SIGACT News Complexity Theory Column 44

Lane A. Hemaspaandra
Dept. of Computer Science, University of Rochester
Rochester, NY 14627, USA lane@cs.rochester.edu

Introduction to Complexity Theory Column 44

This issue's column is by Andris Ambainis, on quantum serach algorithms. Warmest thanks to the Andris, and best wishes to all for a theorem-filled summer.

The columns to appear in this year's remaining issues will be by Venkatesan Guruswami, on codes and expanders, and by John Hitchcock, Jack Lutz, and Elvira Mayordomo, on the fractal geometry of complexity classes.

Guest Column: Quantum Search Algorithms[1]

A. Ambainis[2]

Abstract

We review some of quantum algorithms for search problems: Grover's search algorithm, its generalization to amplitude amplification, the applications of amplitude amplification to various problems and the recent quantum algorithms based on quantum walks.

1 Introduction

Quantum computation explores the possibilities of applying quantum mechanics to computer science. If built, quantum computers would provide speedups over conventional computers for a variety of problems. The two most famous results in this area are Shor's quantum algorithms for factoring and finding discrete logarithms [35] and Grover's search algorithm [22].

Shor's and Grover's algorithms have been followed by a lot of other results. Each of these two algorithms has been generalized and applied to several other problems. New algorithms and new algorithmic paradigms (such as adiabatic computing[21] which is the quantum counterpart of simulated annealing) have been discovered.

In this column, we survey some of the results on quantum algorithms, focusing on the branch of quantum algorithms inspired by Grover's search algorithm [22].

[2] School of Mathematics, Institute for Advanced Study, Princeton, NJ 08540, USA. ambainis@ias.edu. Supported by NSF Grant DMS-0111298. Any opinions, findings and conclusions expressed in this material are those of the author and do not necessarily reflect views of the National Science Foundation.

Instead of the conventional introduction/review on quantum computing which starts with the backgrounds from physics, we follow a different path. We first describe Grover's search result and its generalization, *amplitude amplification* (section 2). Then, we explore what can be obtained by using these results as "quantum black boxes" in a combination with methods from conventional (non-quantum) algorithms and complexity (section 3). We give three examples of quantum algorithms of this type, one very simple and two more advanced ones. After that, in section 4, we show some examples were simple application of Grover's search fails but more advanced quantum algorithms (based on quantum walks) succeed.

2 Grover's search and amplitude amplification

Grover's search algorithm is one of main quantum algorithms. The problem that it solves is very simple to state:

Search. We have an input $x_1, \ldots, x_N \in \{0, 1\}$ specified by a black box that answers queries. In a query, we input i to the black box and it outputs x_i. Our task[3] is to output an $i : x_i = 1$.

Then, N queries are needed for deterministic algorithms and $\Omega(N)$ queries are needed for probabilistic algorithms. (This follows by considering the case when there is exactly one i such that $x_i = 1$ and $N - 1$ variables $i{:}x_i = 0$.)

Grover [22] studied the quantum version of this problem (in which the black box is quantum, the input to the black box is a quantum state consisting of various i and the output is the input state modified depending on x_i). His result is

Theorem 2.1 *[22]* **Search** *can be solved with $O(\sqrt{N})$ quantum queries.*

2.1 Is it "database search"?

Grover's algorithm is often called "database search". In this interpretation, the variables $x_1, \ldots, x_N$ correspond to N entries of the database. A variable is 1 if the corresponding entry of database matches our search criteria. Then, the result is that we can search an unordered database of N entries in time $O(\sqrt{N})$.

This interpretation has caused some heated debates. Essentially, the issue is that, to access N elements, we would need quantum hardware of size $\Omega(N)$ [30, section 6.5]. Since quantum hardware is likely to be expensive, this may be a big obstacle.

Nevertheless, Grover's algorithm can be very useful in problems of a different nature. Say we have an instance of an NP-complete problem, for example, satisfiability. That is, we have a boolean formula $F(y_1, \ldots, y_n)$ and we want to know if one of 2^n assignments $y = (y_1, \ldots, y_n)$ of values to the variables makes F true. The naive exhaustive search requires testing 2^n assignments.

With a quantum computer, we could instead reduce the satisfiability to search on $N = 2^n$ variables $x_1, \ldots, x_N$ with $x_i = 1$ if F is true for the i^{th} candidate assignment $y = (y_1, \ldots, y_n)$. The black box that answers queries is just a circuit that takes an assignment $(y_1, \ldots, y_n)$ and checks if F is true on this assignment. Then, Grover's algorithm allows to solve satisfiability in

[3] There are several variations of this problem. We could either require the algorithm to output "none" if there is no $i : x_i = 1$ or allow any output in this situation. Or we could consider a decision version, where the algorithm only has to determine if there exists $i : x_i = 1$ instead of finding i. The complexity remains almost the same for all variations.

time $O(\sqrt{N}) = O(1.41...^n)$ instead of $N = 2^n$. Similar approach applies to any exhaustive search problem.

A knowledgeable reader might point out that 3-SAT can be solved even faster classically, in time $O(1.329...^n)$ by a non-naive algorithm [33, 25, 31]. We address this issue in section 3.1.

2.2 Facts about Grover's algorithm

Since Grover's result, the search problem has been analyzed in great detail. Here are some of results that we know:

1. In general, Grover's algorithm is bounded-error. Given a black-box $x_1, \ldots, x_N \in \{0,1\}$ where some x_i are equal to 1, the algorithm might not find any of them with a small probability. However, if we know that the number of $i : x_i = 1$ is exactly k, then the algorithm can be tuned so that it finds one of them with certainty (probability 1) in $O(\sqrt{N/k})$ steps [14].

2. Moreover, if we know that the number of $i : x_i = 1$ is exactly k, the algorithm is *exactly* optimal [37]. The number of queries cannot be improved even by 1. For finding $i : x_i = 1$ with certainty, the minimum number of queries is known to be exactly

$$\left\lceil \frac{\pi}{4 \arcsin \frac{1}{\sqrt{N/k}}} - \frac{1}{2} \right\rceil < \frac{\pi}{4}\sqrt{\frac{N}{k}} \tag{1}$$

   If the number of queries t is less than that, the best probability with which any quantum algorithm can find an $i : x_i = 1$ is exactly the one achieved by running Grover's algorithm with t queries.

3. If k is unknown, $O(\sqrt{N})$ queries are still sufficient. If k is unknown but it is known that $k \geq k_0$, $O(\sqrt{N/k_0})$ queries suffice [11].

4. In this case, the algorithm is inherently bounded-error. There is no quantum algorithm with less than N queries that solves Grover's problem with certainty for arbitrary $x_1, \ldots, x_N$ [9].

If we have an instance $x_1, \ldots, x_N$ with k elements equal to 1 and would like to find all k of them, $\Theta(\sqrt{Nk})$ queries are sufficient and necessary.

2.3 Amplitude amplification

Let A be a (classical or quantum) algorithm with one sided error. If the correct answer is "no", A always outputs "no". If the correct answer is "yes", A outputs "yes" with at least some (small) probability $\epsilon > 0$.

An example is an algorithm for SAT (or any other problem in NP) which outputs "the formula is satisfiable" only if it finds a satisfying assignment. Or, a different example is an algorithm for Grover's search problem which outputs "there exists $i : x_i = 1$" only if it has found such i.

How many times do we need to repeat the algorithm to increase its success probability from a small ϵ to a constant (for example, $2/3$)? algorithm to increase the success probability? Classically, $\Theta(1/\epsilon)$ repetitions are needed. In quantum case, a generalization of Grover's algorithm gives

Theorem 2.2 *[14] Let A be a quantum algorithm with one-sided error and success probability at least $\epsilon > 0$. Then, there is a quantum algorithm B that solves the same problem with success probability 2/3 by invoking A $O(\frac{1}{\sqrt{\epsilon}})$ times.*

This result is called *amplitude amplification*. For more details, see [14]. Similar result is also known for algorithms with two-sided error but it has not found as many applications as amplitude amplification for algorithms with one-sided error.

3 Three applications

In this section, we show 3 examples how Grover's algorithm and amplitude amplification can be used to solve other problems. The examples are selected to be solvable by just two ideas from quantum computation (and some algorithmic ingenuity). The first of two ideas is Grover's search and amplitude amplification, described in the previous section. The second idea is that any classical (either deterministic or probabilistic) computation can be simulated on a quantum computer [30, section 1.4]. More precisely,

- In the circuit model, a classical circuit with N gates can be simulated by a quantum circuit with $O(N)$ gates.
- If the query model (when only the number of queries is counted), a classical computation with N queries can be simulated by a quantum computation with N queries.

This greatly simplifies descriptions of quantum algorithms. Instead of describing a quantum algorithm, we can describe a classical algorithm that succeeds with some small probability ϵ. Then, we can transform the classical algorithm to a quantum algorithm and apply the amplitude amplification to the quantum algorithm. The result is a quantum algorithm with the running time or the number of queries that is $O(\frac{1}{\sqrt{\epsilon}})$ times the one for the classical algorithm with which we started.

A similar reasoning can be applied, if instead of a purely classical algorithm, we started with a classical algorithm that involves quantum subroutines. Such algorithms can also be transformed into quantum algorithms with the same complexity.

3.1 3-satisfiability

As we described in section 2.1, Grover's algorithm can solve 3-satisfiability in $O(1.41...^n poly(n))$ steps. However, the best known classical algorithm for 3-satisfiability is faster than that, running in time $O(1.329...^n)$ [31]. Does this mean that Grover's algorithm is not useful for satisfiability?

Not quite. The best classical algorithm can be combined with Grover's search. The result is a quantum algorithm that runs in time $O(1.153...^n poly(n))$, providing a square-root-speedup over [31].

We first describe the classical algorithm (due to Schöning [33], improved by [25, 31]). Its structure is as follows:

1. Pick a random initial assignment $x_1, \ldots, x_n$.
2. $3n$ times repeat:
   (a) If all clauses satisfied, stop.

(b) Otherwise, find an unsatisfied clause. Make it satisfied by choosing a variable in this clause uniformly at random and changing its value.

The result of [31] is that, for an appropriate initial probability distribution in the first step, the algorithm finds a satisfying assignment (if there is one) with a probability at least c^n (where $c = \frac{1}{1.329...}$). Repeating the algorithm $1.329...^n$ times gives an algorithm that finds a satisfying assignment in time $O(1.329...^n poly(n))$ with a constant success probability.

To obtain a quantum algorithm, we just use quantum amplitude amplification instead of classical repetition. As described in section 2.3, amplitude amplification allows to increase the success probability to a constant, repeating the algorithm $O(\frac{1}{\sqrt{\epsilon}})$ times. In this case, this means $O(\sqrt{1.329...^n}) = O(1.153...^n)$ repetitions.

The result is very simple but it illustrates an important point. For some problems, Grover's algorithm can provide a quadratic speedup not just over the naive classical algorithm (testing all assignments) but over better classical algorithms as well.

3.2 Element distinctness

Element Distinctness. We are given $f : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, N\}$ specified by a black box that, given i, answers the value of $f(i)$. The task is to determine if there are two inputs i, j, $i \neq j$ for which $f(i) = f(j)$.

The measure of complexity is the number of queries to the black box. Classically, this problem requires $\Omega(N)$ queries. In quantum case, there are two algorithms. The first, due to [15] uses Grover search in a clever two-level construction and solves the problem $O(N^{3/4})$ queries. The second, due to [5], uses a technique combining search with quantum walks and solves the problem with $O(N^{2/3})$ queries. This is optimal, because of an $\Omega(N^{2/3})$ lower bound by Shi [34].

In this section, we show the $O(N^{3/4})$ algorithm by Buhrman et.al. [15]. While the result is weaker than the later algorithm of [5], the idea is very elegant. Consider the following algorithm:

1. Choose $\sqrt{N}$ random numbers $i_1, \ldots, i_{\sqrt{N}} \in \{1, 2, \ldots, N\}$. Evaluate $f(i_1)$, ..., $f(i_{\sqrt{N}})$. If two of them are equal, stop, output the two equal elements.

2. Use Grover's search to search (among remaining $N - \sqrt{N}$ indices $k \in \{1, 2, \ldots, N\}$) for an index k such that $f(k) = f(i_j)$ for some j.

This algorithm requires $\sqrt{N}$ queries for the first step and $O(\sqrt{N})$ queries for the second step. (Notice that we do not need to query $f(i_j)$ since their values are known from the first step.) The total number is $O(\sqrt{N})$.

If there is a pair i, j such that $f(i) = f(j)$, then, with probability $\frac{\sqrt{N}}{N} = \frac{1}{\sqrt{N}}$, i is among $i_1, \ldots, i_{\sqrt{N}}$. In this case, the second step will find j (or some other element k such that $f(k)$ is equal to one of $f(i_1)$, ..., $f(i_{\sqrt{N}})$) a constant probability. Thus, the algorithm succeeds with probability at least $\frac{const}{\sqrt{N}}$.

We can now apply the amplitude amplification, described in section 2.3. It increases the success probability to a constant with $O(\frac{1}{\sqrt{\epsilon}})$ repetitions of the whole algorithm. Since $\epsilon = \frac{const}{\sqrt{N}}$, $O(N^{1/4})$ repetitions suffice. The total number of queries needed is $O(N^{1/4}N^{1/2}) = O(N^{3/4})$.

3.3 Finding global and local minima

In this section, we describe quantum algorithms for two minimum-finding problems.

Global Minimum. We have an integer-valued function $f(i)$ of one variable $i \in \{1, 2, \ldots, N\}$, specified by black box that answers queries. The input of a query is $i \in \{1, 2, \ldots, N\}$, the output is $f(i)$. The task is to find i such that $f(i) \leq f(j)$ for any $j \neq i$.

Local Minimum. We have an integer-valued function $f(x_1, \ldots, x_n)$ of Boolean variables $x_1, \ldots, x_n \in \{0, 1\}$, specified by black box that answers queries. The input of a query is $x_1, \ldots, x_n \in \{0, 1\}$, the output is $f(x_1, \ldots, x_n)$. The task is to find a local minimum: an assignment $x_1, \ldots, x_n$ such that changing any one variable does not decrease the value of the function:

$$f(x_1, \ldots, x_{i-1}, 1 - x_i, x_{i+1}, \ldots, x_n) \geq f(x_1, \ldots, x_n).$$

In both cases, the measure of complexity is the number of queries (i.e. the number of times that we need to evaluate f). We start by describing an algorithm for the first problem, which will be used as a subroutine in the second algorithm.

Theorem 3.1 *[19]* **Global Minimum** *can be solved with $O(\sqrt{N})$ quantum queries.*

Classically, $\Omega(N)$ queries are required.

The outline of the algorithm is as follows (some technical details are omitted, to simplify the presentation):

1. Choose x uniformly at random from $\{1, \ldots, N\}$;
2. Repeat:
   (a) Use Grover's search to search for y with $f(y) < f(x)$;
   (b) If search succeeds, set $x = y$. Otherwise, stop and output x as the minimum.

We sketch why $O(\sqrt{N})$ queries are sufficient for this algorithm, on intuitive but "hand-waving" level. (For a more detailed and rigorous argument, see [19].) For simplicity, assume that, for all x, the values of $f(x)$ are distinct. Let x_0 be the value of x at the beginning of the algorithm and x_i be the value of x after the i^{th} Grover's search. Since x_0 is a random element of $\{1, \ldots, N\}$, $f(x_0)$ will, on average, be the $(N/2)^{\text{th}}$ smallest element of $\{f(1), \ldots, f(N)\}$. After the first iteration, x_1 is some element with $f(x_1) < f(x_0)$. By inspecting Grover's algorithm, we can find out that the probabilities of algorithm outputting x_1 are equal for all x_1 with $f(x_1) < f(x_0)$. Thus, x_1 is uniformly random among numbers with $f(x_1) < f(x_0)$. Since $f(x_0)$ was, on average, be the $(N/2)^{\text{th}}$ smallest element of $\{f(1), \ldots, f(N)\}$, this means that $f(x_1)$ is, on average, the $(N/4)^{\text{th}}$ smallest element. By a similar argument, $f(x_i)$ is, on average the $(N/2^i)^{\text{th}}$ smallest element in $\{f(1), \ldots, f(N)\}$.

We now remember that Grover's search uses $O(\sqrt{N/k})$ queries where k is the number of solutions. Consider repetitions of the minimum finding algorithm in the order from the last to the first. By the argument above, we would expect that, in the last iteration before finding the minimum, $k \approx 1$, then, in the iteration before that, $k \approx 2$, then $k \approx 4$ and so on. Then, the total number of queries in all the repetitions of Grover's search is of order

$$\sqrt{N} + \sqrt{N/2} + \sqrt{N/4} + \ldots = \sqrt{N}\left(1 + \frac{1}{\sqrt{2}} + \frac{1}{2} + \ldots\right). \tag{2}$$

The term in brackets is a decreasing geometric progression and, therefore, sums up to a constant. This means that the sum of equation (2) is of order $O(\sqrt{N})$.

We now turn to algorithms for **Local Minimum**. Classically, $\Theta(2^{n/2} poly(n))$ queries are necessary and sufficient [3, 1]. In quantum case,

Theorem 3.2 *[1]* **Local minimum** *can be solved with $O(2^{n/3} n^{1/6})$ quantum queries.*

The algorithm (again, in a simplified form) is as follows:

1. Choose m assignments $x = (x_1, \ldots, x_n)$ uniformly at random. Use Grover's search to find one with the smallest value of $f(x_1, \ldots, x_n)$.
2. $2^{n+1}/m$ times repeat:
   (a) Use Grover's search to search for $(y_1, \ldots, y_n)$ with $f(y_1, \ldots, y_n) < f(x_1, \ldots, x_n)$, among n assignments $(y_1, \ldots, y_n)$ that differ from $(x_1, \ldots, x_n)$ in exactly one variable.
   (b) If such $(y_1, \ldots, y_n)$ is found, set $(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$. Otherwise, stop and claim that $(x_1, \ldots, x_n)$ is a local minimum.

The first step requires $O(\sqrt{m})$ queries. The second step requires $O(\sqrt{n})$ queries each time it is repeated. The total number is $O(\sqrt{m} + \frac{2^n}{m}\sqrt{n})$. The minimum of this expression is $O(2^{n/3} n^{1/6})$ which is achieved by setting $m = 2^{2n/3} n^{1/3}$.

The correctness of the algorithm follows from the fact that, if we pick m elements out of 2^n, then, with high probability, one of those m elements will be among $2^{n+1}/m$ smallest among all 2^n elements (see [1] for a proof). If this is the case, the minimum of m elements is also among $2^{n+1}/m$ smallest elements among all 2^n elements. Then, the second step of the algorithm will lead to a local minimum in at most $2^{n+1}/m$ steps because each step replaces $(x_1, \ldots, x_n)$ by an assignment with a smaller value of f and there are at most $2^{n+1}/m$ assignments for which f has smaller value than for the starting point.

4 Local search and quantum walks

4.1 Two problems

Next, we show two situations when a simple application of Grover's algorithm does not give a good quantum algorithm.

Search on grid. Consider N memory cells, arranged into $\sqrt{N} \times \sqrt{N}$ grid. Each cell stores an element $x_i \in \{0, 1\}$. Our task is to find an element $i : x_i = 1$. At each moment of time, we are in some memory location. In one time step, we can either query the current location or move to an adjacent cell. (In the quantum version, we can be in a quantum state consisting of various locations. But we still require that no part of this state moves more than distance 1 in one time unit.)

Grover's algorithm finds an element $i : x_i = 1$ with $O(\sqrt{N})$ queries. But, between any two queries, it needs $\Theta(\sqrt{N})$ moves, since a query to one element can be followed by a query to any other element. The total number of steps is of order $\sqrt{N} \times \sqrt{N} = N$. A similar number of steps can be achieved by a classical algorithm that just traverses the grid row by row and queries every cell. That takes $O(N)$ moves and $O(N)$ queries, for a total of $O(N)$ steps as well. The quantum advantage seems to disappear [10].

If the N items are arranged in 3 dimensions, in a cube with side of length $O(\sqrt[3]{N})$, then the straightforward quantum search takes $O(\sqrt{N}\sqrt[3]{N}) = O(N^{5/6})$ which is better than classical $O(N)$ but still worse than $O(\sqrt{N})$ in the usual Grover's search when only queries are counted.

Searching the element distinctness graph. Element distinctness reduces to search a certain graph. Let $1 \leq M < N$. Define a bipartite graph, with the vertices being all subsets of $\{1, 2, \ldots, N\}$ of size m and size $m+1$. A vertex v_S corresponding to a subset S is connected to a vertex v_T if $|S| = M$, $|T| = M+1$ and $T = S \cup \{i\}$ for some $i \in \{1, 2, \ldots, N\}$. A vertex v_S is marked if the set S contains i, j such that $i \neq j$ and $x_i = x_j$. The task is to find a marked vertex. In one step, we are allowed to examine the current vertex or to move to an adjacent vertex.

If we can search this graph in $f(N)$ steps, we can solve element distinctness with at most $f(N) + M$ queries, in a following way. If we are at a vertex v_S, we will know the values of all x_i, $i \in S$. Then, testing if a vertex is marked can be done with no queries and moving to an adjacent vertex v_T can be done with 1 query by querying the only element $i \in T - S$. To achieve that, we use the first M queries to query all x_i for $i \in S$ where S is the set corresponding to the starting vertex v_S. The total number of queries is M to start the algorithm and at most one query per search algorithm step afterwards.

Again, let us try to use Grover's algorithm to search this graph. If there is exactly one pair of equal elements $x_i = x_j$, then the probability of a random vertex v_S, $|S| = M$ being marked is

$$Pr[i \in S]Pr[j \in S | i \in S] = \frac{M}{N}\frac{M-1}{N-1} = (1+o(1))\frac{M^2}{N^2}$$

Amplitude amplification implies that we can find such set S by testing $O(\frac{N}{M})$ vertices which is a square root of what one would need classically. However, testing each vertex v_S involves querying M elements. The total number of queries is of order $\frac{N}{M}M = N$ which is the same as if we just queried all N elements x_i to begin with.

Both examples illustrate the same general situation. Sometimes, we have a search space, where after testing an item, it is faster to test a neighboring item than an arbitrary item. This could come either from physical constraints on the search space (the first example) or an algorithmic structure (the second example). A straightforward application of Grover's algorithm does not do well on such spaces, because it does not respect the structure of the space.

4.2 Solution to two problems

There is a recent approach, based on quantum walks (quantum counterparts of random walks) that overcomes this problem. (For more information on quantum walks, see the surveys [26, 6].)

Theorem 4.1 *[7]* **Spatial search** *can be solved with*

1. $O(\sqrt{N}\log N)$ *steps in 2 dimensions, if there is a unique* $i : x_i = 1$*;*
2. $O(\sqrt{N}\log^2 N)$ *steps in 2 dimensions in the general case (no assumptions on the number of* $i : x_i = 1$*);*
3. $O(\sqrt{N})$ *steps in 3 and more dimensions.*

Quantum walks also give a better search algorithm for the element distinctness graph. It can be searched in $O(N/\sqrt{M})$ steps, implying an algorithm for element distinctness with $O(M+\frac{N}{\sqrt{M}})$ steps. Setting $M=N^{2/3}$ gives

Theorem 4.2 *[5]* **Element distinctness** *can be solved with $O(N^{2/3})$ queries.*

For the first result (spatial search), there is a different solution that involves amplitude amplification instead of quantum walks [2], giving $O(\sqrt{N})$ in 3 dimensions and $O(\sqrt{N}\log^2 N)$ and $O(\sqrt{N}\log^3 N)$ in the two 2-dimensional cases. For element distinctness, quantum walks are the only approach known to give $O(N^{2/3})$ algorithm.

Szegedy [36] has generalized element distinctness and spatial search, by showing how to convert a general classical Markov chain into a quantum walk algorithm. His generalization of element distinctness is

Theorem 4.3 *[36] Let P be a symmetric Markov chain with a gap between the first and the second eigenvalue being ϵ and at least δ fraction of states being marked. Assume that we can perform the following operations:*

- *generate a uniformly random element in γ_0 steps;*
- *given a state x, generate a sample from $P(x,y)$ in γ_1 steps;*
- *check if a state is marked in γ_2 steps.*

Then, there is a quantum algorithm that finds a marked state in $O(\gamma_0+\frac{1}{\sqrt{\delta\epsilon}}(\gamma_1+\gamma_2))$ steps.

For comparison, a classical random walk would find a marked state in $O(\gamma_0+\frac{1}{\delta\epsilon}(\gamma_1+\gamma_2))$ steps.

The element distinctness algorithm is a special case of Theorem 4.3 where the states of Markov chain are sets S, $|S|=M$ or $|S|=M+1$ and, at each step, the Markov chain adds (if $|S|=M$) or removes (if $|S|=M+1$) a random element from the set.

4.3 Applications of element distinctness

There are several results that build on element distinctness algorithm [27, 18, 16].

Triangle finding. A graph G with N vertices is specified by $\binom{N}{2}$ variables x_{ij} with $x_{ij}=1$ if there is an edge between vertices i and j. The access to x_{ij} is by queries to a black box. The task is to find if the graph G contains a triangle.

Theorem 4.4 *[27]* **Triangle finding** *can be solved with $O(N^{1.3})$ quantum queries.*

The construction [27] uses element distinctness as a subroutine in a clever two-level construction reminiscent of the $O(N^{3/4})$ algorithm for element distinctness in section 3.2. Another problem for which element distinctness is useful as a subroutine is

Matrix product. Three $N\times N$ Boolean matrices A, B and C are specified by variables a_{ij}, b_{ij}, c_{ij}, n^2 variables per matrix. The access to the variables is by queries to a black box. The task is to find if $AB=C$, with the arithmetic operations modulo 2.

Theorem 4.5 *[16]* **Matrix product** *can be solved with $O(N^{1.75})$ quantum queries.*

5 Conclusion and open problems

In this column, we reviewed some of quantum algorithms for search problems: Grover's search, amplitude amplification, their applications to NP-complete problems, element distinctness and finding local and global minima, and improved quantum search algorithms using quantum walks.

There are other interesting results that share similar ideas or use the number of queries as the complexity measure. To mention a few, [12] have constructed a quantum algorithm for collision problem, [13, 23, 29] have given quantum algorithms for approximate counting, finding mean and median, [24] studied quantum complexity of searching among N ordered items and sorting and there is a large amount of work on quantum lower bounds (e.g. [8, 9, 4]).

We conclude with some related open problems.

1. **Complexity of graph problems.** Complexity of several graph problems remains open in the query model. First, can the $O(N^{1.3})$ query triangle algorithm be improved? The best lower bound for this problem is $\Omega(N)$ (folklore). Second, what is the query complexity of finding a matching in a bipartite graph G with N vertices on each side, specified by N^2 variables? There is an $\Omega(N^{1.5})$ lower bound but no quantum algorithm that uses $o(N^2)$ queries.

2. **Generalizing quantum walk algorithms.** As we saw in section 3, amplitude amplification provides an easy way to apply Grover's technique to various problem without going into details of Grover's search algorithm. Quantum walk results (theorems 4.1 and 4.2) share common proof ideas. Can we find an generalization for these two results which would be as easy-to-use as amplitude amplification?

   Results of Szegedy [36] (e.g. Theorem 4.3) are a major advance in this direction.

3. **Space usage in element distinctness.** Both known algorithms for element distinctness use considerable amounts of memory which has caused some criticism [32]. The $O(N^{3/4})$ algorithm of [15] stores values of $O(\sqrt{N})$ variables and the $O(N^{2/3})$ algorithm of [5] stores $O(N^{2/3})$ variables. Is it possible to design a quantum algorithm that uses less space? Or can we prove a time-space lower bound saying that there is no algorithm with better space usage for the given number of queries?

4. **Quantum-classical relations.** The quantum speedups described in this column are polynomial rather than exponential (as in Shor's factoring algorithm). This is inherent for a wide class of problems. Consider computing a total Boolean function $f(x_1, \ldots, x_N)$, with the variables $x_1, \ldots, x_N$ given by a black box that answers queries (as in most problems described in this column). Let $D(f)$ be the number of queries needed to compute f deterministically and $Q(f)$ be the number of queries needed to compute f by a (bounded-error) quantum algorithm. Then, $D(f)$ and $Q(f)$ are polynomially related: $D(f) = O(Q^6(f))$ [9].

   The open question is: what is the biggest possible gap between $D(f)$ and $Q(f)$? The best known result is for Grover's search problem: $D(f) = N$, $Q(f) = \Theta(\sqrt{N})$, $D(f) = \Theta(Q^2(f))$. Can we find f with a bigger gap or improve the $D(f) = O(Q^6(f))$ relation?

   A similar problem is open if we consider $Q_E(f)$, the number of queries needed by the best *exact* quantum algorithm (an exact algorithm is one which gives correct answer with certainty, instead of probability $1-\epsilon$). Then, we know that $D(f) = O(Q_E^3(f))$ [28] but there is no known example for which $Q_E(f) = o(D(f))$. For more information on this topic, we refer the reader to an excellent survey by Buhrman and de Wolf [17].

References

[1] S. Aaronson. Lower bounds for local search by quantum arguments. *Proceedings of STOC'04*, to appear, also[4] quant-ph/0307149.

[2] S. Aaronson and A. Ambainis. Quantum search of spatial regions. In *Proceedings of FOCS'03*, pp. 200–209, 2003. Also quant-ph/0303041.

[3] D. Aldous. Minimization algorithm and random walk on the d-cube. *Annals of Probability*, 11(2):403-413, 1983.

[4] A. Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4): 750-767, 2002. Also STOC'00 and quant-ph/0002066.

[5] A. Ambainis. Quantum walk algorithm for element distinctness. quant-ph/03110001.

[6] A. Ambainis. Quantum walks and their algorithmic applications. quant-ph/0403120.

[7] A. Ambainis, J. Kempe, A. Rivosh. Coins make quantum walks faster. quant-ph/0402107.

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

[9] R. Beals, H. Buhrman, R. Cleve, M. Mosca, R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4): 778-797, 2001. Also FOCS'98 and quant-ph/9802049.

[10] P. Benioff. Space searches with a quantum robot. In *Quantum computation and information (Washington, DC, 2000)*, volume 305 of AMS Series on Contemporary Mathematics, pp. 1–12. Also quant-ph/0003006.

[11] M. Boyer, G. Brassard, P. Høyer, A. Tapp. Tight bounds on quantum searching. Fortsch.Phys. 46:493-506, 1998. Also quant-ph/9605034.

[12] G. Brassard, P. Høyer, A. Tapp. Quantum cryptanalysis of hash and claw-free functions. *Proceedings of LATIN'98*, pp. 163-169. Also quant-ph/9705002.

[13] G. Brassard, P. Høyer, A. Tapp. Quantum counting. *Proceedings of ICALP'98*, pp. 820-831. Also quant-ph/9805082.

[14] G. Brassard. P. Høyer, M. Mosca, A. Tapp. Quantum amplitude amplification and estimation. *Quantum Computation and Quantum Information Science*, AMS Contemporary Math Series, vol. 305, pp. 53–74. Also quant-ph/0005055.

[15] H. Buhrman, C. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, R. de Wolf. Quantum algorithms for element distinctness. *Proceedings of Complexity'01*, pp. 131-137. Also quant-ph/0007016.

[4] quant-ph preprints are available from `http://www.arxiv.org/abs/quant-ph/xxxxxx`, where xxxxxx is the 7-digit preprint number

[16] H. Buhrman, R. Spalek. manuscript, 2003.

[17] H. Buhrman, R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288:21-43, 2002.

[18] A.M. Childs and J.M. Eisenberg. Quantum algorithms for subset finding. quant-ph/0311038.

[19] C. Dürr, P. Høyer. A quantum algorithm for finding the minimum. quant-ph/9607014.

[20] C. Dürr, M. Heiligman, P. Høyer, M. Mhalla. Quantum query complexity of some graph problems. quant-ph/0401091.

[21] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren and D. Preda, A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem, *Science* 292, 472 (2001), quant-ph/0104129.

[22] L. Grover. A fast quantum mechanical algorithm for database search. In *Proceeding of STOC'96*, pp. 212–219. Also quant-ph/9605043.

[23] L. Grover. A framework for fast quantum mechanical algorithms. *Proceedings of STOC'98*, pp. 53-62. Also quant-ph/9711043.

[24] P. Høyer, J. Neerbek, Y. Shi. Quantum complexities of ordered searching, sorting, and element distinctness. *Algorithmica*, 34(4): 429-448, 2002. Also quant-ph/0102078.

[25] T. Hofmeister, U. Schöning, R. Schuler, O. Watanabe. A probabilistic 3-SAT algorithm further improved. *Proceedings of STACS'02*, pp. 192-202.

[26] J. Kempe. Quantum random walks - an introductory overview, *Contemporary Physics*, 44 (4):307-327, 2003. Also quant-ph/0303081.

[27] F. Magniez, M. Santha, and M. Szegedy. Quantum algorithms for the triangle problem. quant-ph/0310134.

[28] G. Midrijānis. Exact quantum query complexity for total Boolean functions. quant-ph/0403168.

[29] A. Nayak, F. Wu. The quantum query complexity of approximating the median and related statistics. *Proceedings of STOC'99*, pp. 384-393. Also quant-ph/9804066.

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

[31] D. Rolf. $3-SAT \in RTIME(1.32971^n)$. Diploma thesis, Humboldt-Universität zu Berlin, 2003.

[32] T. Rudolph, L. Grover. How significant are the known collision and element distinctness quantum algorithms? quant-ph/0309123.

[33] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. *Proceedings of FOCS'99*, pp. 410-414.

[34] Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. *Proceedings of FOCS'02*, pp. 513-519. Also quant-ph/0112086.

[35] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5): 1484-1509, 1997. Also FOCS'94.

[36] M. Szegedy. Spectra of quantized walks and a $\sqrt{\delta\epsilon}$ rule, quant-ph/0401053.

[37] C. Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60:27462751, 1999. Also quant-ph/9711070.