Quantum query complexity in computational geometry revisited *

by A. Bahadur, C. Dürr, T. Lafaye and R. Kulkarni

abhinav.bahadur@gmail.com (IIT Bombay)
durr@lri.fr (LRI, Univ. Paris-Sud)
thibault@lafaye.org (Conservatoire de musique, Lyon)
raghav@cs.uchicago.edu (CS dep. Univ. of Chicago)

1 Introduction

We are interested in finding quantum algorithms for problems in the area of computation geometry. Many of the problems we study have already polynomial time algorithms. But since in this area the input sizes are huge, quadratic time algorithms are often not good enough. Bounded error quantum algorithms can actually have sublinear running time.

To our knowledge there have been two works on the subject. One is by K. Sadakane, N. Sugawara, T. Tokuyama [6],[7] and the other by W. Smith [8]. These papers do not contain lower bounds. The main quantum ingredient in their algorithms is a quantum algorithm for the Element Distinctness problem based on Grover's quantum search algorithm. We revisit the problems using the recent quantum algorithm for Element Distinctness based on a quantum walk [1]. We also show new lower bounds and study new problems, in particular problems on polygons and the 3-Sum problem.

2 Quantum ingredient

The main quantum ingredient to our algorithms will be Andris Ambainis' quantum walk algorithm. It can be described in a language of databases, which we reproduce here, for more details we refer to [1, 5]. In a very abstract level, suppose we have a function $f\colon \{1, ..., n\} \to X$ and we wish to find a k-tuple $(i_1, ..., i_k)$ such that we have $R(i_1, ..., i_k, f(i_1), ..., f(i_k))$ for some fixed integer k and fixed relation R. We call it a *good tuple*. Moreover suppose we have database storing a subset of $\{1, ..., n\}$ and its image by f, which can be implemented on a quantum computer, meaning that operations on it can be done in superposition. Let s be the cost (in terms of queries to f) of setting up the database, u the cost for adding or removing an element, c the cost of checking whether it contains a good tuple. Then the quantum walk algorithm finds a good tuple with high probability, provided there is one, in $\tilde{O}(s + \frac{n}{k}(c + \sqrt{k}u))$ queries to f. Typically the database consists of r elements, and the setup cost is $s = r$, the update cost $u = 1$ and the checkup cost $c = 0$, which gives an algorithm with $\tilde{O}(n^{2/3})$ queries for the optimal choice $r = n^{2/3}$.

3 Decision problems on polygons

As a warm-up we study the quantum query complexity for very basic problems and show tight

*. This document has been written using the GNU TEXMACS text editor (see www.texmacs.org).; This work was done in spring/summer 2004 while A.B. was visiting the LRI, T.L. was at training at LRI with C.D and R.K. was on training at ENS, Paris.

bounds in most cases. The proofs are very simple.

A polygon is given as a table of n points $P = p_1, ..., p_n$ on the two-dimensional space. The polygon is then defined by the segments $\overline{p_1p_2}, ..., \overline{p_{n-1}p_n}, \overline{p_np_1}$. A polygon is simple if the segments do not intersect, except endpoints of successive segments. The region circumscribed by the polygon is defined as follows. Let o be a point which x-coordinate is strictly less than the x-coordinates of all points in P and which y-coordinates are strictly less than the y-coordinate of all points in P. Then o is defined to be outside of P. Roughly the inside of P consists of all points q such that the segment $\overline{oq}$ intersects an odd number of segments of P. (A correct definition needs to take care of the case, when one or worse two points of P lay on the segment $\overline{oq}$. So in a correct definition the segment $\overline{oq}$ should be replaced by a closed path avoiding points of P.)

The orientation of P is defined as follows. Let q be a point in P. Let $\overline{p_ip_{i+1}}$ be the segment of P intersecting $\overline{oq}$, with the intersection point closest to o. Then if the angle between $\overline{oq}$ and $\overline{p_ip_{i+1}}$ is negative, the orientation is *normal*, also called *counter-clockwise* and otherwise it is *clockwise*. A simple, normal oriented polygon is convex, if the angle between $\overline{p_{i-1}p_i}$ and $\overline{p_ip_{i+1}}$ is less or equal π.

We are interested in bounded error algorithms for various problems. These are algorithms which give the correct answer with probability at least 2/3. By repetition, the error probability can be made exponentially small. Since it is very hard to show lower bounds on the required running time of such algorithms, we focus on the query complexity. In this section the query model is the following. The algorithm is given n (and q for the membership problems) and has not direct access to the input string, but get the point p_i in return of index i. Clearly the query complexity of an algorithm is a lower bound of its time complexity in that model, and in many cases both measures are of the same order. We summarize in a table our results.

bounded-error query complexity	quantum	classical
Test if a given polygon P is simple ?	$\Omega(\sqrt{n}), O(n^{2/3})$	$\Theta(n)$
Test orientation of a simple polygon	$\Theta(\sqrt{n})$	$\Theta(n)$
Test convexity of a simple polygon	$\Theta(\sqrt{n})$	$\Theta(n)$
Point membership of a polygon	$\Theta(n)$	$\Theta(n)$
Point membership of a simple polygon	$\Theta(\sqrt{n})$	$\Theta(n)$
Point membership of a convex polygon	$\Theta(\log n)$	$\Theta(\log n)$

The classical bounded error lower bounds follow immediately from the reductions introduced in this section.

These results use reductions between problems. Reductions in the query model are different from reductions in the standard model. In the later we transform an input x of the first problem L into an input x' of the second problem L', with the property that the new instance is positive (has a solution) if and only if the initial instance is positive, and some other natural restrictions on the length of x' and the time to compute it. In the query model, we define the new input x' such that every query can be answered by a constant number of queries to x.

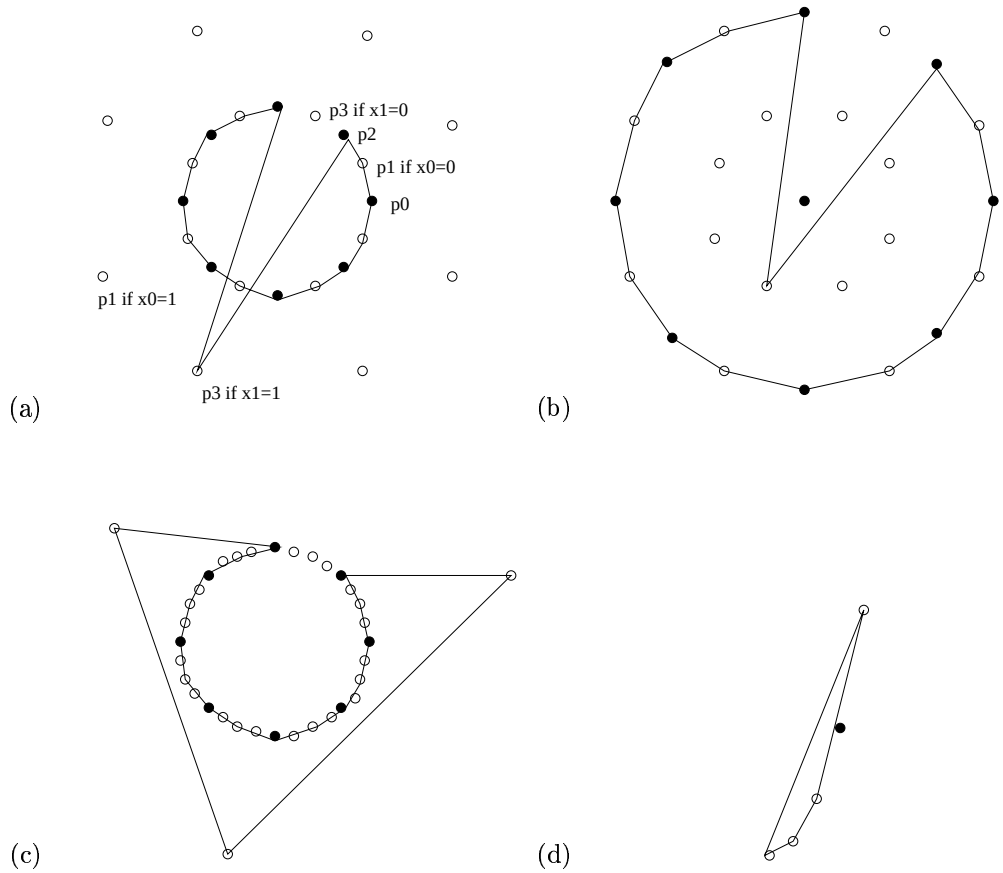

Figure 1. Reductions (a) from OR to testing if a polygon is simple, and from XOR to testing membership of a point, (b) from OR to testing convexity, and to testing membership of a point in a simple polygon, (c) from OR to testing orientation of a simple polygon, (d) from searching in an ordered list to testing membership of a point in a convex polygon.

To show the following lower bounds we use a reduction from the OR-problem: we are given an n-bit string x and wish to decide whether it is all zero or not. It is well-known that even under the promise that there is at most one 1 in x, the bounded error quantum query complexity is $\Theta(\sqrt{n})$.

Lemma 1. *The bounded error quantum query complexity of testing if a given polygon is simple is* $\Omega(\sqrt{n})$.

Proof. We reduce the OR-problem to this test, see figure 1a. Let be the n-bit string $x_0x_1...x_{n-1}$. We define a polygon P on $2n$ points as follows. For every $0 \leq i \leq n-1$, in polar coordinates, $p_{2i} := (\pi i/n, 1)$, $p_{2i+1} := (\pi(i+1)/n, 1)$ if $x_i = 0$ and $p_{2i+1} := (\pi + \pi(i+1)/n, 2)$ if $x_i = 1$.

Now P is simple if and only if x contains only zeros. Each query to P requires at most one query to x. This establishes the required lower bound. □

The same construction can be used to reduce XOR to testing point member ship in a polygon. Indeed P contains the origin $o := (0,0)$ if and only if x has even parity. Since it is well-known that the bounded error quantum query complexity of XOR is $\Omega(n)$, this establishes the lower bound.

A slightly different reduction is used to reduce OR — with the promise that x contains at most one 1 — to testing convexity of a simple polygon: we simply exchange radians 1 and 2 in the construction above, see figure 1b. Then if x is all zero, P is convex and contains the origin, and if x contains a single 1, then P is simple, but not convex, and does not contain the origin. So, the same reduction shows a lower bound for the problem of testing point-membership in a simple polygon.

For testing the orientation of a simple polygon we construct a polygon P on $3n$ points, where for all $0 \le i \le n-1$ and $0 \le j \le 2$ we set $p_{3i+j} := (\pi(3i+j)/n, 1)$ if $x_i = 0$ and $p_{3i+j} := (\pi(j+1)/3 + \pi 3i/n, w)$ for a large enough distance w, see figure 1c. Then if x is all zero, P is normal oriented. If there is a unique i with $x_i = 1$, P is clockwise oriented. In both cases it is simple.

Lemma 2. *The bounded error quantum query complexity of testing whether a given point q is contained in a given convex polygon P is $\Omega(\log n)$.*

Proof. This time we use a reduction for the problem of searching an integer x in a sorted table of positive integers $y_1, ..., y_n$, see figure 1d. We set $q = (x, x^2 + 1/2)$ and $p_i = (y_i, (y_i)^2)$. Then P is convex, and contains q if and only if $x = y_i$ for some index i. It is well-known that the bounded error query complexity of the search problem is $\Omega(\log n)$. □

For the upper bounds we essentially make a reduction to the OR-problem. For example to test whether a given simple natural oriented polygon is not convex, we have to search an index i such that the angle between $\overline{p_{i-1}p_i}$ and $\overline{p_i p_{i+1}}$ is strictly more than π. We simply Grover's search algorithm. To test the orientation of a simple polygon we use the quantum minimum search algorithm to find in $O(\sqrt{n})$ queries the point p_i with minimal x-coordinate. Then we just need to compute the angle between $\overline{p_{i-1}p_i}$ and $\overline{p_i p_{i+1}}$. To test whether a given point q is contained in a simple polygon P, we again apply the quantum minimum search algorithm use $O(\sqrt{n})$ queries to find the segment $\overline{p_i p_{i+1}}$ which is closest to q. Then we test the orientation of P, and test on which side of the segment q lays.

Lemma 3. *The bounded error quantum query complexity of testing whether a given point q is contained in a given convex polygon P is $O(\log n)$.*

Proof. Let o be some point in P, for example the median of 3 arbitrary points of P. Using binary search we detect the two segments of P intersected by the line $\overline{oq}$. Then we simply decide whether q lays in between the two intersections. □

Lemma 4. *The bounded error quantum query complexity of testing whether a given polygon P is simple is $O(n^{2/3})$.*

Proof. (sketch) We simply adapt Ambainis' quantum walk algorithm for element distinctness. In the later problem, we are given a table $x_0, ..., x_{n-1}$ of n integers, and wish to decide whether there exist indices $i \neq j$ such that $x_i = x_j$. In this case we call i, j a collision pair. We simply replace the equality test $x_i = x_j$ with the test of deciding if the segments $\overline{p_i p_{i+1}}$ and $\overline{p_j p_{j+1}}$ intersect. This gives the required upper bound on the query complexity. However the time complexity will be worse than for element distinctness since we cannot construct the equivalent of a hash table for a subset of indices S in order to decide quickly if a given index i is in collision with some index $j \in S$. □

4 3-Sum

There are several problems in computational geometry which generalize Element Distinctness. One of it is 3-Sum. The problem 2-Sum is essentially Element Distinctness: we are given a set of n integers S, and wish to decide whether there are two integers $a, b \in S$ such that $a + b = 0$.

For 3-Sum, we are given a set of n integers S and wish to decide whether there are three integers $a, b, c \in S$ such that $a + b + c = 0$. This is a very important problem in computational geometry and various problems have been reduced to it, ranging from deciding whether 3 out of n points are aligned, to whether a given triangle is completely covered by n other given triangles [4]. We show the following results. The lower bound for 3-Sum follows simply from the fact that it generalizes 2-Sum.

bounded-error query complexity	quantum	classical
2-Sum	$\Theta(n^{2/3})$	$O(n \log n)$
3-Sum	$\Omega(n^{2/3})$, $O(n \log n)$	$O(n^2)$

A very simple classical deterministic algorithm for 3-Sum is the following. First in time $O(n \log n)$ sort S which is given in form of a table. Then for every $a \in S$, let $a + S$ be the sorted table consisting of all elements $a + b$ where $b \in S$ and $-S$ be the (descending sorted) table, consisting of all elements $-c$ for $c \in S$. For a fixed element a we apply synchronized table scanning to find in time $O(n)$ whether the two tables have an element in common, that is $a + b \in a + S$ and $-c \in -S$ such that $a + b + c = 0$. Ranging over all values $a \in S$ this gives an algorithm with running time $O(n^2)$.

There is a lower bound of $\Omega(n^2)$ for this problem in the model of algebraic decision trees [2]. We are not aware of any results concerning bounded error algorithms for 3-Sum.

Lemma 5. *The bounded error quantum query complexity of* 3-*Sum is* $O(n \log n)$.

Proof. We proceed as in the classical case, except that for looking if two sorted tables of length n have an element in common, we can use a bounded error quantum algorithm from [3] which uses $O(\sqrt{n} \log n)$ queries — $O(\sqrt{n}\, c^{\log^\star n})$ to be precise. For a fixed integer $a \in S$ the procedure above has a success probability of $O(1/n)$. Using amplitude amplification, we get the required number of queries. □

5 Closest/Furthest pairs

The input of this problem is a table of n points, and we wish to find two points p_i and p_j with $i \neq j$ such that their distance is minimal among all pairs.

bounded-error query complexity	quantum	classical
Closest pair in 2-dimensions	$\Omega(n^{2/3})$, $O(n^{2/3} \log n)$.	$\Theta(n)$
Furthest pair in 2-dimensions	$\Omega(n^{2/3} \log^{-1/2} n)$	$\Theta(n)$

The problem Element Distinctness can be seen as asking whether there are two out of n points on a line, which have distance 0. The corresponding optimization problem asks for the closest pair, among n given points. Here we consider the 2-dimensional space, and consider the Euclidean distance L_2. Not surprisingly, what we found out, is that the problem has essentially the same query complexity than Element Distinctness. Clearly the closest pair problem has bounded error quantum query complexity $\Omega(n^{2/3})$, by a direct reduction from Element Distinctness, the function f defines n points on the 1-dimensional space, which contains a pair of distance 0, if and only if f contains a collision pair. For the furthest pair problem, the reduction is more tricky.

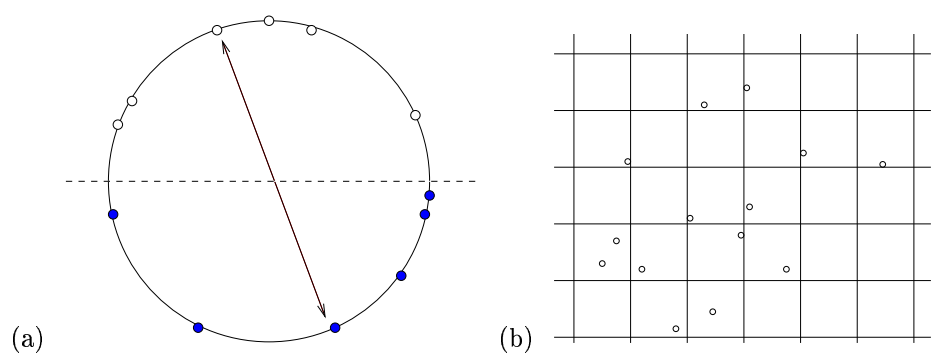

Figure 2. (a) reduction from Element Distinctness to furthest points: point i is on the unit circle with an angle depending on $f(i)$, and on the upper or lower half, depending the k-th bit in binary expansion of i.

(b) To find a closest pair, we divide the space into grid cells, and essentially use the algorithm for Element Distinctness.

Lemma 6. *The quantum query complexity of the furthest pair problem is* $\Omega(n^{2/3} \log^{-1/2} n)$.

Proof. First we reduce ELEMENT DISTINCTNESS to CLAW FINDING. Let be an instance of the ELEMENT DISTINCTNESS problem $h\colon [2n] \to \mathbb{N}$, a function for which we want to find a collision pair. An instance of Claw Finding consists of two functions f, $g\colon [n] \to \mathbb{N}$ and wish to find two indices i, j such that $f(i) = g(j)$. For a fixed bit position $0 \le p \le \log_2 n$ we define s to be a function which inserts a bit at position p to a given integer, i.e. $s(u2^p + v, p, b) = u2^{p+1} + b2^p + v$, where b is a bit, and u, v non-negative integers with $v < 2^p$. We define $f(i) = h(s(i, p, 0))$ and $g(i) = h(s(i, p, 1))$. Now if h is injective, then there is now a claw between f and g, otherwise if $h(i) = h(j)$ for some indices $i \neq i$ there is a claw between f and g for those bit-positions p where i and j differ. So using amplitude amplification, any algorithm for CLAW FINDING using T queries can be turned into a bounded error quantum algorithm for ELEMENT DISTINCTNESS using $O(T \log^{1/2} n)$ queries.

We define a list of $2n$ points, where for every $0 \le j < n$ we define $p_j := (\pi g(j)/n, 1)$ and $p_{n+j} := (\pi + \pi f(j)/n, 1)$ in polar coordinates. Now there is a pair of points which have distance 2 iff the claw finding problem has a solution. The lower bound $\Omega(n^{2/3})$ for ELEMENT DISTINCTNESS gives the required bound. □

We consider the problem of deciding whether there is a pair of distance less than some given threshold δ. Further we can assume that such a pair is unique. We proceed as follows. Let b_x, $b_y \in \{0, 1\}$ be two randomly chosen bits. We divide the space into a regular $2\delta \times 2\delta$ grid, every cell (a, b), for integers a, b consists of all points (x, y) with $(2a + b_x)\delta \le x < (2(a+1) + b_x)\delta$ and $(2a + b_y)\delta \le y < (2(a+1) + b_y)\delta$.

Lemma 7. *Suppose that there are two points u,v which are at a distance less than δ. Then with probability greater than $\frac{1}{4}$, both these points lie inside the same cell of the grid.*

Proof. Consider projections of u,v on x-axis. Distance between the projections will also be less than δ. The grid will divide the x-axis into the intervals of length 2δ. With probability at least half, both the projections will lie in the same interval.

The same is true with projections on y-axis. u,v will be in the same box iff both x and y projections lie in the same interval on x and y-axes respectively. So, probability of u,v being in the same box is greater than $\frac{1}{4}$. □

Lemma 8. *The quantum query complexity the closest pair problem is $O(n^{2/3} \log n)$.*

Proof. We assume that there is a single pair of points laying inside a box. We define the relation R, where $R(i, j, p_i, p_j)$ if $i \neq j$ and moreover p_i and p_j lay in the same box and are of distance less then δ. In this case we say that p_i, p_j is a *good pair*. By the assumption that there is at most one pair of distance closer than δ we deduce that every box contains at most 10 points. Now we define a database, where we store r points in a table which contains for every box 10 slots for points. In addition we have a counter (or flag) for the number of good pairs. The setting cost is $O(r)$, the update cost is $O(1)$ and the checking cost is 0. Using the quantum walk algorithm we get the required query complexity for $r = n^{2/3}$. □

6 Final remarks

It remains to show an upper bound for the quantum bounded error query complexity of the FURTHEST Points problem. But more important seems to reduce the gap between lower and upper bounds for the 3-Sum problem, which is very elementary and a central problem in computational geometry.

Bibliography

[1] A. Ambainis. Quantum walk algorithm for element distinctness. In Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS), pages 22–31, 2004.

[2] Michael Ben-Or. Lower bounds for algebraic computation trees. In Proceedings of the Annual ACM Symposium on the Theory of Computing (STOC), pages 80–86, 1983.

[3] Harry Buhrman, Christoph Dürr, Mark Heiligman, Peter Høyer, Frédéric Magniez, Miklos Santha, and Ronald de Wolf. Quantum algorithms for element distinctness. In IEEE Conference on Computational Complexity, pages 131–137, 2001. to appear in SIAM Jour. on. Computing.

[4] Anka Gajentaan and Mark H. Overmars. On a class of $o(n^2)$ problems in computational geometry. Comput. Geom. Theory Appl., 5:165–185, 1995.

[5] F. Magniez, M. Santha, and M. Szegedy. Quantum algorithms for the triangle problem. In Proceedings of 16th ACM-SIAM Symposium on Discrete Algorithms, 2005.

[6] Kunihiko Sadakane, Norito Sugawara, and Takeshi Tokuyama. Quantum algorithms for intersection and proximity problems. In Algorithms and computation (Christchurch, 2001), volume 2223 of Lecture Notes in Comput. Sci., pages 148–159. Springer, Berlin, 2001.

[7] Kunihiko Sadakane, Norito Sugawara, and Takeshi Tokuyama. Quantum computation in computational geometry. Interdiscip. Inform. Sci., 8(2):129–136, 2002.

[8] Warren D. Smith. Sublinear time quantum algorithms in computational geometry and related areas. unpublished manuscript, 2001.