Quantum Algorithms for Computational Geometry Problems

Andris Ambainis [ORCID]

Faculty of Computing, University of Latvia, Raina bulvaris 19, Riga, LV-1586, Latvia
ambainis@lu.lv

Nikita Larka

Faculty of Computing, University of Latvia, Raina bulvaris 19, Riga, LV-1586, Latvia
nikitalarka@gmail.com

─── **Abstract** ───

We study quantum algorithms for problems in computational geometry, such as POINT-ON-3-LINES problem. In this problem, we are given a set of lines and we are asked to find a point that lies on at least 3 of these lines. POINT-ON-3-LINES and many other computational geometry problems are known to be 3SUM-HARD. That is, solving them classically requires time $\Omega(n^{2-o(1)})$, unless there is faster algorithm for the well known 3SUM problem (in which we are given a set S of n integers and have to determine if there are $a, b, c \in S$ such that $a + b + c = 0$).

Quantumly, 3SUM can be solved in time $O(n \log n)$ using Grover's quantum search algorithm. This leads to a question: can we solve POINT-ON-3-LINES and other 3SUM-HARD problems in $O(n^c)$ time quantumly, for $c < 2$?

We answer this question affirmatively, by constructing a quantum algorithm that solves POINT-ON-3-LINES in time $O(n^{1+o(1)})$. The algorithm combines recursive use of amplitude amplification with geometrical ideas. We show that the same ideas give $O(n^{1+o(1)})$ time algorithm for many 3SUM-HARD geometrical problems.

1 Introduction

The 3SUM problem is as follows: given a set of numbers S, do there exist $a, b, c \in S$ such that $a + b + c = 0$? There is a nearly trivial classical algorithm that solves this problem in time $O(n^2)$. More advanced algorithms give only a logarithmic improvement to this quadratic complexity [15]. It is conjectured that no classical algorithm can solve 3SUM problem in $O(n^{2-\epsilon})$ time.

Many problems in computational geometry (for example, determining whether a given set of points contains 3 points that lie on a line) also seem to require $\Omega(n^2)$ time classically. Gajentaan and Overmars [12] showed that the 3SUM problem can be embedded into them. This implies that they cannot be solved in $O(n^{2-\epsilon})$ time, unless the 3SUM problem can also be solved in $O(n^{2-\epsilon})$ time. Such problems are called 3SUM-HARD. Besides 3 points on a line, examples of 3SUM-HARD problems include determining whether a given set of points contain 3 points that lie on a line, determining whether a given set of triangles covers given polygon, and determining whether a given set of axis-parallel segments are separable with a line into two nonempty subsets [12].

Quantum computing allows designing quantum algorithms that outperform classical algorithms. One such example is Grover search [14] which achieves quadratic speedup over classical exhaustive search and can be used as a subroutine to speedup more complicated problems [3, 17].

In particular, the 3Sum problem can be solved by a quantum algorithm in $O(n \log n)$ time, by a fairly simple application of Grover search procedure. Indeed, we can do an exhaustive search over pairs $a, b \in S$ and look for $-(a + b) \in S$ using some data structure (for example, we can use a balanced search tree). However, a direct application of Grover search does not give a quadratic speedup for many geometrical 3Sum-Hard class problems. For example, if we need to determine whether a set of points contain three points that lie on the same line, we need to search for all possible triplets of points, which results in $O(n^{\frac{3}{2}})$ time quantum algorithm [11].

In this paper we combine quantum effects with more sophisticated geometric techniques to design a quantum algorithm with complexity $O(n^{1+o(1)})$ for Point-On-3-Lines problem. We use ideas from this algorithm to solve many other 3Sum-Hard problems in time $O(n^{1+o(1)})$.

Related work.   The he 3Sum problem has been studied in the context of query complexity and it can be solved with $O(n^{3/4})$ queries, as it is a special case of the subset finding problem of Childs and Eisenberg [10] in which one has to find constant-size subset S of an n-element set, with S satisfying a certain property. A matching $\Omega(n^{3/4})$ quantum query lower bound is known [6]. However, the subset finding algorithm of [10] does not have a time-efficient implementation in the general case. Some special cases (for example, the element distinctness and k-distinctness algorithms of [2]) can be implemented efficiently but no efficient implementation is known for the 3-SUM case.

We think that it is unlikely that this line of work would lead to an $o(n)$ time quantum algorithm for the 3Sum problem. The element distinctness algorithm [2] and the subset finding algorithm [10] are special cases of a quadratic speedup for hitting times of Markov chains [18, 4, 5]. It is unlikely that these methods will lead to a quantum algorithm that is more than quadratically faster than the best classical algorithm.

More generally, we conjecture that the 3Sum problem cannot be solved in $O(n^{1-\epsilon})$ quantum time in the QRAM model, neither with methods based on subset finding nor any oher approach. This could serve as a basis for a quantum version of fine-grained complexity, similarly to recent quantum fine grained lower bounds of [1, 8] based on quantum versions of Strong Exponential Time Hypothesis (SETH).

2   Preliminaries

2.1   Problems

Here we define problems we focus on in this paper. All of them belong to 3Sum-Hard class. [12]

- Point-On-3-Lines: Given a set of lines in the plane, is there a point that lies on at least three of them? [12]
- 3-Points-On-Line: Given a a set of points in the plane, is there a line that contains at least three points? [12]
- Strips-Cover-Box: Given a set of strips in the plane (strip is defined as an infinite area between two parallel lines (see Figure 1)), does their union contain a given axis-parallel rectangle? [12]

- TRIANGLES-COVER-TRIANGLE: Given a set of triangles in the plane, does their union contain another given triangle? [12]
- POINT-COVERING: Given a set of n half-planes and a number t, determine whether there is a point that is covered by at least t half-planes. [12]
- SEGMENT-SEPARATOR: Given a set of vertical line segments, does there exists a non vertical line that does not intersect any of given segments and contaisn at least one given segment in each of two half-planes? [12]
- VISIBILITY-BETWEEN-SEGMENTS: Given a set of n vertical line segments S and two particular line segments s_1 and s_2, determine whether there is a point on s_1 and a point on s_2, such that segment between these two points doesn't intersect any segment from S. [12]

We also define GENERAL-COVERING problem. We will design quantum algorithm for this problem with $O(n^{1+o(1)})$ complexity and then reduce many 3SUM-HARD problems to this problem.

- GENERAL-COVERING: We are given a set of n strips and angles (angle is defined as an infinite area between two non-parallel lines (see Figure 2)) in the plane. The task is to find a point X that satisfies the following conditions:
  - the point X is an intersection of two angle or strip boundary lines ℓ_1, ℓ_2 (ℓ_1 and ℓ_2 may be boundary lines of two different angles/strips);
  - the point X does not belong to the interior of any angle or strip;
  - the point X satisfies a given predicate $P(X)$ that can be computed in $O(1)$ time.
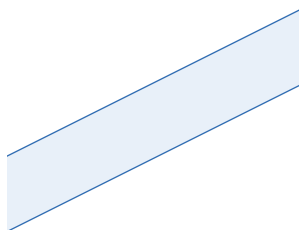
Figure 1 Strip.
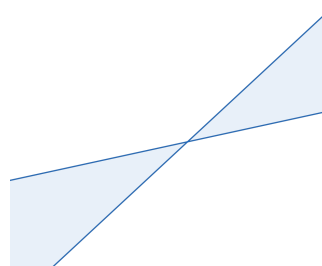
Figure 2 Angle.

2.2 Model

We assume a query model in which the query returns the description D_i of the ith object (point, line, strip, triangle, etc.), given i. The description consists of several numbers that specify the ith object (e.g. coordinates of a point or values of coefficients in the equation that specifies a line). In the quantum case, we can query superpositions of indices i. The input to quantum query Q consists of two registers, with one register holding i and the other register provides the space for D_i. The query transformation acts as $Q|i, x\rangle = |i, x \oplus D_i\rangle$. In particular, given a superposition $|\psi\rangle = \sum_i \alpha_i |i, 0\rangle$ in which $x = 0$, applying Q gives the state $Q|\psi\rangle = \sum_i \alpha_i |i, D_i\rangle$. Applying Q to $|\phi\rangle = \sum_i \alpha_i |i, D_i\rangle$ gives the state $Q|\phi\rangle = \sum_i \alpha_i |i, 0\rangle$ in which the descriptions D_i are erased from the second register.

Our algorithms work in the commonly used QRAM (quantum random access memory) model of computation [13] which assumes quantum memory can be accessed in a superposition. QRAM has the property that any time-T classical algorithm that uses random access memory

can be invoked as a subroutine for a quantum algorithm in time $O(T)$. We can thus use primitives for quantum search (e.g., Grover's quantum search or Amplitude amplification) with conditions checking which requires data stored in a random access memory.

2.3   Tools

We will use two well known quantum procedures in our algorithm.

▶ **Theorem 1** (Grover search [16])**.** *Given a set of n elements $X = \{x_1, x_2, ..., x_n\}$ and a boolean function $f : X \to \{0, 1\}$. The task is to find $x \in X$ such that $f(x) = 1$. There is a bounded-error quantum procedure that solves this problem using $O(\sqrt{n})$ quantum queries.*

▶ **Theorem 2** (Amplitude amplification [7])**.** *Let A be a quantum procedure with one-sided error and success probability at least ϵ. Then, there is a quantum procedure B that solves the same problem with success probability $\frac{2}{3}$ invoking A $O(\frac{1}{\sqrt{\epsilon}})$ times.*

Note that any constant success probability $1 - \epsilon$ can be achieved with repeating Amplitude Amplification constantly many times.

We will also use the following well known computational geometry algorithm.

▶ **Theorem 3** (Arrangement of lines [9])**.** *Given a set of n lines in the plane, we can compute partition of the plane formed by those lines in time $O(n^2)$.*

We will also use point-line dualization for problem reductions. Point-line dualization is a plane transformation that maps points to lines and lines to points in the following way:
- Line $\ell : y = ax + b$ is mapped to point $\ell^* = (a, -b)$
- Point $P = (a, b)$ is mapped to line $P^* : y = ax - b$

One may note, that the following properties are true:
1. $(P^*)^* = P$ and $(\ell^*)^* = \ell$
2. $P \in \ell \iff \ell^* \in P^*$
3. If point A, B, C lie on one non-vertical line, then lines A^*, B^*, C^* meet at one point.
4. If non-vertical lines p, q, r meet at one point, then points p^*, q^*, r^* lie on one line.
5. Points from vertical line segment are mapped to a strip.
6. Points from non-vertical line segment are mapped to an angle.

3   Point on three lines

In this part we describe a quantum algorithm which solves POINT-ON-3-LINES problem in $O(n^{1+o(1)})$ time, improving over the $O(n^{3/2})$ time quantum algorithm of Furrow [11]. The idea behind our algorithm is as follows. Suppose we are given a set S of lines in the plane. From this set we randomly pick k lines. Those k lines split plane into no more than $\binom{k+1}{2} + 1$ regions. Each line from set S intersects $k + 1$ regions. So, on average, a region contains $O(\frac{n}{k})$ lines crossing that region.

These facts give an opportunity to design a quantum algorithm that improves over the complexity $O(n^{\frac{3}{2}})$ of simple quantum search (as in Furrow's algorithm [11]). If we have a point A, then we can decide if A belongs to some line from S using $O(\sqrt{\frac{n}{k}} + k)$ quantum time. To do so, we need to find a region which contains point A and check if A belongs to a line which crosses that region. For finding the region, $O(k)$ time suffices. Checking if A belongs to a line from S can be done by Grover's search over $O(\frac{n}{k})$ lines that cross this region. For this, $O(\sqrt{\frac{n}{k}})$ time suffices.

If we need to find three lines that intersect in one point, we run Grover search over all pairs of lines (ℓ_i, ℓ_j). For each pair, we find the intersection point $P_{i,j} = l_i \cap l_j$ and check if point $P_{i,j}$ belongs to a third line using algorithm described earlier. If the subdivision of the plane into regions can be done in time $O(nk^2)$, this algorithm runs in time $O(nk^2 + \sqrt{n^2}(k + \sqrt{\frac{n}{k}}))$. Setting $k = n^{\frac{1}{5}}$ gives $O(n^{\frac{7}{5}})$. But we can find even better algorithm. After dividing the plane into regions, instead of searching for an intersection point of three lines, we search for a region which has this point (search is done using Grover search) and we recursively apply $O(n^{\frac{7}{5}})$ algorithm to find the intersection point of three lines inside that region. We can then add more levels of recursion to decrease the complexity further. We now describe the final algorithm (in which we recurse at the optimal choice of k and the number of levels of recursion grows with n).

Let S be the given set of lines in a plane. Let P be a subset of S containing exactly k lines. Lines in P divide plane into convex (possibly infinite) polygons. We arbitrarily triangulate regions, which are bounded by at least 4 lines. This results in a subdivision of the plane into regions $R_1, R_2, ..., R_t$ where each region is bounded by at most 3 segments (see Figure 3). Let $s(R_i) = \{\ell \mid \ell \in S \text{ and } \ell \text{ intersect } R_i\}$.
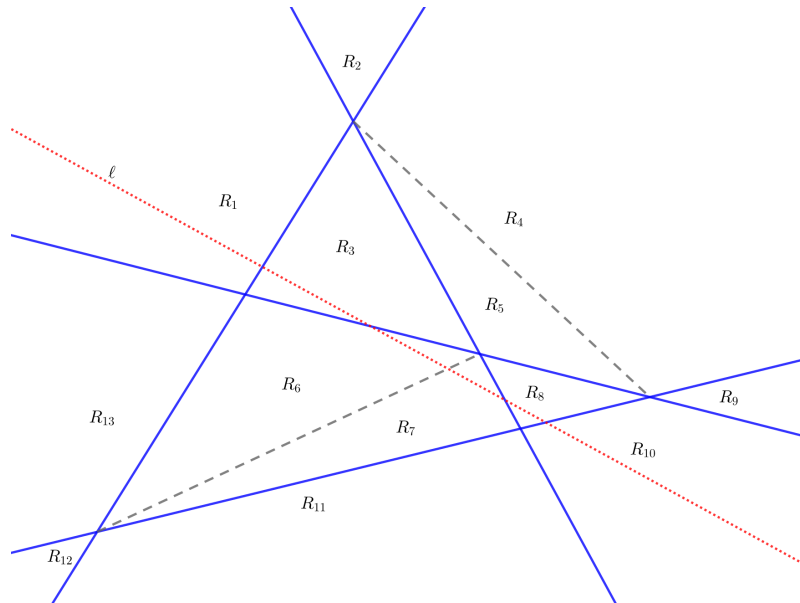
Figure 3 Blue lines divide plate into 13 regions: $R_1, R_2, ..., R_{13}$. Red line ℓ passes through regions: $\ell \in s(R_1), s(R_3), s(R_6), s(R_7), s(R_8), s(R_{10})$.

We start with the following three observations:

▶ **Lemma 4.** *If after the triangulation we get t regions: $R_1, R_2, ..., R_t$, then $t \le 2|P|^2 = 2k^2$*

Proof. If f_i is the number of faces bounded by i lines before the triangulation, then

$$t = f_1 + f_2 + f_3 + 2f_4 + 3f_5 + ... + (k-2)f_k \le \sum_{i=1}^{k} i f_i \le 2k^2 \tag{1}$$

The last inequality holds because $\sum_{i=1}^{k} i f_i$ is equal to twice the total number of line segments before the triangulation (because each line segment is on the boundary of two faces, one on each side) and the number of line segments is at most k^2 (because each of k lines is split by the other $k-1$ lines into at most k segments). ◀

▶ **Lemma 5.** *Sets $s(R_1), ..., s(R_t)$ can be built classically in time $O(|S| \times |P|^2)$.*

Proof. We can construct regions $R_1, R_2, ..., R_t$ in time $O(|P|^2)$ using Theorem 3. We build $s(R_i)$ by iterating through each line from S and checking whether the line intersects region R_i. This step takes time $O(|S| \times t) = O(|S| \times |P|^2)$.   ◀

▶ **Lemma 6.** *If P is chosen uniformly at random, then*

$$\Pr[\max_i |s(R_i)| \geq 3\frac{|S|}{|P|}(5\log(|S|) + \log(\epsilon^{-1}))] \leq \epsilon \tag{2}$$

Proof. Let ℓ be an arbitrary line (possibly not from the set S). Lines from set S intersect line ℓ in points $X_1, X_2, ..., X_m$ in this order (if two lines ℓ_i and ℓ_j intersect ℓ in the same point, then $X_i = X_j$). We note that $m \leq |S|$, since some lines from S might be parallel to ℓ. We color a point X_i with white color if the corresponding line ℓ_i from S is in the set P. Otherwise we color the point X_i with black color. We define $L = \left\lceil \frac{|S|}{|P|}(5\log(|S|) + \log(\epsilon^{-1})) \right\rceil$ and we assume that $L \leq |S|$, since otherwise lemma is obviously true.

We say that a line ℓ is bad, if there exists index i, such that X_{i+j} is black for all $j \in [0...L-1]$. The probability of ℓ being bad can be upper bounded as follows

$$\Pr\left[\bigvee_{i=1}^{m-L+1} (X_i, X_{i+1}, ..., X_{i+L-1} \text{ are all black}) \right] = \tag{3}$$

$$\leq \sum_{i=1}^{m-L+1} \Pr[X_i, X_{i+1}, ..., X_{i+L-1} \text{ are all black}] = (m - L + 1)\frac{\binom{|S|-L}{|P|}}{\binom{|S|}{|P|}} \tag{4}$$

$$\leq |S|\left(\frac{|S|-L}{|S|}\right)^{|P|} \leq |S|\left[\left(1 - \frac{L}{|S|}\right)^{\frac{|S|}{L}}\right]^{L\frac{|P|}{|S|}} \leq \frac{|S|}{e^{5\log|S|+\log\epsilon^{-1}}} = \frac{\epsilon}{|S|^4} \tag{5}$$

Consider set S' which consists of lines from S and lines that pass through at least two intersection points of lines from S. Then, every edge e of every region R_i lies on a line that belongs to S' (because e is either a segment of one of original lines from S or is created during the triangulation and, in the second case, both endpoints of e are intersection points of two lines from S) Since there are at most $\binom{|S|}{2}$ intersection points of lines from S, we have $|S'| \leq |S|^4$ and

$$\Pr[S' \text{ contains bad line}] \leq \sum_{\ell \in S'} \Pr[\ell \text{ is bad}] \leq |S'| \times \frac{\epsilon}{|S^4|} \leq \epsilon \tag{6}$$

To finish the proof, it is enough to see that the fact that S' doesn't contain bad line implies $|s(R_i)| \leq 3(L-1) < 3\frac{|S|}{|P|}(5\log(|S|) + \log(\epsilon^{-1}))$ for all i. Indeed, if no lines in S' is bad, then each side of each region R_i contains less than L black points. Since a black point corresponds to a line which intersects a region R_i and each region is bounded by at most three segments, the region R_i is intersected by no more than $3(L-1)$ lines from S.   ◀

▶ **Theorem 7.** *There is a bounded error quantum algorithm for* POINT-ON-3-LINES *problem, that runs in time $O(|S|^{1+o(1)})$.*

Proof. The algorithm has a parameter k and allowable error probability ϵ. The algorithm consists of a recursive procedure that takes a set of lines X as input and returns 3 lines from the set X which intersects at one point or tells that there are no such 3 lines.

Procedure $Algo_k(X)$.

---

if $|X| < k$ **then**
| Check for an intersection of 3 lines classically, by exhaustive search
end
$R_1, R_2, ..., R_t = \text{RandomPlaneSeparation}_{\frac{\epsilon}{2}}(X)$
Build sets: $s(R_1), s(R_2), ..., s(R_t)$
if $max_i |s(R_i)| > 3\frac{|X|}{k}(5\log(|X|) + \log(\frac{2}{\epsilon}))$ **then**
| return error
end
Let A be the algorithm that randomly chooses $j \in [t]$ and runs $Algo_k(s(R_j))$.
With Amplitude amplification, run A with the success probability amplified to at
    least $1 - \frac{\epsilon}{2}$.

The recursive procedure can be described as follows. If the input set X contains less than k lines, we solve POINT-ON-3-LINES classically in time $O(|X|^2)$. Otherwise, we split the plane into regions $R_1, R_2..., R_t$ with k random lines from X and build sets $s(R_1), s(R_2), ..., s(R_t)$. If there are three lines that intersect at one point, then this point is located in one of the regions R_i (if this point is on the boundary of a region, then this point can be found during the construction of sets $s(R_i)$). We use amplitude amplification to find the region R_i which contains intersection point of three lines and recursively apply the procedure to lines from $s(R_i)$.

Suppose that S contains three lines that intersect at one point. If $|S| < k$, the algorithm finds those 3 lines with certainty. Let $|S| \geq k$. The probability that $s(R_i) > 3\frac{|S|}{k}(5\log(|S|) + \log(\frac{2}{\epsilon}))$ for some R_i and the algorithm returns an error is less than $\frac{\epsilon}{2}$. By executing amplitude amplification (running $Algo_k(s(R_i))$ $O(\sqrt{t}) = O(k)$ times), we can reduce probability of not finding the 3 lines to $\frac{\epsilon}{2}$. So, $Algo_k(S)$ finds desired three lines with probability at least $1 - \epsilon$.

If $T(|X|)$ is a runtime of $Algo_k(X)$, then:

$$T(|X|) = O(|X| \times k^2) + O(\sqrt{k^2}) \times T\left(3\frac{|X|}{k}(5\log(|X|) + \log(\frac{2}{\epsilon}))\right) \tag{7}$$

If $k = |S|^{\frac{1}{\alpha}} \cdot 3(5\log(|S|) + \log(\frac{2}{\epsilon}))$, then:

$$T(|X|) \leq O(|X| \times |S|^{\frac{2}{\alpha}} \log^2(|S|)) + O(\sqrt{k^2}) \times T(|X| \times |S|^{-\frac{1}{\alpha}}) \tag{8}$$

There are $(C_1 k)^{2j}$ problems on recursion level j for some constant C_1 and each problem size is at most $|S|^{1-\frac{j}{\alpha}}$.

$$T(|S|) \leq \sum_{j=0}^{\alpha} \sqrt{(C_1 k)^{2j}} \times \left[|S|^{1-\frac{j}{\alpha}} \times |S|^{\frac{2}{\alpha}} \log^2(|S|)\right] =$$

$$= \sum_{j=0}^{\alpha} \left(\frac{C_1 k}{|S|^{\frac{1}{\alpha}}}\right)^j (|S|^{1+\frac{2}{\alpha}} \log^2(|S|)) \leq \tag{9}$$

$$\leq \alpha (C_2 \log(|S|))^{\alpha} (|S|^{1+\frac{2}{\alpha}} \log^2(|S|))$$

If $\alpha = \sqrt{\frac{2\log(|S|)}{\log(C_2) + \log\log|S|}}$, then $T(|S|) = O(\alpha |S|^{1+\frac{4}{\alpha}} \log^2(|S|)) = O(|S|^{1+o(1)})$ ◄

4   Other 3SUM hard problems

In this section, we show how it is possible to apply plane separation ideas, described in the previous section, to speed up other 3-SUM-HARD problems defined in [12].

- 3-POINTS-ON-LINE

  This problem is dual to the POINT-ON-3-LINES problem [12], and so is solvable with the same quantum algorithm in time $O(n^{1+o(1)})$, as described in the previous section.

- GENERAL-COVERING

  The given n strips and angles form $2n$ lines. We divide the plane into regions by randomly choosing k out of those $2n$ lines, similarly to the algorithm described in the previous section. A region and a strip/angle can be in one of the following relations: the strip/angle fully covers the region, the strip/angle partly covers the region or the strip/angle has no common points with the region. We can identify all regions that are fully covered by some strip/angle in time $O(nk^2)$. For regions that are not fully covered by some strip/angle, we identify the set $s(R_i)$ of strips and angles which cross that region. Non covered regions may contain the desired intersection point, but this intersection point is formed by the lines that are boundary lines of the strips/angles in the set $s(R_i)$. Similarly to the algorithm POINT-ON-3-LINES, our task is divided into $O(k^2)$ tasks, each of which involved $O(\frac{n \log n}{k})$ objects. The time complexity is

$$T(n) = O(nk^2) + O(\sqrt{k^2}) \times T\Big(3\frac{2n}{k}(5\log(2n) + \log(\frac{2}{\epsilon}))\Big). \tag{10}$$

  Similarly to the analysis of POINT-ON-3-LINES problem, we get $T(n) = O(n^{1+o(1)})$.

- STRIPS-COVER-BOX

  This problem is just the special case of the GENERAL-COVERING problem with the predicate $P(X)$ being true if the point X is located inside the given box. Then, the point X from GENERAL-COVERING problem corresponds to an uncovered point in STRIPS-COVER-BOX problem. So, STRIPS-COVER-BOX can also be solved in time $O(n^{1+o(1)})$.

- TRIANGLES-COVER-TRIANGLE

  The given n triangles consist of $3n$ segments. We extend each segment to a line and separate the plane into regions, similarly to the POINT-ON-3-LINES problem with randomly chosen k lines. A triangle and a region can be in one of the following relations: the triangle fully covers the region, the triangle partly covers the region or the triangle has no common point with the region. We can identify all regions that are fully covered by some triangle in time $O(nk^2)$. All other regions may contain a point X which is not covered by any triangle. Similarly to the POINT-ON-3-LINES problem, we search for the region R_i which contains that point. Note that, if a triangle partly covers the region R_i, then at least one of the segments that form this triangle is in $s(R_i)$. So, we can finish our algorithm, just as in GENERAL-COVERING problem, with the predicate $P(X)$ being true, if the point X is located inside the triangle that must be covered.

- POINT-COVERING

  The given n half-planes are specified by n lines. We separate the plane into regions, similarly to the POINT-ON-3-LINES problem, by randomly choosing k out of n given lines. For each region R_i we compute the number of half-planes r_i that fully cover this region. This takes $O(nk^2)$ time. To determine if there exists a point that is covered by at least t half-planes, we need to tell, if there exists a point inside a region R_i that is covered by at least $t - r_i$ half-planes from $s(R_i)$. As in the GENERAL-COVERING problem, the algorithm takes $O(n^{1+o(1)})$ time.

≈ VISIBILITY-BETWEEN-SEGMENTS

We dualize the given n vertical segments, to get n strips. We need to find a point that does not belong to any of the strips and has the property that the corresponding line in the initial plane intersects two given segments s_1 and s_2. This problem is just the special case of the GENERAL-COVERING problem, where the predicate $P(X)$ is true if the line corresponding to X intersects two given segments s_1 and s_2. Just like in the GENERAL-COVERING problem, this problem can also be solved in $O(n^{1+o(1)})$ time.

≈ SEGMENT-SEPARATOR

This problem can be solved in exactly the same way as the VISIBILITY-BETWEEN-SEGMENTS problem, with the only difference being the predicate $P(X)$. Now this predicate is true, if the line corresponding to a point separates the given segments in the way required for the SEGMENT-SEPARATOR problem. Since X is the intersection point of two lines in the dual plane, the line, corresponding to the point X, must go through two endpoints of two different given segments. So, $P(X)$ is false, if the corresponding line goes through an edge of the convex hull of the endpoints of given segments. This can be detemined in time $O(1)$, after we precompute the convex hull in time $O(n\log(n))$. This results in an $O(n^{1+o(1)})$ time quantum algorithm.

───── **References** ─────

1 Scott Aaronson, Nai-Hui Chia, Han-Hsuan Lin, Chunhao Wang, and Ruizhe Zhang. On the quantum complexity of closest pair and related problems. *CoRR*, abs/1911.01973, 2019. `arXiv:1911.01973`.

2 Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM J. Comput.*, 37(1):210–239, 2007. `doi:10.1137/S0097539705447311`.

3 Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 1783–1793, Philadelphia, PA, USA, 2019. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=3310435.3310542`.

4 Andris Ambainis, András Gilyén, Stacey Jeffery, and Martins Kokainis. Quadratic speedup for finding marked vertices by quantum walks. *CoRR*, abs/1903.07493, 2019. `arXiv:1903.07493`.

5 Simon Apers, András Gilyén, and Stacey Jeffery. A unified framework of quantum walk search. *CoRR*, abs/1912.04233, 2019. `arXiv:1912.04233`.

6 Aleksandrs Belovs and Robert Spalek. Adversary lower bound for the k-sum problem. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 323–328. ACM, 2013. `doi:10.1145/2422436.2422474`.

7 Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

8 Harry Buhrman, Subhasree Patro, and Florian Speelman. The quantum strong exponential-time hypothesis. *CoRR*, abs/1911.05686, 2019. `arXiv:1911.05686`.

9 Bernard Chazelle, Leo J. Guibas, and D. T. Lee. The power of geometric duality. *BIT Numerical Mathematics*, 25, 1985. `doi:10.1007/BF01934990`.

10 Andrew M. Childs and Jason M. Eisenberg. Quantum algorithms for subset finding. *Quantum Information & Computation*, 5(7):593–604, 2005. URL: `http://portal.acm.org/citation.cfm?id=2011663`.

11 Bartholomew Furrow. A panoply of quantum algorithms. *Quantum Information & Computation*, 8(8):834–859, 2008. URL: `http://www.rintonpress.com/xxqic8/qic-8-89/0834-0859.pdf`.

12 Anka Gajentaan and Mark H. Overmars. On a class of o(n2) problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995. `doi:10.1016/0925-7721(95)00022-2`.

13   Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100:160501, April 2008. `doi:10.1103/PhysRevLett.100.160501`.

14   Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM. `doi:10.1145/237814.237866`.

15   Allan Grønlund Jørgensen and Seth Pettie. Threesomes, degenerates, and love triangles. *CoRR*, abs/1404.0799, 2014. `arXiv:1404.0799`.

16   Lov K. Grover. Framework for fast quantum mechanical algorithms. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, 80, December 1997. `doi:10.1145/276698.276712`.

17   Frédéric Magniez, Miklos Santha, and Mario Szegedy. Quantum algorithms for the triangle problem. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 1109–1117, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=1070432.1070591`.

18   Mario Szegedy. Quantum speed-up of markov chain based algorithms. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 32–41. IEEE Computer Society, 2004. `doi:10.1109/FOCS.2004.53`.