SIAM J. COMPUT.
Vol. 37, No. 2, pp. 413–424


# QUANTUM ALGORITHMS FOR THE TRIANGLE PROBLEM*

FRÉDÉRIC MAGNIEZ†, MIKLOS SANTHA†, AND MARIO SZEGEDY‡

**Abstract.** We present two new quantum algorithms that either find a triangle (a copy of $K_3$) in an undirected graph $G$ on $n$ nodes, or reject if $G$ is triangle free. The first algorithm uses combinatorial ideas with Grover Search and makes $\tilde{O}(n^{10/7})$ queries. The second algorithm uses $\tilde{O}(n^{13/10})$ queries and is based on a design concept of Ambainis [in *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, 2004, pp. 22–31] that incorporates the benefits of quantum walks into Grover Search [L. Grover, in *Proceedings of the Twenty-Eighth ACM Symposium on Theory of Computing*, 1996, pp. 212–219]. The first algorithm uses only $O(\log n)$ qubits in its quantum subroutines, whereas the second one uses $O(n)$ qubits. The Triangle Problem was first treated in [H. Buhrman et al., *SIAM J. Comput.*, 34 (2005), pp. 1324–1330], where an algorithm with $O(n + \sqrt{nm})$ query complexity was presented, where $m$ is the number of edges of $G$.

**Key words.** 05C85, 68R10, 68W99

**AMS subject classifications.** quantum algorithm, query complexity, triangle problem, quantum walk

**DOI.** 10.1137/050643684

## 1. Introduction.

Quantum computing is an extremely active research area (for introductions, see, e.g., [22, 20]), where a growing trend is the study of quantum query complexity. The quantum query model was implicitly introduced by Deutsch [15], Deutsch and Jozsa [16], Simon [25], and Grover [18], and explicitly by Beals et al. [9]. In this model, as in its classical counterpart, we pay for accessing the oracle (the black box), but unlike in the classical case, the machine can use the power of quantum parallelism to make queries in superpositions. While no significant lower bounds are known in quantum time complexity, the black box constraint sometimes enables us to prove such bounds in the query model.

For promise problems quantum query complexity indeed can be exponentially smaller than the randomized query complexity; a prominent example for that is the Hidden Subgroup Problem [25, 17]. On the other hand, Beals et al. [9] showed that for total functions the deterministic and the quantum query complexities are polynomially related. In this context, a large axis of research pioneered by Grover [18] was developed around search problems in unstructured, structured, or partially structured databases.

The classical query complexity of graph properties has made its fame through the notoriously hard evasiveness conjecture of Aanderaa and Rosenberg [24] which states that every nontrivial and monotone boolean function on graphs whose value remains invariant under the permutation of the nodes has deterministic query complexity exactly $\binom{n}{2}$, where $n$ is the number of nodes of the input graph. Though this conjecture

*Received by the editors October 27, 2005; accepted for publication (in revised form) December 4, 2006; published electronically May 16, 2007. A preliminary version of this paper appeared in *Proceedings of the Sixteenth ACM-SIAM Symposium on Discrete Algorithms*, 2005, pp. 1109–1117.
http://www.siam.org/journals/sicomp/37-2/64368.html

†CNRS–LRI, UMR 8623 Université Paris–Sud, 91405 Orsay, France (magniez@lri.fr, santha@lri.fr). The research of this author was partially supported by the EU 5th and 6th framework programs RESQ IST-2001-37559, RAND-APX IST-1999-14036 and QAP IST, and by ACI Cryptologie CR/02 02 0040 and ACI Sécurité Informatique 03 511 grants of the French Research Ministry.

‡Department of Computer Science, Rutgers University, Piscataway, NJ 08854 (szegedy@cs.rutgers.edu). This author's research was supported by NSF grant 0105692 and the EU 5th framework program RESQ IST-2001-37559. The research was done while the author was visiting LRI.

is still open, an $\Omega(n^2)$ lower bound has been established by Rivest and Vuillemin [23]. In randomized bounded error complexity the general lower bounds are far from the conjectured $\Omega(n^2)$. The first nonlinear lower bound was shown by Yao [30]. For a long time Hajnal's $\Omega(n^{4/3})$ bound [19] was the best, until it was slightly improved in [13] to $\Omega(n^{4/3}\log^{1/3} n)$. The question of the quantum query complexity of graph properties was first raised in [11], where it is shown that in the exact case an $\Omega(n^2)$ lower bound still holds. In the bounded error quantum query model, the $\Omega(n^2)$ lower bound does not hold anymore in general. An $\Omega(n^{2/3}\log^{1/6} n)$ lower bound, first observed by Yao [31], can be obtained combining Ambainis' technique [4] with the above randomized lower bound.

We address the Triangle Problem in this setting. In a graph $G$, a complete subgraph on three vertices is called a *triangle*. In this write-up we study the following oracle problem:

TRIANGLE
*Oracle Input:* The adjacency matrix $f$ of a graph $G$ on $n$ nodes.
*Output:* A triangle if there is any, otherwise reject.

TRIANGLE has been studied in various contexts, partly because of its relation to matrix multiplication [3]. Its quantum query complexity was first raised in [12], where the authors show that in the case of sparse graphs the trivial (that is, using Grover Search) $O(n^{3/2})$ upper bound can be improved. Their method breaks down when the graph has $\Theta(n^2)$ edges.

The quantum query complexity of TRIANGLE as well as of many of its kins with small one-sided certificate size is notoriously hard to analyze, because one of the main lower bounding methods breaks down near the square root of the instance size [27, 21, 32, 26]: *If the* 1*-certificate size of a boolean function on* $N$ *boolean variables is* $K$*, then even the most general variants* [8, Theorem 4], [5], [21] *of Ambainis' quantum adversary technique* [4] *can prove only a lower bound of* $\Omega(\sqrt{NK})$. Indeed only the $\Omega(n)$ lower bound is known for TRIANGLE, which, because of the remark above, cannot be improved using any quantum adversary technique ($N = n^2$ and $K = 3$). Problems with small certificate complexity include various collision type problems such as the 2-1 Collision Problem and the Element Distinctness Problem. The first polynomial lower bound for the 2-1 Collision Problem was shown by Aaronson and Shi [1] using the polynomial method of Beals et al. [9]. For the Element Distinctness Problem, a randomized reduction from the 2-1 Collision Problem gives $\Omega(n^{2/3})$.

In this paper we present two different approaches that give rise to new upper bounds. First, using combinatorial ideas, we design an algorithm for TRIANGLE (Theorem 3.5) whose quantum query complexity is $\tilde{O}(n^{10/7})$. Surprisingly, its quantum parts consist in only Grover Search subroutines. Indeed, Grover Search coupled with the Szemerédi lemma [28] already gives an $o(n^{3/2})$ bound. We exploit this fact using a simpler observation that leads to the $\tilde{O}(n^{10/7})$ bound. Moreover, our algorithm uses only small quantum memory, namely $O(\log n)$ qubits (and $O(n^2)$ classical bits). Then, we generalize the new elegant method used by Ambainis [6] for solving the Element Distinctness Problem in $O(n^{2/3})$, to solve a general Collision Problem by a dynamic quantum query algorithm (Theorem 4.1). The solution of the general Collision Problem will be used in our second algorithm for TRIANGLE. As an intermediate step, we introduce the Graph Collision Problem, which is a variant of the Collision Problem, and solve it in $\tilde{O}(n^{2/3})$ query complexity (Theorem 4.4). Whereas a reduction of TRIANGLE to the Element Distinctness Problem does not give a better algorithm than $O(n^{3/2})$, using a recursion of our dynamic version of Ambainis' method we prove the $\tilde{O}(n^{13/10})$ query complexity for TRIANGLE (Theorem 4.5). We end by

generalizing this result for finding the copy of any given graph (Theorem 4.6) and then for every graph property with small 1-certificates (Corollary 4.7).

## 2. Preliminaries.

**2.1. Query model.** In the query model of computation each query adds one to the complexity of an algorithm, but all other computations are free. The state of the computation is represented by three registers, the query register $x$, the answer register $a$, and the work register $z$. The computation takes place in the vector space spanned by all basis states $|x, a, z\rangle$. In the *quantum query model* the state of the computation is a complex combination of all basis states which has unit length in the norm $l_2$.

The query operation $O_f$ maps the basis state $|x, a, z\rangle$ into the state $|x, a \oplus f(x), z\rangle$ (where $\oplus$ is bitwise XOR). Nonquery operations are independent of $f$. A *$k$-query algorithm* is a sequence of $(k+1)$ operations $(U_0, U_1, \ldots, U_k)$, where $U_i$ is unitary. Initially the state of the computation is set to some fixed value $|0, 0, 0\rangle$, and then the sequence of operations $U_0, O_f, U_1, O_f, \ldots, U_{k-1}, O_f, U_k$ is applied.

**2.2. Notation.** We denote the set $\{1, 2, \ldots, n\}$ by $[n]$. A simple undirected graph is a set of edges $G \subseteq \{(a, b) \mid\ a, b \in [n];\ a \neq b\}$ with the understanding that $(a, b) \stackrel{\text{def}}{=} (b, a)$. Let $t(G)$ denote the number of triangles in $G$. The complete graph on a set $\nu \subseteq [n]$ is denoted by $\nu^2$. The neighborhood of a $v \in [n]$ in $G$ is denoted by $\nu_G(v)$, and it is defined by $\nu_G(v) = \{b \mid\ (v, b) \in G\}$. We denote $|\nu_G(v)|$ by $\deg_G v$. For sets $A, B \subseteq [n]$ let $G(A, B) = \{(a, b) \mid\ a \in A;\ b \in B;\ (a, b) \in G\}$.

The following function will play a major role in our proof. We denote the number of paths of length 2 from $a \in [n]$ to $b \in [n]$ in $G$ with $t(G, a, b)$: $t(G, a, b) = |\{x \mid (a, x) \in G;\ (b, x) \in G\}|$. For a graph $G \subseteq [n]^2$ and an integer $k \geq 0$, we define $G^{\langle k \rangle} = \{(a, b) \in [n]^2 \mid\ t(G, a, b) \leq k\}$.

**2.3. Quantum subroutines.** We will use a safe version of Grover Search [18], namely Safe Grover Search($t$), based on $t$ iterations of Grover Search and followed by a checking process for markedness of output instances.

FACT 2.1. *Let $c > 0$. Safe Grover Search($\Theta(c \log N)$) on a database of $N$ items has quantum query complexity $O(c\sqrt{N} \log N)$, and it always rejects if there is no marked item; otherwise it finds a marked item with probability at least $1 - \frac{1}{N^c}$.*

For quantum walks on graphs we usually define two operators: *coin flip* and *shift*. The state of the walk is held in a pair of registers, the *node* and the *coin*. The coin flip operator acts only on the coin register and is the identity on the node register. The shift operation changes only the node register, but it is controlled by the content of the coin register (see [29, 2, 7]). Often the coin flip is actually the diffusion operator.

DEFINITION 2.2 (diffusion over $T$). *Let $T$ be a finite set. The* diffusion *operator over $T$ is the unitary operator on the Hilbert space $\mathbf{C}^T$ that acts on a basis element $|x\rangle$, $x \in T$ as: $|x\rangle \mapsto -|x\rangle + \frac{2}{|T|} \sum_{y \in T} |y\rangle$.*

In [6] a new walk is described that plays a central role in our result. Let $S$ be a finite set of size $n$. The node register holds a subset $A$ of $S$ of size either $r$ or $r+1$ for some fixed $0 < r < n$, and the coin register holds an element $x \in S$. Thus the basis states are of the form $|A\rangle|x\rangle$, where we also require that if $|A| = r$, then $x \notin A$, and if $|A| = r + 1$, then $x \in A$. We also call the node register the *set register*.

**Quantum Walk.**

1. Diffuse the coin register over $S - A$.
2. Add $x$ to $A$.
3. Diffuse the coin register over $A$.
4. Remove $x$ from $A$.

Ambainis [6] showed that, inside some specific stable subspaces, $\Theta(\sqrt{r})$ iterations of Quantum Walk can play the role of the diffusion over $\{(A,x) : A \subset S, |A| = r, x \notin S\}$. This nice result leads to a more efficient Grover Search for some problems such as the Element Distinctness Problem [6]. We will describe this in a general setting in section 4.1.

## 3. Combinatorial approach.

**3.1. Preparation.** The algorithm presented here is based on three combinatorial observations. Throughout this section we do not try to optimize $\log n$ factors and we will hide time in the $\tilde{O}$ notation. The first observation is based on the amplitude amplification technique of Brassard et al. [10].

LEMMA 3.1. *For any known graph $E \subseteq [n]^2$, a triangle with at least one edge in $E$ can be detected with $\tilde{O}(\sqrt{E} + \sqrt{n|G \cap E|})$ queries and probability $1 - \frac{1}{n}$.*

Perhaps the most crucial observation to the algorithm is the following simple one.

LEMMA 3.2. *For every $v \in [n]$, using $\tilde{O}(n)$ queries, we either find a triangle in $G$ or verify that $G \subseteq [n]^2 \setminus \nu_G(v)^2$ with probability $1 - \frac{1}{n^3}$.*

*Proof.* We query all edges incident to $v$ classically using $n-1$ queries. This determines $\nu_G(v)$. With Safe Grover Search we find an edge of $G$ in $\nu_G(v)^2$ if there is any. □

This lemma, with the observation that hard instances have to be dense, already enables us to show that the quantum query complexity of TRIANGLE is $o(n^{3/2})$, using the Szemerédi lemma [28]. However, another fairly simple observation can help us to decrease the exponent.

LEMMA 3.3. *Let $0 < \varepsilon < 1$, $k = \lceil 4n^{\varepsilon} \log n \rceil$, and let $v_1, v_2, \ldots, v_k$ be randomly chosen from $[n]$ (with no repetitions). Let $G' = [n]^2 \setminus \cup_{i=1}^{k} \nu_G(v)^2$. Then $\Pr_{v_1,v_2,\ldots,v_k}(G' \subseteq G^{\langle n^{1-\varepsilon} \rangle}) > 1 - \frac{1}{n}$.*

Let us first remind the reader about the following lemma that is useful in many applications.

LEMMA 3.4. *Let $X$ be a fixed subset of $[n]$ of size $pn$ and $Y$ be a random subset of $[n]$ of size $qn$, where $p + q < 1$. Then the probability that $X \cap Y$ is empty is $(1-pq)^{n(1 \pm O(p^3+q^3+1/n))}$.*

*Proof.* The probability we are looking for is estimated using the Stirling formula as

$$\frac{\binom{n(1-p)}{nq}}{\binom{n}{nq}} = \frac{[n(1-p)]![nq]![n(1-q)]!}{[nq]![n(1-p-q)]!n!}$$

$$= \sqrt{\frac{(1-p)(1-q)}{1-p-q}} \left[\frac{(1-p)^{1-p}(1-q)^{1-q}}{(1-p-q)^{1-p-q}}\right]^n (1 \pm o(1))$$

$$= (1-pq)^{n(1 \pm O(p^3+q^3+1/n))}. \qquad \square$$

*Proof of Lemma* 3.3. Consider now a fixed edge $(a,b)$ such that $t(G,a,b) \geq n^{1-\varepsilon}$. The probability that $(a,b) \in G'$ is the same as the probability that the set $X = \{x \in [n] : (x,a) \in G \text{ and } (x,b) \in G\}$ is disjoint from the random set $\{v_1, v_2, \ldots, v_k\}$. Notice that $|X| = t(G,a,b)$. By Lemma 3.4 we can now estimate this probability as, for sufficiently large $n$,

$$\left(1 - \frac{4n^{\varepsilon} \log n}{n} \times \frac{n^{1-\varepsilon}}{n}\right)^{n(1+o(1))} = \left(1 - \frac{4 \log n}{n}\right)^{n(1+o(1))} < e^{-3 \log n} = n^{-3}.$$

Then the lemma follows from the union bound, since the number of possible edges $(a, b)$ is at most $n^2$. □

**3.2. Algorithm and analysis.** We now describe our algorithm where all searches are done using Safe Grover Search. We delay details of Step 6 for a while.

**Combinatorial Algorithm**$(\varepsilon, \delta, \varepsilon')$.

1. Let $k = \lceil 4n^{\varepsilon} \log n \rceil$.
2. Randomly choose $v_1, \ldots, v_k$ from $[n]$ (with no repetition).
3. Compute every $\nu_G(v_i)$.
4. If $G \cap \nu_G(v_i)^2 \neq \emptyset$, for some $i$, then output the triangle induced by $v_i$.
5. Let $G' = [n]^2 \setminus \cup_i (\nu_G(v_i)^2)$.
6. Classify the edges of $G'$ into $T$ and $E$ such that
   – $T$ contains only $O(n^{3-\varepsilon'})$ triangles,
   – $E \cap G$ has size $O(n^{2-\delta} + n^{2-\varepsilon+\delta+\varepsilon'})$.
7. Search for a triangle in $G$ among all triangles inside $T$.
8. Search for a triangle of $G$ intersecting with $E$.
9. Output a triangle if it is found; otherwise reject.

THEOREM 3.5. *Combinatorial Algorithm*$(\varepsilon, \delta, \varepsilon')$ *rejects with probability one if there is no triangle in* $G$*; otherwise, it returns a triangle of* $G$ *with probability* $1 - O(\frac{1}{n})$. *Moreover, it has query complexity* $\tilde{O}(n^{1+\varepsilon} + n^{1+\delta+\varepsilon'} + \sqrt{n^{3-\varepsilon'}} + \sqrt{n^{3-\min(\delta, \varepsilon-\delta-\varepsilon')}})$.

With $\varepsilon = \frac{3}{7}$, $\varepsilon' = \delta = \frac{1}{7}$ this gives $\tilde{O}(n^{1+\frac{3}{7}})$ for the total number of queries.

We require every probabilistic step to be correctly performed with probability $1 - O(\frac{1}{n^3})$, so that the overall probability of a correct execution is $1 - O(\frac{1}{n})$, using the union bound and since the number of such steps is at most $O(n^2)$. Thus we will always assume that an execution is correct. Since an incorrect execution might increase the query complexity of the algorithm, we also assume there is a counter so that the algorithm rejects and stops when a threshold is exceeded. This threshold is defined as the maximum of query complexities over all correct executions.

The main step of Combinatorial Algorithm is step 6, which we implement in the following way.

**Classification**$(G', \delta, \varepsilon')$**.**

1. Set $T = \emptyset$, $E = \emptyset$.
2. While $G' \neq \emptyset$ do
   (a) While there is an edge $(v, w) \in G'$ s.t. $t(G', v, w) < n^{1-\varepsilon'}$, add $(v, w)$ to $T$, and delete it from $G'$.
   (b) Pick a vertex $v$ of $G'$ with nonzero degree and decide
      1. *low degree hypothesis*: $|\nu_G(v)| \leq 10 \times n^{1-\delta}$;
      2. *high degree hypothesis*: $|\nu_G(v)| \geq \frac{1}{10} \times n^{1-\delta}$.
   (c) If Hypothesis 1, add all edges $(v, w)$ of $G'$ to $E$ and delete them from $G'$.
   (d) If Hypothesis 2, then
      i. Compute $\nu_G(v)$;
      ii. If $G \cap \nu_G(v)^2 \neq \emptyset$, output the triangle induced by $v$ and stop;
      iii. Add all edges in $G'(\nu_G(v), \nu_{G'}(v))$ to $E$ and delete them from $G'$.

In step 2(b), we use an obvious sampling strategy:

> Set a counter $C$ to 0. Query $\lceil n^\delta \rceil$ random edge candidates from $v \times [n]$. If there is an edge of $G$ among them, add one to $C$. Repeat this process $K = c_0 \log n$ times, where $c_0$ is a sufficiently large; constant. Accept the low degree hypothesis if by the end $C < K/2$; otherwise accept the large degree hypothesis.

Observe that one could use here a quantum procedure based on Grover Search. Since the cost of this step is negligible from that of others, this would not give any better bound.

FACT 3.6. *When $c_0$ is large enough in Step* 2(b),

1. *the probability that* $\deg_G(v) > 10 \times n^{1-\delta}$ *and the low degree hypothesis is accepted is* $O(\frac{1}{n^3})$;
2. *the probability that* $\deg_G(v) < \frac{1}{10} \times n^{1-\delta}$ *and the high degree hypothesis is accepted is* $O(\frac{1}{n^3})$.

*Proof.* Indeed, using Lemma 3.4, considering a single round of sampling, the probability that our sample set does not contain an edge from $G$ even though $\deg_G(v) > 10 \times n^{1-\delta}$ is, for sufficiently large $n$,

$$\left(1 - \frac{10n^{1-\delta}}{n} \times \frac{n^\delta}{n}\right)^{n(1+o(1))} = \left(1 - \frac{10}{n}\right)^{n(1+o(1))} < 0.1.$$

Similarly, the probability that our sample set contains an edge from $G$ even though $\deg_G(v) < \frac{1}{10} \times n^{1-\delta}$ is

$$1 - \left(1 - \frac{n^{1-\delta}}{10n} \times \frac{n^\delta}{n}\right)^{n(1+o(1))} = 1 - \left(1 - \frac{1}{10n}\right)^{n(1+o(1))} < 0.2.$$

Now for $K = c_0 \log n$ rounds, where $c_0$ is large enough, the Chernoff bound gives the claim. □

LEMMA 3.7. *If $G \subseteq G' \subseteq G^{\langle n^{1-\varepsilon} \rangle}$, then Classification$(G', \varepsilon', \delta)$ outputs the desired partition $(T, E)$ of $G$ with probability $1 - O(\frac{1}{n})$ and has query complexity $\tilde{O}(n^{1+\delta+\varepsilon'})$.*

*Proof of Theorem* 3.5. Clearly, if there is no triangle in the graph, the algorithm rejects since the algorithm outputs a triplet only after checking that it is a triangle in $G$. Therefore the correctness proof requires us only to calculate the probability with which the algorithm outputs a triangle, if there is any, and the query complexity of the algorithm.

Assume that the execution is without any error. Using the union bound, we can indeed upper bound the probability of incorrect execution by $O(\frac{1}{n})$.

By Lemma 3.2, we already know that the construction of $G'$ requires $\tilde{O}(n^\varepsilon \times n)$ queries. Moreover, either $G \subseteq G'$ or a triangle is found, with probability $1 - O(\frac{1}{n})$. From Lemma 3.3, we also know that $G' \subseteq G^{\langle n^{1-\varepsilon} \rangle}$, with probability $1 - O(\frac{1}{n})$.

Assume that $G'$ lends all its edges to $T$ and $E$; that is, no triangle is found at the end of Classification. Since $G \subseteq G'$, every triangle in $G$ either has to be contained totally in $T$ or has to have a nonempty intersection with $E$. Using Lemma 3.7, we know that the partition $(T, E)$ is correct with probability $1 - O(\frac{1}{n})$. Assume this is the case. $T$ is a graph that is known to us, and so we can find out if one of these triangles belongs to $G$ with $\tilde{O}(\sqrt{n^{3-\varepsilon'}})$ queries, using Safe Grover Search. By Lemma 3.1, the complexity of finding a triangle in $G$ that contains an edge from $E$ is $\tilde{O}(n + \sqrt{n^{3-\min(\delta, \varepsilon-\delta-\varepsilon')}})$.

From the analysis we conclude that the total number of queries is upper bounded by

$$\tilde{O}\left(n^{1+\varepsilon} + n^{1+\delta+\varepsilon'} + \sqrt{n^{3-\varepsilon'}} + \sqrt{n^{3-\min(\delta,\varepsilon-\delta-\varepsilon')}}\right). \qquad \square$$

In the rest of this section we prove Lemma 3.7 using a sequence of facts. Then the proof derives directly. For the query complexity, we detail the analysis using Fact 3.8. Only steps 2(b) and 2(d) of Classification have nonzero query complexity. As explained before, step 2(b) can be implemented with query complexity $\tilde{O}(n^{\delta})$, and it is iterated at most $n$ times. Step 2(d) has three substeps, with only the first two having nonzero query complexity. The first has query complexity $O(n)$, and the second can be implemented using Safe Grover Search with query complexity $\tilde{O}(\sqrt{n^2}) = \tilde{O}(n)$. Using Fact 3.8, we upper bound the number of iterations of step 2(d) by $O(n^{\delta+\varepsilon'})$, which gives a total amount of queries in $\tilde{O}(n^{1+\delta+\varepsilon'})$.

FACT 3.8. *During a correct execution, there are at most $O(n^{\delta+\varepsilon'})$ iterations of step* 2(d).

*Proof.* We will estimate the number of executions of step 2(d) by lower bounding $|G'(A, A')|$, where $A = \nu_G(v)$ and $A' = \nu_{G'}(v)$. For each $x \in A$ we have $t(G', v, x) \geq n^{1-\varepsilon'}$; otherwise in step 2(a) we would have classified $(v, x)$ into $T$. A triangle $(v, x, y)$ contributing to $t(G', v, x)$ contributes with the edge $(x, y)$ to $G'(A, A')$. Two different triangles $(v, x, y)$ and $(v, x', y')$ can give the same edge in $G'(A, A')$ only if $x = y'$ and $y = x'$. Thus,

$$|G'(A, A')| \geq \frac{1}{2} \sum_{x \in \nu_G(v)} t(G', v, x) \geq \frac{|A| n^{1-\varepsilon'}}{2}. \tag{3.1}$$

Since we executed step 2(d) only under the large degree hypothesis on $v$, if the hypothesis is correct, the right-hand side of (3.1) is at least $\frac{1}{10} \times n^{1-\delta} \times n^{1-\varepsilon'}/2 = \Omega(n^{2-\delta-\varepsilon'})$. Since $G'$ has at most $\binom{n}{2}$ edges, it can execute step 2(d) at most $O(n^{\delta+\varepsilon'})$ times. $\square$

FACT 3.9. *During a correct execution, there are at most $O(n)$ iterations of step* 2(c).

*Proof.* We claim that each vertex is processed in step 2(c) at most once. Indeed, if a vertex $v$ gets into step 2(c), its incident edges are all removed, and its degree in $G'$ becomes 0, making it ineligible for being processed in step 2(c) again. $\square$

Now we state that $T$ contains $O(n^{3-\varepsilon'})$ triangles using the following quite general fact.

FACT 3.10. *Let $H$ be a graph on $[n]$. Assume that a graph $T$ is built by a process that starts with an empty set and at every step either discards some edges from $H$ or adds an edge $(a, b)$ of $H$ to $T$ for which $t(H, a, b) \leq \tau$ holds. For the $T$ created by the end of the process we have $t(T) \leq \binom{n}{2}\tau$.*

*Proof.* Let us denote by $T[i]$ the edge of $T$ that $T$ acquired when it was incremented for the $i$th time, and let us use the notation $H^i$ for the current version of $H$ before the very moment when $T[i] = (a_i, b_i)$ was copied into $T$. Since $\{T[i], T[i+1], \dots\} \stackrel{\text{def}}{=} T^i \subseteq H^i$, we have $t(T^i, a_i, b_i) \leq t(H^i, a_i, b_i) \leq \tau$. Now the fact follows from $t(T) = \sum_i t(T^i, a_i, b_i) \leq \binom{n}{2}\tau$, since $i$ can go up to at most $\binom{n}{2}$. $\square$

FACT 3.11. *During a correct execution, $E \cap G$ has size $O(n^{2-\delta} + n^{2-\varepsilon+\delta+\varepsilon'})$.*

*Proof.* In order to estimate $E \cap G$ observe that we added edges to $E$ only in steps 2(c) and 2(d). In each execution of step 2(c), we added at most $10n^{1-\delta}$ edges

to $E$, and we had $O(n)$ such executions (Fact 3.9) that give a total of $O(n^{2-\delta})$ edges. The number of executions of step 2(d) is $O(n^{\delta+\varepsilon'})$ (Fact 3.8). Our task is now to bound the number of edges of $G$ that each such execution adds to $E$.

We estimate $|G \cap G'(A, A')|$ from the $A'$ side, where $A = \nu_G(v)$ and $A' = \nu_{G'}(v)$. This is the only place where we use the fact that $G' \subseteq G^{\langle n^{1-\varepsilon}\rangle}$: For every $x \in A'$ we have $t(G, v, x) \leq n^{1-\varepsilon}$. On the other hand, when $y \in A$ and $x \in A'$, every edge $(y, x) \in G'$ creates a $(v, x)$-based triangle. Thus

$$|G \cap G'(A, A')| \leq |A'|n^{1-\varepsilon} \leq n^{2-\varepsilon}.$$

Therefore the total number of edges of $G$ step 2(d) contributes to $E$ is $n^{2-\varepsilon+\delta+\varepsilon'}$. In conclusion,

$$|G \cap E| \leq O(n^{2-\delta} + n^{2-\varepsilon+\delta+\varepsilon'}). \qquad \square$$

## 4. Quantum Walk approach.

**4.1. Dynamic quantum query algorithms.** The algorithm of Ambainis in [6] is somewhat similar to the brand of classical algorithms, where a database is used (as in heapsort) to quickly retrieve the value of those items needed for the run of the algorithm. Of course, this whole paradigm is placed into the context of query algorithms. We shall define a class of problems that can be tackled very well with the new type of algorithm. Let $S$ be a finite set of size $n$ and let $0 < k < n$.

$k$-Collision
*Oracle Input:* A function $f$ which defines a relation $\mathcal{C} \subseteq S^k$.
*Output:* A $k$-tuple $(a_1, \ldots, a_k) \in \mathcal{C}$ if it is nonempty; otherwise reject.

By carefully choosing the relation $\mathcal{C}$, $k$-Collision can be a useful building block in the design of different algorithms. For example, if $f$ is the adjacency matrix of a graph $G$, and the relation $\mathcal{C}$ is defined as "being an edge of a triangle of $G$," then the output of Collision yields a solution for Triangle with $O(\sqrt{n})$ additional queries (Grover Search for the third vertex).

Unique $k$-Collision: The same as $k$-Collision with the promise that $|\mathcal{C}| = 1$ or $|\mathcal{C}| = 0$.

The type of algorithm we study will use a database $D$ associating some data $D(A)$ to every set $A \subseteq S$. From $D(A)$ we would like to determine if $A^k \cap \mathcal{C} \neq \emptyset$. We expedite this using a quantum query procedure $\Phi$ with the property that $\Phi(D(A))$ rejects if $A^k \cap \mathcal{C} = \emptyset$ and, otherwise, both accepts and outputs an element of $A^k \cap \mathcal{C}$, which is a *collision*. When operating with $D$, the following three types of costs are incurred, all measured in the number of queries to the oracle $f$.

*Setup cost $s(r)$:* The cost to set up $D(A)$ for a set of size $r$.
*Update cost $u(r)$:* The cost to update $D$ for a set of size $r$, i.e., moving from $D(A)$ to $D(A')$, where $A'$ results from $A$ by adding an element, or moving from $D(A'')$ to $D(A)$, where $A$ results from $A''$ by deleting an element.
*Checking cost $c(r)$:* The query complexity of $\Phi(D(A))$ for a set of size $r$.

Next we describe the algorithm of Ambainis [6] in general terms. The algorithm has three registers $|A\rangle|D(A)\rangle|x\rangle$. The first is called the *set register*, the second the *data register*, and the last the *coin register*.

**Generic Algorithm**$(r, D, \Phi)$**.**

1. Create the state $\sum_{A\subset S:|A|=r} |A\rangle$ in the set register.
2. Set up $D$ on $A$ in the data register.
3. Create a uniform superposition over elements of $S - A$ in the coin register.
4. Do $\Theta(n/r)^{k/2}$ times:
   (a) If $\Phi(D(A))$ accepts, then do the phase flip; otherwise do nothing
   (b) Do $\Theta(\sqrt{r})$ times **Quantum Walk**, updating the data register.
5. If $\Phi(D(A))$ rejects, then reject; otherwise output the collision given by $\Phi(D(A))$.

THEOREM 4.1 (see [6]). Generic Algorithm *solves* UNIQUE $k$-COLLISION *with some positive constant probability and has query complexity*

$$O(s(r) + (\tfrac{n}{r})^{k/2} \times (c(r) + \sqrt{r} \times u(r))).$$

Moreover, it turns out that, when UNIQUE $k$-COLLISION has no solution, Generic Algorithm always rejects, and when UNIQUE $k$-COLLISION has a solution $c$, Generic Algorithm outputs $c$ with probability $p = \Omega(1)$ which depends only on $k$, $n$, and $r$. Thus using quantum amplification, one can modify Generic Algorithm to an exact quantum algorithm.

COROLLARY 4.2. UNIQUE $k$-COLLISION *can be solved with probability* 1 *in quantum query complexity*

$$O(s(r) + (\tfrac{n}{r})^{k/2} \times (c(r) + \sqrt{r} \times u(r))).$$

One can make a random reduction from COLLISION to UNIQUE COLLISION if the definition on $\Phi$ is slightly generalized. We add to the input of the checking procedure a relation $\mathcal{R} \subseteq S^k$ which restricts the collision set $\mathcal{C}$ to $\mathcal{C} \cap \mathcal{R}$. The reduction goes in the standard way using a logarithmic number of randomly chosen relations $\mathcal{R}$, and hence an additional logarithmic factor appears in the complexity. If the collision relation is robust in some sense, one can improve this reduction by removing the log factors (see, for example, the reduction used by Ambainis in [6]).

COROLLARY 4.3. COLLISION *can be solved in quantum query complexity*

$$\tilde{O}(s(r) + (\tfrac{n}{r})^{k/2} \times (c(r) + \sqrt{r} \times u(r))).$$

The tables below summarize the use of the above formula for various problems (GRAPH COLLISION$(G)$ is defined in section 4.2).

| Problem | Collision relation |
|---|---|
| ELEMENT DISTINCTNESS | $(u, v) \in \mathcal{C}$ iff $u \neq v$ and $f(u) = f(v)$ |
| GRAPH COLLISION$(G)$ | $(u, v) \in \mathcal{C}$ iff $f(u) = f(v) = 1$ and $(u, v) \in G$ |
| TRIANGLE | $(u, v) \in \mathcal{C}$ iff there is a triangle $(u, v, w)$ in $G$ |

| Problem | Setup cost $s(r)$ | Update cost $u(r)$ | Checking cost $c(r)$ |
|---|---|---|---|
| ELEMENT DISTINCTNESS | $r$ | 1 | 0 |
| GRAPH COLLISION$(G)$ | $r$ | 1 | 0 |
| TRIANGLE | $O(r^2)$ | $r$ | $O(r^{2/3}\sqrt{n})$ |

**4.2. Graph Collision Problem.** Here we deal with an interesting variant of COLLISION which will be also useful for finding a triangle. The problem is parametrized by some graph $G$ on $n$ vertices which is given explicitly.

GRAPH COLLISION($G$)
*Oracle Input:* A boolean function $f$ on $[n]$ which defines the relation $\mathcal{C} \subseteq [n]^2$ such that $\mathcal{C}(u,u')$ if and only if $f(u) = f(u') = 1$ and $(u,u') \in E$.
*Output:* A pair $(u,u') \in \mathcal{C}$ if it is nonempty; otherwise reject.

Observe that an equivalent formulation of the problem is to decide if the set of vertices of value 1 form an independent set in $G$.

THEOREM 4.4. GRAPH COLLISION($G$) *can be solved with positive constant probability in quantum query complexity* $\tilde{O}(n^{2/3})$.

*Proof.* We solve GRAPH COLLISION($G$) using Corollary 4.3, with $S = [n]$ and $r = n^{2/3}$. For every $U \subseteq [n]$ we define $D(U) = \{(v, f(v)) : v \in U\}$ and let $\Phi(D(U)) = 1$ if there are $u, u' \in U$ that satisfy the required property. Observe that $s(r) = r$, $u(r) = 1$, and $c(r) = 0$. Therefore we can solve the problem in quantum query complexity $\tilde{O}(r + \frac{n}{r}(\sqrt{r}))$ which is $\tilde{O}(n^{2/3})$ when $r = n^{2/3}$. ∎

**4.3. Triangle Problem.**

THEOREM 4.5. TRIANGLE *can be solved with positive constant probability in quantum query complexity* $\tilde{O}(n^{13/10})$.

*Proof.* We use Corollary 4.3, where $S = [n]$, $r = n^{2/3}$, and $\mathcal{C}$ is the set of triangle edges. We define $D$ for every $U \subseteq [n]$ by $D(U) = G|_U$, and $\Phi$ by $\Phi(G|_U) = 1$ if a triangle edge is in $G|_U$. Observe that $s(r) = O(r^2)$ and $u(r) = r$. We claim that $c(r) = \tilde{O}(\sqrt{n} \times r^{2/3})$.

To see this, let $U$ be a set of $r$ vertices such that $G|_U$ is explicitly known, and let $v$ be a vertex in $[n]$. We define an input oracle for GRAPH COLLISION($G|_U$) by $f(u) = 1$ if $(u,v) \in E$. The edges of $G|_U$ which together with $v$ form a triangle in $G$ are the solutions of GRAPH COLLISION($G|_U$). Therefore finding a triangle edge, if it is in $G|U$, can be done in quantum query complexity $\tilde{O}(r^{2/3})$ by Theorem 4.4. Now using quantum amplification [10], we can find a vertex $v$, if it exists, which forms a triangle with some edge of $G|_U$, using only $\tilde{O}(\sqrt{n})$ iterations of the previous procedure, and with a polynomially small error (which has no influence in the whole algorithm).

Therefore, we can solve the problem in quantum query complexity $\tilde{O}(r^2 + \frac{n}{r}(\sqrt{n} \times r^{2/3} + \sqrt{r} \times r))$ which is $\tilde{O}(n^{13/10})$ when $r = n^{3/5}$. ∎

**4.4. Monotone graph properties with small certificates.** Let now consider the property of having a copy of a given graph $H$ with $k > 3$ vertices. Using directly Ambainis' algorithm, one gets an algorithm whose query complexity is $\tilde{O}(n^{2-2/(k+1)})$. In fact we can improve this bound to $\tilde{O}(n^{2-2/k})$. Note that only the trivial $\Omega(n)$ lower bound is known. This problem was independently considered by Childs and Eisenberg [14] whenever $H$ is a $k$-clique. Beside the direct Ambainis algorithm, they obtained an $\tilde{O}(n^{2.5-6/(k+2)})$ query algorithm. For $k = 4, 5$, this is faster than the direct Ambainis algorithm, but slower than ours.

THEOREM 4.6. *Finding in a graph a copy of a given graph* $H$*, with* $k > 3$ *vertices, can be done with quantum query complexity* $\tilde{O}(n^{2-2/k})$.

*Proof.* We follow the structure of the proof of Theorem 4.5. We distinguish an arbitrary vertex of $H$. Let $d$ be the degree of this vertex in $H$.

We say that a vertex $v$ and a set $K$ of $(k-1)$ vertices of $G$ are *$H$-compatible* if the subgraph induced by $K \cup \{v\}$ in $G$ contains a copy of $H$, in which $v$ is the

distinguished vertex. We also say that the set $K$ is an *$H$-candidate* when there exists a vertex $v$ such that $v$ and $K$ are $H$-compatible. Our algorithm will essentially find a set that contains an $H$-candidate.

We define an instance of $(k-1)$-Collision, where $S = [n]$ and $\mathcal{C}$ is the set of $H$-candidates. We define $D$ for every $U \subseteq [n]$ by $D(U) = G|_U$, and $\Phi$ by $\Phi(G|_U) = 1$ if $U$ contains an $H$-candidate. Again $s(r) = O(r^2)$ and $u(r) = r$. We now claim that $c(r) = \tilde{O}(\sqrt{n} \times r^{d/(d+1)})$.

The checking procedure uses a generalization of Graph Collision to $d$-ary relations. If some vertex $v$ of $G$ is fixed, then we say that a subset $W \subseteq U$ of size $d$ is in relation if there exists $W \subseteq K \subseteq U$ such that $v$ and $K$ are $H$-compatible in $G$, and $v$ is connected to every vertex of $W$. Following the arguments of the proof of Theorem 4.4 (where the function $f$ takes the value 1 on a vertex $u \in U$ if $(u, v)$ is an edge in $G$), we find a $d$-collision in quantum query complexity $\tilde{O}(r^{d/(d+1)})$ when it exists. The checking procedure searches for a vertex $v$ for which this generalized Graph Collision has a solution using a standard Grover Search.

The overall parametrized query complexity is therefore

$$\tilde{O}\left(r^2 + \left(\frac{n}{r}\right)^{(k-1)/2}\left(\sqrt{n} \times r^{d/(d+1)} + \sqrt{r} \times r\right)\right).$$

By optimizing this expression (that is, by balancing the first and third terms), it turns out that the best upper bound does not depend on $d$. Precisely, the expression is optimal with $r = n^{1-1/k}$, which gives the announced bound. However, one can imagine a different algorithm for the checking procedure where the choice of $d$ might be crucial.

To conclude, note that once a set $U$ of size $r$ that contains an $H$-candidate is found, one can obtain a copy of $H$ in $G$ in the complexity of the checking cost $c(r)$. □

We conclude by extending this result for monotone graph properties which might have several small 1-certificates.

Corollary 4.7. *Let $\varphi$ be a monotone graph property whose 1-certificates have at most $k > 3$ vertices. Then deciding $\varphi$ and producing a certificate whenever $\varphi$ is satisfied can be done with quantum query complexity to the graph in $\tilde{O}(n^{2-2/k})$.*

**Acknowledgment.** We would like to thank Andris Ambainis for useful discussions and for sending us a preliminary version of [6].

## REFERENCES

[1] S. Aaronson and Y. Shi, *Quantum lower bounds for the collision and the element distinctness problems*, J. ACM, 51 (2004), pp. 595–605.

[2] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani, *Quantum walks on graphs*, in Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, 2001, pp. 50–59.

[3] N. Alon, R. Yuster, and U. Zwick, *Finding and counting given length cycles*, Algorithmica, 17 (1997), pp. 209–223.

[4] A. Ambainis, *Quantum lower bounds by quantum arguments*, J. Comput. System Sci., 64 (2002), pp. 750–767.

[5] A. Ambainis, *Polynomial degree vs. quantum query complexity*, J. Comput. System Sci., 72 (2006), pp. 220–238.

[6] A. Ambainis, *Quantum walk algorithm for element distinctness*, in Proceedings of the 45th IEEE Symposium on Foundations of Computer Science, 2004, pp. 22–31.

[7] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous, *One-dimensional quantum walks*, in Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, 2001, pp. 37–49.

[8] H. Barnum, M. Saks, and M. Szegedy, *Quantum decision trees and semidefinite programming*, in Proceedings of the 18th IEEE Annual Conference on Computational Complexity, 2003, pp. 179–193.

[9] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, *Quantum lower bounds by polynomials*, J. ACM, 48 (2001), pp. 778–797.

[10] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, *Quantum amplitude amplification and estimation*, in Quantum Computation and Quantum Information: A Millennium Volume, Contemp. Math. 305, AMS, Providence, RI, 2002, pp. 53–74.

[11] H. Buhrman, R. Cleve, R. de Wolf, and C. Zalka, *Bounds for small-error and zero-error quantum algorithms*, in 40th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1999, pp. 358–368.

[12] H. Buhrman, C. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf, *Quantum algorithms for element distinctness*, SIAM J. Comput., 34 (2005), pp. 1324–1330.

[13] A. Chakrabarti and S. Khot, *Improved lower bounds on the randomized complexity of graph properties*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 2076, Springer-Verlag, Berlin, 2001, pp. 285–296.

[14] A. Childs and J. Eisenberg, *Quantum Algorithms for Subset Finding*, Tech. report, Massachusetts Institute of Technology, Cambridge, MA, 2003; available online from http://www.arxiv.org/abs/quant-ph/0311038.

[15] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proc. Roy. Soc. London Ser. A, 400 (1985), pp. 97–117.

[16] D. Deutsch and R. Jozsa, *Rapid solution of problems by quantum computation*, Proc. Roy. Soc. London Ser. A, 439 (1992), pp. 553–558.

[17] M. Ettinger, P. Høyer, and E. Knill, *The quantum query complexity of the hidden subgroup problem is polynomial*, Inform. Process. Lett., 91 (2004), pp. 43–48.

[18] L. Grover, *A fast quantum mechanical algorithm for database search*, in Proceedings of the Twenty-Eighth ACM Symposium on Theory of Computing, 1996, pp. 212–219.

[19] P. Hajnal, *An $n^{4/3}$ lower bound on the randomized complexity of graph properties*, Combinatorica, 11 (1991), pp. 131–143.

[20] A. Kitaev, A. Shen, and M. Vyalyi, *Classical and Quantum Computation*, Grad. Stud. Math. 47, AMS, Providence, RI, 2002.

[21] S. Laplante and F. Magniez, *Lower bounds for randomized and quantum query complexity using Kolmogorov arguments*, in Proceedings of 19th IEEE Annual Conference on Computational Complexity, 2004, pp. 214–304.

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

[23] R. Rivest and J. Vuillemin, *On recognizing graph properties from adjacency matrices*, Theoret. Comput. Sci., 3 (1976), pp. 371–384.

[24] A. Rosenberg, *On the time required to recognize properties of graphs: A problem*, SIGACT News, 5 (1973), pp. 15–16.

[25] D. R. Simon, *On the power of quantum computation*, SIAM J. Comput., 26 (1997), pp. 1474–1483.

[26] R. Špalek and M. Szegedy, *All quantum adversary methods are equivalent*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 3580, Springer-Verlag, Berlin, 2005, pp. 1299–1311.

[27] M. Szegedy, *On the Quantum Query Complexity of Detecting Triangles in Graphs*, Tech. report, 2003; available online from http://www.arxiv.org/abs/quant-ph/0310107.

[28] E. Szemerédi, *Regular partitions of graphs*, in Problèmes combinatoires et théorie des graphes (1976), Colloq. Internat. CNRS 260, CNRS, Paris, 1978, pp. 399–401.

[29] J. Watrous, *Quantum simulations of classical random walks and undirected graph connectivity*, J. Comput. System Sci., 62 (2001), pp. 376–391.

[30] A. Yao, *Lower bounds to randomized algorithms for graph properties*, J. Comput. System Sci., 42 (1991), pp. 267–287.

[31] A. Yao, *private communication*, 2003.

[32] S. Zhang, *On the power of Ambainis's lower bounds*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 3142, Springer-Verlag, Berlin, 2004, pp. 1238–1250.