions $f : [N] \to [N]$ such that $|C_f| = (N-1)/2$, and $\left|f^{-1}(z)\right| \leq 2$, for all $z \in [N]$. Then R satisfies that $m = m' = \Theta(N)$ and $l = l' = \Theta(1)$.

No-range problem Let $A = B$ be the set of functions $f : [N] \to [N]$ such that $C_f = \{(1, 2)\}$. Then a similar computation gives $m = m' = \Theta(N)$ and $l = l' = \Theta(1)$.

Note that the no-collision problem and the no-range problem are not functions in general (several outputs may be valid for one input), but that they *are* functions on the sets A and B chosen above (there is a unique correct output for each input). Thus, Theorem 9 implies a lower bound of $\Omega(N)$ for the evaluation-complexity of each of our three problems. □

7 Finding a triangle in a graph

Finally we consider a related search problem, which is to find a triangle in a graph, provided one exists. Consider an undirected graph $G = (V, E)$ on $|V| = n$ nodes with $m = |E|$ edges. There are $N = \binom{n}{2}$ edge slots in E, which we can query in a black box fashion (see also [11, Section 7]). The triangle-finding problem is:

Triangle-finding problem
Given undirected graph $G = (V, E)$, find distinct vertices $a, b, c \in V$ such that $(a, b), (a, c), (b, c) \in E$.

Since there are $\binom{n}{3} < n^3$ triples a, b, c, and we can decide whether a given triple is a triangle using 3 queries, we can use Grover's algorithm to find a triangle in $O(n^{3/2})$ queries. Below we give an algorithm that works more efficiently for sparse graphs.

Algorithm: Triangle-finder

1. Use quantum search to find an edge $(a, b) \in E$ among all $\binom{n}{2}$ potential edges.

2. Use quantum search to find a node $c \in V$ such that a, b, c is a triangle.

3. Apply amplitude amplification on steps 1–2.

Step 1 takes $O(\sqrt{n^2/m})$ queries and step 2 takes $O(\sqrt{n})$ queries. If there is a triangle in the graph, then the probability that step 1 finds an edge belonging to this specific triangle is $\Theta(1/m)$. If step 1 indeed finds an edge of a triangle, then with probability at least 1/2, step 2 finds a c that completes the triangle. Thus the success probability of steps 1–2 is $\Theta(1/m)$ and the amplitude amplification of step 3 requires $O(\sqrt{m})$ iterations. The total complexity is hence $O((\sqrt{n^2/m}+\sqrt{n})\sqrt{m})$ which is $O(n+\sqrt{nm})$. If G is sparse in the sense that $m = |E| \in o(n^2)$, then $o(n^{3/2})$ queries suffice. Of course for dense graphs our algorithm will still require $\Theta(n^{3/2})$ queries.

We again obtain lower bounds by a reduction from the **OR**-problem. Consider an **OR**-input $X \in \{0, 1\}^{\binom{n}{2}}$ as a graph on n edges. Let G be the graph obtained from this by adding an $(n + 1)$-th node and connecting this to all other n nodes. Now G has $|X| + n$ edges, and **OR**$(X) = 1$ if and only if G contains a triangle. This gives $\Omega(n^2)$ bounds for exact quantum and bounded-error classical, and an $\Omega(n)$ bound for bounded-error quantum. We have shown:

Theorem 10 *If $\Omega(n) \leq |E| \leq \binom{n}{2}$, then the edge-query-complexity of triangle-finding is*

- $\Omega(n) \leq Q_2(\textbf{Triangle}) \leq O(n + \sqrt{nm})$
- $Q_E(\textbf{Triangle}) = \Theta(n^2)$

where $n = |V|$ and $m = |E|$ for $G = (V, E)$.

Note that for graphs with $\Theta(n)$ edges, the bounded-error quantum bound becomes $\Theta(n)$ queries, whereas the classical bound remains $\Theta(n^2)$. Thus we have a quadratic gap for such very sparse graphs.

8 Concluding remarks

The main problem left open by this research is to close the gap between upper and lower bounds for element distinctness. We find it hard to conjecture what the true bound is. None of the known methods for proving quantum lower bounds seem to be directly applicable to improve the $\Omega(\sqrt{N})$ lower bound, and we feel that if element distinctness is strictly harder than unordered search, then proving it will require new ideas.

An interesting direction could be to take into account simultaneously *time* complexity and *space* complexity, as has been done for classical algorithms by Yao [21], Ajtai [1], Beame, Saks, Sun, and Vee [4], and others. In particular, Yao shows that the time-space product of any classical deterministic comparison-based branching program solving element distinctness satisfies $T \cdot S \geq \Omega(N^{2-\varepsilon(N)})$,where $\varepsilon(N) = 5/\sqrt{\ln N}$. A possible extension of this result to quantum computation could be that the time T and space S of any quantum bounded-error algorithm for element distinctness satisfies

$$T^2 \cdot S \geq \Omega(N^{2-\varepsilon(N)}), \tag{3}$$

for some appropriate function $\varepsilon(N) = o(N)$. For the algorithms of this paper, the comparison complexity and the time complexity are equal up to logarithmic factors. Ignoring such logarithmic factors, the algorithm for element distinctness we presented in Section 3 has $T = N^{3/4}$ and $S = N^{1/2}$ and hence would satisfy Equation (3) up to logarithmic factors. An alternative quantum algorithm is to search the space of all $\binom{N}{2}$ (x, y)-pairs to try and find a collision. This algorithm has roughly $T = N$ and $S = \log N$ and hence also satisfies Equation (3) up to logarithmic factors. In fact, for all other (T, S)-pairs in between these two cases (i.e., with $\log N \leq S \leq \sqrt{N}$ and $T^2 \cdot S \geq N^2$) there exists an analogous quantum algorithm with roughly those amount of time and space as well.

References

[1] M. Ajtai. Determinism versus non-determinism for linear time RAMs. In *Proceedings of 31st ACM STOC*, pages 632–641, 1999.

[2] A. Ambainis. Quantum lower bounds by quantum arguments. In *Proceedings of 32nd ACM STOC*, pages 636–643, 2000. quant-ph/0002066.

[3] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. In *Proceedings of 39th IEEE FOCS*, pages 352–361, 1998. quant-ph/9802049.

[4] P. Beame, M. Saks, X. Sun, and E. Vee. Super-linear time-space tradeoff lower bounds for randomized computation. In *Proceedings of 41st IEEE FOCS*, pages 169–179, 2000. Available at ECCC TR00–025.

[5] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997. quant-ph/9701001.

[6] G. Brassard and P. Høyer. An exact quantum polynomial-time algorithm for Simon's problem. In *Proceedings of the 5th Israeli Symposium on Theory of Computing and Systems (ISTCS'97)*, pages 12–23, 1997. quant-ph/9704027.

[7] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. quant-ph/0005055. This the upcoming journal version of [9, 17], 15 May 2000.

[8] G. Brassard, P. Høyer, and A. Tapp. Quantum algorithm for the collision problem. *ACM SIGACT News (Cryptology Column)*, 28:14–19, 1997. quant-ph/9705002.

[9] G. Brassard, P. Høyer, and A. Tapp. Quantum counting. In *Proceedings of 25th ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 820–831. Springer, 1998. quant-ph/9805082.

[10] H. Buhrman, R. Cleve, and A. Wigderson. Quantum vs. classical communication and computation. In *Proceedings of 30th ACM STOC*, pages 63–68, 1998. quant-ph/9802040.

[11] H. Buhrman, R. Cleve, R. de Wolf, and Ch. Zalka. Bounds for small-error and zero-error quantum algorithms. In *Proceedings of 40th IEEE FOCS*, pages 358–368, 1999. cs.CC/9904019.

[12] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Invariant quantum algorithms for insertion into an ordered list. quant-ph/9901059, 19 Jan 1999.

[13] D. Grigoriev. Randomized complexity lower bounds. In *Proceedings of 30th ACM STOC*, pages 219–223, 1998.

[14] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th ACM STOC*, pages 212–219, 1996. quant-ph/9605043.

[15] P. Høyer, J. Neerbek, and Y. Shi. Quantum bounds for ordered searching and sorting. quant-ph/0102078, 15 Feb 2001.

[16] A. Lubiw and A. Rácz. A lower bound for the integer element distinctiveness problem. *Information and Control*, 94(1):83–92, 1991.

[17] M. Mosca. Quantum searching, counting and amplitude amplification by eigenvector analysis. In *MFCS'98 workshop on Randomized Algorithms*, 1998.

[18] A. Nayak and F. Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of 31st ACM STOC*, pages 384–393, 1999. quant-ph/9804066.

[19] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[20] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. Earlier version in FOCS'94. quant-ph/9508027.

[21] A. C.-C. Yao. Near-optimal time-space tradeoff for element distinctness. *SIAM Journal on Computing*, 23(5):966–975, 1994. Earlier version in FOCS'88.