Interdisciplinary Information Sciences, Vol. 8, No. 2, pp. 129–136 (2002)

Quantum Computation in Computational Geometry

Kunihiko SADAKANE*, Norito SUGAWARA* and Takeshi TOKUYAMA*

Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan

Received October 1, 2001; final version accepted July 2, 2002

We discuss applications of quantum computation to geometric data processing. These applications include problems on convex hulls, minimum enclosing balls, linear programming, and intersection problems. Technically, we apply well-known Grover's algorithm (and its variants) combined with geometric algorithms, and no further knowledge of quantum computing is required. However, revealing these applications and emphasizing potential usefulness of quantum computation in geometric data processing will promote research and development of quantum computers and algorithms.

1. Introduction

Quantum computation is one of recent computing paradigms that may give breakthrough against barriers for the standard RAM (random access machine) model computation. Although the computational ability of current quantum computers is far from practical, its potential power is quite attractive. The theory of quantum computing has two major trends: (1) investigating the computational complexity classes to structure new hierarchy by using quantum models (e.g. [4]), and (2) comparing theoretical efficiency of the quantum model with the RAM model. In the latter trend, positive results demonstrating advantage of the quantum computation to the RAM model computation give motivations for development of quantum computers. In the literature, Grover's data search algorithm [14] and Shor's factorization algorithm [23] have been major driving-forces for the recent development of quantum computers.

One major use of commercialized computers is to handle a data set consisting of d-dimensional vectors of real numbers. A set of d-dimensional vectors can be considered to be a set of n points in the d-dimensional Euclidean space. More generally, we can consider geometric data processing problems, where the input is a set of n geometric objects in a d-dimensional space.

For one dimensional vectors (i.e. real numbers), they can be ordered with respect to their values. In the RAM model, sorted list is a very powerful data structure to handle an ordered set (of size n), so that we can query/insert/delete a data in $O(\log n)$ time. Therefore, if each item of the data consists of a single real number, we often assume that data is sorted. Unfortunately, querying a data in a sorted list needs $\Omega(\log n)$ time in a quantum model [2]. This implies that the advantage of quantum computation is not very large for processing an sorted list.

If $d \geq 2$, in contrast to the one-dimensional case, no universal method to ''sort'' points in the space is known; indeed, a ''geometric sorted'' data structure highly depends on both the problem that we want to solve and the number of dimensions. Voronoi diagrams, convex hulls, and simplex range search data structures are examples of data structures for solving a variety of geometric problems; Unfortunately, they are not very effective if the number of dimensions is large (say, larger than three).

The aim of this paper is to show that the quantum model can be considerably advantageous to the RAM model for geometric data processing. The introduction of random sampling method [9, 10] has created a new trend in computational geometry, and enormous research results have been produced on the application of probabilistic methods to geometric data processing. In the same way, one may expect that quantum computation (regarded as a kind of biased random sampling method) opens new aspects of computational geometry.

We mainly deal with problems in the complexity class **P** (polynomial-time soluble problems in RAM model) in this paper, and want to consider the following question: ''What kind of geometric problems can be solved in sublinear time in the quantum model?''. In contrast to the parallel computation (PRAM) model where we seek for NC (i.e. polylogarithmic time and polynomial-size work) algorithms, it seems to be valuable to seek for sublinear time algorithms for the quantum model.

For example, consider the problem of finding the lowest point in the intersection of n upper-halfspaces in the d-dimensional Euclidean space. The problem can be solved in $\Theta(n)$ time if d is a constant in RAM model. Regarding the problem as a parametric minimax problem [25], and by combining the quantum minimum finding algorithm and multidimensional searching technique, we can solve it in $O(\sqrt{n}\log^{2d-1} n)$ time.

We do not introduce novel quantum operations for designing our algorithms; Indeed, little knowledge of quantum computation is necessary to read this paper, provided that we admit some well-known theorems on modified versions of Grover's algorithm. In other words, we introduce some advanced geometric functions in a quantum database system based on Grover's query algorithm.

* E-mail: {sada,norito,tokuyama}@dais.is.tohoku.ac.jp

We first utilize a view of Grover's algorithm as a biased random sampling method [15, 16], which we call *quantum sampling*, and show that several query problems in a set of n points can be done in $O(\sqrt{n})$ time for any d without any preprocessing. For example, the extremal query, the nearest neighbor query, and the farthest neighbor query are such queries. In the dual form, if the input is a set of n hyperplanes (or halfspaces), we can do the separation query and ray shooting query in $O(\sqrt{n})$ time.

Next, as a showcase problem of our approach, we consider the problem of computing the lowest point in the *upper envelope* of an arrangement of hyperplanes; in other words, the lowest point in the intersection of given upper halfspaces. Then, we show some other geometric optimization problems such as the minimum enclosing ball problem and linear programming problems can be solved efficiently in quantum model if d is a constant. Also, some output-sensitive convex-hull algorithms and the ellipsoid algorithm for solving a general dimensional convex programming problem can be accelerated by using the quantum model.

Finally, we discuss another approach based on the ideas in quantum element-distinctness algorithm [8], and apply it to some geometric intersection detection problems, although the details will be given in a separate paper [19].

Recently, we have informed that W. Smith also investigate similar geometric algorithms in the quantum model [24].

2. Quantum Sampling and Geometric Queries

2.1 Preliminaries on Grover's algorithm and its applications

Grover's database search algorithm can be considered to be a biased (or controlled) sampling method. Although this fact is well-known, we briefly introduce the outline.

We have a set S of n data items $p_1, p_2, \ldots, p_n \in U$, where U is a universe to which data items belong. We also have a function f from U to $\{1, -1\}$ such that $f(x)$ can be computed for each element x of U independently.

What we want is to find a data item p_i satisfying $f(p_i) = 1$ (we call such a data item *target data item* or *target data.*) If there is no target data in S, we report "none".

In the RAM model, a naive way is to check all the member of U in $O(n)$ time (here, we assume that $f(x)$ can be computed in $O(1)$ time for each x for simplicity). If there are k target data in S, random sampling is a well-known method: we randomly choose a sample from S until we find a target data. Unfortunately, this takes $O(n/k)$ time, and does not effectively work if there are few (or no) target data in S. On the other hand, in a parallel model such as CRCW PRAM, we can obviously solve the problem in $O(1)$ time by using n processors.

The quantum model is an intermediate model between them (note that the description in this paper is simplified so that it is sufficient for solving the problems discussed here): In the quantum model, the value $f(p_i)$ is computed for each i ($i = 1, 2, \ldots, n$) independently in parallel. We also compute a *state vector* $v = (v_1, v_2, \ldots, v_n)$ in a parallel fashion, so that we can read the contents (e.g., p_i and $f(p_i)$) of the i-th "processor" as an output with a probability $|v_i|^2/|v|^2$. A major restriction is that we can only apply parallel computations realized as unitary transformations on the state vector. Moreover, the read operation affects on the state vector, and hence, we cannot re-use intermediate state vectors once we read an output. Note that the classical random algorithm can be considered a special quantum algorithm where no interaction between processors is allowed.

We only need the following well-known result in the quantum computation, and the rest of the paper does not require knowledge from quantum computation at all.

Theorem 2.1 (Boyer et al. [6]). *If the number k of target data in S satisfies that* $k \geq k_0 > 0$, *we can read a target data with a probability larger than* $1/c$ *for a fixed constant c after* $O(\sqrt{n/k_0})$ *steps of quantum computation. Moreover, the output target data is randomly sampled from the set of all target data.*

The following corollary is also well known.

Corollary 2.1 (Dürr and Høyer [13]). *Given a real valued function g on S, we can report maximizing* $g(p_i)$ *in* $O(\sqrt{n})$ *time with a probability greater than* $1/2$ *in the quantum model.*

Theorem 2.1 and corollary 2.1 are essential tools in this paper. For the quantum oracle used in Grover's algorithm, we can use Deutsch–Jozsa algorithm [11].

2.2 Geometric queries

We show some straightforward quantum algorithms for geometric queries that are important subroutines in several geometric algorithms. Consider a set S of n points in the d-dimensional space. Many of geometric queries for the point set S are basically maximum-finding queries. For example,

Nearest-neighbor query: Answer the nearest point in S to a query point q.

Farthest-neighbor query: Answer the farthest point in S from a query point q.

Extremal query: Given a linear function f, compute a point p in S maximizing the value $f(p)$.

A typical approach in computational geometry is to preprocess S to have a data structure to attain a fast query time algorithm. For a set of points in a plane (i.e., $d = 2$), Voronoi diagrams, farthest Voronoi diagrams, and convex hull are efficient data structures for the purpose. Unfortunately, these methods are inefficient in high dimensional cases (for example, $d \geq 4$). Methods based on *simplex range searching* works for a general constant dimensional case, although (unless we use huge data space and preprocessing) the query time is $O(n^{1-t(d)})$ where $t(d)$ approaches to zero if d becomes large. Note that such a query can be done in $O(\log n)$ time in the PRAM model with n processors.

Lemma 2.1. *Each of nearest-neighbor query, farthest-neighbor query, and extremal query can be done in* $O(\sqrt{n})$ *time in the quantum model, where the success probability is at least* $1/2$.

Proof. As we have seen in the previous section, maximum element can be found in $O(\sqrt{n})$ time in the quantum model, and each of the geometric queries is basically a maximum-finding procedure. For example, for the nearest neighbor query, each point of S compute the distance from q in parallel to form a state vector, and we apply a quantum maximum-finding algorithm. □

Corollary 2.2. *Each of nearest-neighbor query, farthest-neighbor query, and extremal query can be done in* $O(\sqrt{n}\log n)$ *time in the quantum model, where the success probability is at least* $1 - 1/n^c$ *for any constant c.*

Remark: We could apply a better analysis given in [7], although we do not apply it in this paper for simplicity.

3. Lowest Point in the Upper Envelope

It is known [1, 25] that several geometric problems can be considered as parametric versions of geometric query problems, and hence efficiently solved if the queries can be done efficiently. As a show-case problem of designing quantum algorithms in geometric data processing, we consider the problem of computing the lowest point in an *upper envelope* in an arrangement of hyperplanes, which can be considered as a *parametric maximum finding problem.*

Given a set $\mathcal{H}$ of n hyperplanes in the d-dimensional space $\mathbb{R}^d$ where the d-th coordinate is denoted by y, consider the locus $\mathcal{U}(\mathcal{H})$ of the highest (with respect to y-value) halfplane. In other words, if $y = g_i(\boldsymbol{x}) = g_i(x_1, x_2, \dots x_{d-1})$ is the equation defining the i-th hyperplane H_i of $\mathcal{H}$, $\mathcal{U}(\mathcal{H})$ is defined by $y = \max_{1 \leq i \leq n} g_i(\boldsymbol{x})$.

Lowest point in the upper envelope of an arrangement of hyperplanes: Given a set $\mathcal{H}$ of n hyperplanes compute the point $\boldsymbol{p}$ in the upper envelope $\mathcal{U}(\mathcal{H})$ minimizing the y-coordinate value.

Since an upper envelope is a convex surface, the problem is a minimization problem of a piecewise linear convex function. The problem is a special LP problem, which finds the point minimizing y satisfying $y \geq g_i(\boldsymbol{x})$ for $i = 1, 2, \dots, n$.

3.1 Two-dimensional problem

We first consider two-dimensional case to get intuition. The set $\mathcal{H} = \{H_1, H_2, \dots H_n\}$, where H_i is a line defined by $y = g_i(x)$. The upper envelope $\mathcal{U}(\mathcal{H})$ is the trajectory of the function

$$G(x) = \max_{1 \leq i \leq n} g_i(x).$$

Thus, if the solution point is $\boldsymbol{p} = (p_x, p_y)$, $p_y = G(p_x) = \min_{-\infty \leq x \leq \infty} G(x)$. For simplicity, we assume that $\mathcal{H}$ has both negative-slope lines and positive-slope lines, and also it has no line horizontal or vertical to the x-axis. Hence, p_x is neither ∞ nor $-\infty$.

For a given value a, $G(a)$ can be computed as the maximum of n values $g_i(a)$ $(i = 1, 2, \dots, n)$. Hence, if we consider x as a parameter, the problem of computing $\boldsymbol{p}$ is a minimax parametric optimization problem [25].

Hence, we can apply a simple binary searching strategy: We consider an interval $I = [s(I), t(I)]$ of x-values containing p_x. We also have the lines $l(s(I))$ and $l(t(I))$ that contain points $(s(I), G(s(I)))$ and $(t(I), G(t(I)))$, respectively. The line $l(s(I))$ must have a negative slope and $l(t(I))$ have a positive slope.

We then compute $G(m(I))$ for the mid point $m(I)$ of I. If $G(m(I)$ is on a line $l(m(I))$ with a positive (resp. negative) slope, $p_x < m(I)$ (resp. $p_y > m(I)$). Let us assume that the line has a positive slope. Then, we compute the intersection point $\boldsymbol{q} = (q_x, q_y)$ of $l(s(I))$ and $l(t(I))$, and check whether q is on the upper envelope or not by computing $G(q_x)$. If $\boldsymbol{q}$ is on the upper envelope, it is the solution. Otherwise, we replace I with $(s(t), m(t))$, and recursively continue the operation. If the coefficients of lines are represented by quotient numbers of integers in the range $[-\Gamma, \Gamma]$ (we call Γ the *precision* of the input), the binary search stops in $O(\log \Gamma)$ steps, where each step corresponding to computation of a value on the upper envelope.

In the quantum model, we can compute $G(a)$ for a given a in $O(\sqrt{n})$ time with a probability larger than $1/2$ by using the maximum finding algorithm (Corollary 2.1). Since we do not want to fail in the decision of the binary search procedure, we spend $O(\sqrt{n}\log n)$ time to reduce the failure probability to n^{-c} of a suitable large constant c.

Hence, we can compute the lowest point in the upper envelope (thus, in the intersection of upper-halfplanes) in $O(\sqrt{n}\log n \log \Gamma)$ time. The above algorithm is weakly polynomial (dependent on precision of the input). Theoretically, we want to have a strongly-polynomial efficient algorithm. There are several techniques to transform a weakly-polynomial algorithm into a strongly-polynomial: Parametric searching [20] is a famous general method, however it does not seem to be adequate for designing quantum algorithms because it needs help of an algorithm with a parallel structure with small work, while most of quantum algorithms have large amount of total work. Another popular method for linear programming type problems is the *prune-and-search* method [12, 17], and its randomized version can be applied as follows:

Theorem 3.1. *The lowest point in the intersection of n upper-halfplanes can be computed in $O(\sqrt{n}\log^2 n)$ time with a probability $1 - n^{-c}$ for any constant c in the quantum model.*

Proof. Without loss of generality, we assume that no three lines intersect at a point. Let $\boldsymbol{p} = (p_x, p_y)$ be the solution point. We randomly pick a line ℓ from $\mathcal{H}$, and consider (implicitly) the intersection points on ℓ with other lines. We can binary search on the set of intersection points in the quantum model to find two consecutive intersection points $\boldsymbol{q} = (q_x, q_y))$ and $\boldsymbol{q}' = (q'_x, q'_y))$ such that $q_x \leq p_x \leq q'_x$. This can be done in $O(n^{1/2}\log n)$ time by using the quantum binary searching [18] $O(\log n)$ times, each of which takes $O(n^{1/2})$ time. Now, we recognize the set S_1 of lines above (or through) $\boldsymbol{q}$. The expected number of lines in S_1 is $n/2$, and since $\boldsymbol{q}$ and $\boldsymbol{q}'$ are consecutive on ℓ, $\boldsymbol{p}$ is the minimum point in the upper envelope of S_1. We amplify the lines in S_1, and apply the same procedure to S_1 until the size of the set of lines becomes constant. This is a prune-and-search strategy. In the RAM model, the prune and search strategy shrinks the input size and hence attains $O(t(n))$ time complexity if it takes $t(n)$ time for pruning $n/2$ elements; however, in the quantum model, we cannot explicitly prune away elements, and we can only attain $O(t(n)\log n)$ time complexity by using the fact that the problem size is reduced to a constant in $O(\log n)$ iterations. Thus, the time complexity is $O(n^{1/2}\log^2 n)$. □

3.2 Higher dimensional case

Next, we consider the case where the number of dimensions is more than two. It is easy to design an $O(\sqrt{n}\log n \log^{d-1}\Gamma)$ time algorithm for computing the lowest point in the intersection of n upper-halfspaces, since the binary searching for the first coordinate works, where the decision is given by solving the $(d-1)$-dimensional problem. In precise, suppose that x_1 is the first coordinate, y is the last coordinate, and $x_2, \ldots, x_{d-1}$ are other coordinates. For a given value a of x_1, we solve the $(d-1)$ dimensional problem of computing the lowest point in the intersection of the upper envelope and the vertical hyperplane $H(a) : x_1 = a$. The solution lies on a line (intersection of $(d-1)$ hyperplanes in $\mathcal{H}$), and we can decide in which side of $H(a)$ the solution point $\boldsymbol{p}$ lies by computing the slope of the line.

In order to make the algorithm strongly-polynomial, we need to use a multidimensional searching technique in an arrangement of hyperplanes instead of binary searching on a line.

3.2.1 Multidimensional searching in an arrangement

Given a set $\mathcal{H}$ of n hyperplanes in the d-dimensional space, consider the arrangement $A(\mathcal{H})$ of the hyperplanes. We are given a point $\boldsymbol{p}$ in the space, and assume that we can detect whether $\boldsymbol{p} \in H$, $\boldsymbol{p} \in H^+$, or $\boldsymbol{p} \in H^-$ for any given hyperplane H in $O(t)$ time (H need not be in $\mathcal{H}$). Multidimensional searching is to find a (possibly lower dimensional) simplex $\Delta_{\boldsymbol{p}}$ which contains $\boldsymbol{p}$ and is contained in the same face as $\boldsymbol{p}$ in the arrangement. In one-dimensional case, this is a binary searching in a given set of n real values.

We first select r hyperplanes randomly from $\mathcal{H}$, and triangulate it; in other words, we construct a *cutting of the arrangement* of a constant size. We can find a triangle Δ in the cutting containing p in $O(t)$ time (in each of PRAM and the quantum model). Now, we randomly choose r hyperplanes intersecting Δ by using the quantum model in $O(\sqrt{n/m})$ time with a high probability, and recursively search in Δ. With high probability, $m < cn/r$ for a constant c independent of n and r, and we take $r > 2c$. Thus, we can find a point in the same face as p in $O(\sqrt{n} + t\log n)$ time with a high probability.

3.2.2 Algorithm for a higher dimensional case

Theorem 3.2. *The lowest point in the upper envelope of an d-dimensional arrangement can be computed in $O(n^{1/2}\log^{2d-1} n)$ time with high probability in the quantum model.*

Proof. We only give an outline of the algorithm. Let $\boldsymbol{p} = (p_1, p_2, \ldots, p_{d-1}, p_y)$ be the solution point (i.e., minimizing the y-coordinate value). We randomly select a hyperplane F in $\mathcal{H}$, and consider the arrangement $A(\mathcal{H}) \cap F$ obtained by restricting the arrangement $A(\mathcal{H})$ on F. Let $\boldsymbol{q} = Proj(\boldsymbol{p}, F)$ be the point on F whose first $d-1$ coordinate values are the same as those of $\boldsymbol{p}$. We apply the multidimensional searching in the arrangement $A(\mathcal{H}) \cap F)$ to find a simplex $\Delta_{\boldsymbol{q}}$ containing $\boldsymbol{q}$. Here, we can decide in which side of a hyperplane H in $A(\mathcal{H}) \cap F$ the point $\boldsymbol{q}$ is located by solving the $(d-1)$-dimensional problem on the hyperplane containing H and parallel to the y-axis in the original space. Thus, $t = T(d-1)$, where $T(d-1)$ is the time for solving the $(d-1)$-dimensional problem.

The set of hyperplanes above $\Delta_{\boldsymbol{q}}$ must contain the optimal solution, and the expected size of the set is approximately

$n/2$. Hence, the size of the problem is reduced to half in $O(T(d-1)\log n)$ time, and reduce to a constant in $O(T(d-1)\log^2 n)$ time. Thus, $T(d) = O(\sqrt{n}\log^{2d-1} n)$. □

4. Geometric Optimization Problems

4.1 Minimum enclosing ball

Given a set S of n points, its minimum enclosing ball is the minimum radius ball containing all the points of S. We assume that the number d of dimensions of the space is a constant independent of n. The problem of computing the minimum enclosing ball is an *LP-type* problem, and can be solved in $O(n)$ time, by using a randomized incremental algorithm [22]. A natural question is whether we can solve it in sublinear time in the quantum model. We could modify the incremental algorithm into a quantum algorithm that seems to be efficient; however, the authors have not yet succeeded to analyze it theoretically.

Instead, we regard the minimum enclose ball problem as a minimization problem of the parametric farthest-neighbor-query problem [25]; in other words, it is computation of the lowest point in the upper envelope of surfaces defined from the distance functions from points. For each point $\boldsymbol{u} = (u_1, u_2, \ldots u_d) \in S$, we consider a function $g_{\boldsymbol{u}}$ on R^d defined by $g_{\boldsymbol{u}}(x) = \sum_{1\le i\le d}(u_i - x_i)^2$ for $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$. Let us consider the upper envelope $\mathcal{U}$ of the surfaces $y = g_{\boldsymbol{u}}(x)$ for all the points $\boldsymbol{u} \in S$ in the $(d+1)$-dimensional space. Then, the following is easy to see:

Lemma 4.2. *The surface $\mathcal{U}$ is a convex surface, and if its lowest point is $\boldsymbol{p} = (p_1, p_2, \ldots, p_d, p_y)$, the point $Proj(\boldsymbol{p}) = Proj(\boldsymbol{p}, \mathbb{R}^d) = (p_1, p_2, \ldots, p_d)$ is the center of the minimum enclosing ball.*

Moreover, if we consider the function $\tilde{g_{\boldsymbol{u}}}(x) = g_{\boldsymbol{u}}(x) - \sum_{1\le i\le d} x_i^2$, it is a linear function, and its upper envelope $\tilde{\mathcal{U}}$ is a piecewise convex function. Although the lowest point in $\tilde{\mathcal{U}}$ does not correspond to the lowest point $\boldsymbol{p}$ in $\mathcal{U}$, $Proj(\boldsymbol{p})$ becomes a projection image of a vertex of $\tilde{\mathcal{U}}$. Note that a projection image of a vertex of $\tilde{\mathcal{U}}$ is called a farthest-neighbor Voronoi vertex in the literature. First, we give a weakly-polynomial sublinear time algorithm.

Proposition 4.1. *Minimum enclosing ball can be computed in $O(\sqrt{n}\min\{\log^d \Gamma \log n, \log^{2d+1} n\})$ time with a high probability in the quantum model, where Γ is the precision of the input.*

Proof. We first apply a binary searching paradigm where we utilize convexity of $\mathcal{U}$. Consider a vertical hyperplane $H : x_1 = a$ for a given a. We compute the lowest point in $H \cap \mathcal{U}$ by solving the $(d-1)$-dimensional problem. From the solution, we can decide in which side of H the point $\boldsymbol{p}$ is located similarly to the argument in the previous section. Hence, we have a recursion that $T(d) = \log\Gamma T(d-1)$ for the time complexity $T(d)$ for solving the d-dimensional problem. For the case where $d = 0$, the problem is simply computing the maximum of n real values, and hence can be solved in $O(\sqrt{n}\log n)$ time with a probability $1 - n^{-c}$ for any constant c. Hence, we obtain $T(d) = O(\sqrt{n}\log^d \Gamma \log n)$.

We apply the prune-and-search strategy to design a strongly-polynomial sublinear time algorithm. Here, we utilize the arrangement of hyperplanes defined by $y = \tilde{g_{\boldsymbol{u}}}(x)$ $(\boldsymbol{u} \in S)$. We pick randomly a hyperplane F from it, and apply multidimensional searching for the point obtained by projecting $\boldsymbol{p}$ onto F along the y-direction, and then prune away hyperplanes (and also associated hypersurfaces in $\mathcal{U}$) below F at $Proj(\boldsymbol{p})$. Thus, we can applying similar analysis as the previous section to obtain the $O(\sqrt{n}\log^{2d+1} n)$ time complexity. □

4.2 Common-intersection emptiness and LP

Consider a set of n convex objects in $\mathbb{R}^d$, and we want to detect whether the common intersection of all these objects is empty or not.

The easiest case is that we have a set of n intervals in $\mathbb{R}$. For this case, if we compute the maximum x_1 of the left endpoints of intervals and the minimum x_2 of the right endpoints of interval, the intersection of the intervals is nonempty if and only if $x_1 \le x_2$. Hence, the problem is reduced to the maximum finding problem, and solved in $O(\sqrt{n})$ time in the quantum model. A similar argument holds for a set of axis parallel rectangles in $\mathbb{R}^d$.

A little more nontrivial case is where we have a set of convex polytopes consisting of n facets in total. This includes intersection detection of two convex polytopes in d-dimensional space as a special case. Here, the problem is just the feasibility problem of the linear programming problem associated with the n linear inequalities defining facets.

Theorem 4.1. *A linear programming problem with n linear constraints in the d-dimensional space can be solved in $O(\sqrt{n}\log^{2d-1} n)$ time with a high probability in the quantum model. In particular, the feasibility problem of a system of n linear inequalities can be solved in $O(\sqrt{n}\log^{2d-1} n)$ time.*

Proof. Without loss of generality, we assume that the problem is minimizing the y-coordinate value under a constraint given by n linear inequalities. We consider the halfspaces associated with the linear inequalities, and classify them into upper-halfspaces and lower-halfspaces with respect to the d-th coordinate y. Indeed, $y \ge f_i(\boldsymbol{x})$ $(i = 1, 2, \ldots, m)$ and

$y \leq f_i(\boldsymbol{x})$ $(i = m+1, m+2, \ldots, n)$ are corresponding inequalities. Let $\mathcal{U}$ and $\mathcal{L}$ be the upper envelope of the hyperplanes defining upper-halfspaces and the lower envelope of the hyperplanes defining lower-halfspaces, respectively. We first check the feasibility in our algorithm. If the system is feasible, we keep a point $\boldsymbol{u}$ in the feasible region. we find the lowest point $\boldsymbol{p}$ of the upper envelope of the upper halfspaces in S under the condition that the point is in the feasible region. The only difference from the solution in Sect. 3 is that if we solve a $(d-1)$-dimensional problem in a vertical hyperplane and find out that the problem is infeasible in the hyperplane, we decide $\boldsymbol{p}$ is on the same side of the hyperplane as $\boldsymbol{u}$.

Thus, it suffice to show how to check feasibility. Let us consider $G^+(\boldsymbol{x}) = \max_{1 \leq i \leq m} f_i(\boldsymbol{x})$, $G^-(\boldsymbol{x}) = \min_{m+1 \leq i \leq n} f_i(\boldsymbol{x})$, and $G(\boldsymbol{x}) = G^+(\boldsymbol{x}) - G^-(\boldsymbol{x})$. Clearly, $G(\boldsymbol{x})$ is a convex function. The system is feasible if and only if there is a point $\boldsymbol{p}$ in $\mathcal{L}$ above $\mathcal{U}$, in other words, satisfying that $G(\boldsymbol{p}) \leq 0$. Thus, this is a parametric optimization problem, and we can apply the multidimensional prune-and-search strategy. If we randomly select a hyperplane, and execute the multidimensional searching in the arrangement on it. If the hyperplane defines a upper-halfspace, half of the upper-halfspaces are eliminated (in expectation) implicitly; otherwise, half of the lower-halfspaces are eliminated. The expected number of eliminated halfspaces is $m^2/2n + (n-m)^2/2n \geq n/4$. Hence, the prune-and-search works, and we obtain the proposition. □

Next, we consider the case where we have a set of general convex objects $Q_1, Q_2, \ldots, Q_n$, and would like to detect whether they have a common intersection point or not. We take a representative point $\boldsymbol{q}_i$ in each Q_i. We assume that the objects are defined by constant degree algebraic equations. For any point $\boldsymbol{x}$ in the space, consider the half-line ℓ_i emanated from $\boldsymbol{q}_i$ through $\boldsymbol{x}$, and define a *separation distance* $d_{\mathrm{sep}}(\boldsymbol{x}, Q_i) = d(\boldsymbol{x}, \boldsymbol{q}_i)/d(\boldsymbol{q}_i, \boldsymbol{r}_i(\boldsymbol{x}))$, where $d()$ is the Euclidean distance and $\boldsymbol{r}_i(\boldsymbol{x})$ is the intersection point of ℓ_i and the boundary of Q_i. If ℓ_i does not intersect the boundary (this may occur only if Q_i is an unbounded region), we set $d_{\mathrm{sep}}(\boldsymbol{x}, Q_i) = 0$. The intersection $\cup_{i=1}^n Q_i \neq \emptyset$ if and only if there exists a point $\boldsymbol{x}$ satisfying that $\max_{1 \leq i \leq n} d_{\mathrm{sep}}(\boldsymbol{x}, Q_i) \leq 1$.

Under the assumption that we can compute $d_{\mathrm{sep}}(\boldsymbol{x}, Q_i)$ in $O(1)$ time for each i, and the comparison of $d_{\mathrm{sep}}(\boldsymbol{x}, Q_i)$ and $d_{\mathrm{sep}}(\boldsymbol{x}, Q_j)$ can be done in $O(1)$ time for each i and j, we have the following:

Theorem 4.2. *We can detect whether* $\cup_{i=1}^n Q_i = \emptyset$ *or not in* $O(\sqrt{n} \log^d \Gamma \log n)$ *time with a high probability in the quantum model.*

Proof. The function $\max_{1 \leq i \leq n} d_{\mathrm{sep}}(\boldsymbol{x}, Q_i)$ is a convex function, and for a given $\boldsymbol{p}$, $\max_{1 \leq i \leq n} d_{\mathrm{sep}}(\boldsymbol{p}, Q_i)$ can be computed in $O(\sqrt{n} \log n)$ time with a probability $1 - n^{-c}$. Hence, we can solve the problem as a parametric optimization problem to have the time complexity. □

We finally consider the case where each Q_i $(i = 1, 2, \ldots, m)$ is given as the convex hull of a point set P_i. We assume that the total of the numbers of the points in P_i $(i = 1, 2, \ldots, m)$ is n. In this case, if we can compute the facet representation of Q_i, we can apply the method given above; however, it needs $\Omega(n)$ time even in the two-dimensional case (even in the quantum model), and much more expensive if $d \geq 4$. Instead, we solve the problem without constructing convex hulls explicitly as follows:

Theorem 4.3. *We can detect whether* $\cup_{i=1}^m Conv(P_i) = \emptyset$ *or not in* $O(\sqrt{n} \log^{2d} \Gamma \log n)$ *time with a high probability in the quantum model. Here,* $n = \sum_{i=1}^m |P_i|$.

Proof. As the representative point of $Q_i = Conv(P_i)$, we take an arbitrary point $\boldsymbol{q}_i \in P_i$ for each $i = 1, 2, \ldots, m$. For a given point $\boldsymbol{x}$ in the space, it suffices to compute the distance $d_{\mathrm{sep}}(\boldsymbol{x}, Q_i)$. For the purpose, we need to find the point $\boldsymbol{r}_i(\boldsymbol{x})$ (the intersection point of the halfline emanating from $\boldsymbol{q}_i$ through $\boldsymbol{x}$ on the boundary of the convex hull). This is indeed the dual of the LP problem. and thus we can obtain the theorem. □

4.3 Convex hull computation

Consider the problem of computing the convex hull $Conv(S)$ of a set S of n points in the d-dimensional space. This is a fundamental problem, and an $O(n^{\lfloor d/2 \rfloor})$ time algorithm, which is optimal in the worst case, is known. Our interest is on the output sensitive computation. See [21] for a survey of convex hull algorithms.

Let M be the number of faces of $Conv(S)$. Cole et. al. gave a method based on *hand probing* oracle, where a hand probing is indeed an extremal query. In the algorithm, we can compute the convex hull with $O(ML(n))$ time, where $L(n)$ is the time for the extremal query. Moreover, Seidel gave an $O(n^2 + M \log n)$ time algorithm, and Matousek improved its first term to $n^{2-2/\lfloor (d+2)/2 \rfloor}$. One important operation in Seidel's algorithm is to detect whether a point S is a vertex of $conv(S)$ or not. This is the dual of the linear programming query, which, as we have seen, can be performed $O(\sqrt{n} \log^{2d-1} n)$ time.

Hence, we have the following:

Theorem 4.4. *Conv(S) can be computed in* $O(\min\{M\sqrt{n} \log n, n^{1.5} \log^{2d-1} n + M \log n\})$ *time in the quantum model* *with a high probability.*

If M is small, the time complexity may become sublinear. For example, if the point set is uniformly distributed in a (two-dimensional) disk, the expected number of M is $O(n^{1/3})$ [5], and hence the expected time complexity becomes $O(n^{5/6}\log n)$.

As the dual problem of the convex hull, we have the following:

Corollary 4.3. *Let $\mathcal{A}$ be an arrangement of n hyperplanes in the d-dimensional space. Given a point p, we can construct the cell of $\mathcal{A}$ containing p in $O(\min\{M\sqrt{n}\log n, n^{1.5}\log^{2d-1} n + M\log n\})$ time in the quantum model with a high probability, where M is the number of faces in the cell.*

4.4 Separation query and convex programming

Let S be a set of n regions in the d-dimensional space, and consider the following *separation query*: Given a query point $\boldsymbol{p}$, if $\boldsymbol{p}$ is in the intersection of the region of S, report that fact, otherwise, report a region that does not contain $\boldsymbol{p}$. Note that if each region is a halfspace, this can be considered as the dual of the extremal query (with a slight modification).

Lemma 4.3. *If it can be detected whether **p** is contained in each region of S in $O(1)$ time, the separation query can be done in $O(\sqrt{n}\log n)$ time with a high probability.*

The ellipsoid algorithm for solving a convex programming problem is based on the separation queries. Thus, we can speed up the ellipsoid algorithm by using the quantum model. In particular, for the feasibility check problem of the constant dimensional linear programming problem, the number of calls of separation queries is $O(\log\Gamma)$, if each entry of the constraint matrix is represented as a quotient number of two integers whose absolute values are bounded by Γ. We can solve an (optimization version of) linear programming problem by calling the feasibility check $O(\log\Gamma)$ time. Hence, we have the following:

Proposition 4.2. *A constant-dimensional linear programming problem can be solved in $O(\sqrt{n}\log n\log^2\Gamma)$ time in the quantum model.*

The above time complexity is better than Theorem 4.1 if d is large and Γ is small, although we do not know how to convert it into a strongly polynomial one; Indeed, it is a big (and widely considered to be difficult) open problem how to convert the ellipsoid method into strongly polynomial in the usual RAM model computation.

5. Intersection Detection and Proximity Problems

We can solve some intersection problems and proximity problems by using another type of quantum algorithms; In one-dimensional case, given a set of n points on a line, the problem to detect a pair of ‘‘intersecting’’ points (points at the same position) is called the *element distinctness problem*. Buhrman et al. [8] gave a one-sided bounded error quantum algorithm solving the element distinctness problem in $O(n^{3/4}\log n)$ time. The algorithm is based on a quantum version of the *birthday trick*, which is a basic strategy in randomized algorithms. We can apply the same strategy to some geometric problems. Because of space limitation, we only list results, and details will be given in a separate paper [19].

Theorem 5.1 (Segment intersection detection). *Given a set S of n line segments in the plane, we can detect whether S has a pair of intersecting segments in $\tilde{O}(n^{7/8})$ time in the quantum model, where $\tilde{O}$ is the Big-O notation ignoring polylogarithmic factors.*

Theorem 5.2 (Nearest pair computation). *Given a set S of n points in the $\mathbb{R}^d$, we can compute a pair of points of S with the minimum distance in $O(n^{3/4}\log^2 n)$ time for $d = 2$, and in $\tilde{O}(n^{1-1/4\lceil d/2\rceil})$ time for $d \geq 3$.*

Theorem 5.3 (Hausdorff distance computation). *Given two convex polygonal bodies P and Q in the plane, we can compute the Hausdorff distance $d(P,Q)$ in $O(n^{7/8}\log^2 n)$ time. Here, the Hausdorff distance is defined by $d(P,Q) = \inf\{\rho | P \subset Q + \rho B, Q \subset P + \rho B\}$, where $+$ is the Minkowski sum and B is the unit disk.*

6. Concluding Remarks

We have considered two types of problems: (1) geometric optimization problems considered as a minimization of convex objective functions in parameters, and (2) geometric version of the element distinctness problem. The solutions are depending on Grover's data search algorithm; however, we can notice difference between solutions of problems of (1) and those of (2). In precise, the solutions for problems of (2) needs higher space complexity and mechanism to construct data structures such as Voronoi diagrams (details will be given in [19]), whereas those of (1) only need Grover's algorithm. It is desired to develop many useful subroutines to attack geometric problems by the quantum computing model. On the other side, there are several geometric problems with linear lower bounds; for example, counting range searching query cannot be done in sublinear time without preprocessing in the quantum model (we can apply Ambanis's lower bound technique [3]).

Acknowledgement

The authors gratefully acknowledge to Kazuo Iwama, who informed the reference [8]. Also they thank Takashi Mihara and Peter Høyer for valuable comments on an initial version of the paper.

REFERENCES

[1] Agarwal, P., and Sharir, M., (1997), Efficient Algorithms for Geometric Optimization, Research Report, Duke University, 2708-0129.

[2] Ambanis, A., (1999), A better bound for quantum algorithms searching an ordered list, Proc. 40th FOCS, 352–357.

[3] Ambanis, A., (2000), Quantum lower bounds by quantum arguments, Proc. 32nd STOC, 636–643.

[4] Bernstein, E., and Vaziriani, U., (1993), Quantum complexity theory, Proc. 25th STOC, 11–20.

[5] Borgwart, K., (1987), The Simplex Method — A Probabilistic Approach, Springer-Verlag, Heidelberg.

[6] Boyer, M., Brassard, G., Høyer, P., and Tapp, A., (1998), Tight bounds on quantum searching, Fortschr. Phys.

[7] Buhrman, H., Cleve, R., de Wolf, R., and Zalka, C., (1999), Bounds for Small-Error and Zero-Error Quantum Algorithms, Proc. 40th FOCS, 358–368.

[8] Buhrman, H., Dürr, C., Heiligman, M., Høyer, P., Magniez, F., Santha, M., and de Wolf, R., Quantum Algorithms for Element Distinctness, quant-ph/0007016, to appear in Proc. IEEE Complexity 2001.

[9] Clarkson, K., (1987), New applications of random sampling in computational geometry, Discrete Comput. Geom., 2: 195–222.

[10] Clarkson, K. and Shor, P., (1989), Applications of random sampling in computational geometry II, Discrete Comput. Geom., 4: 387–421.

[11] Deutcsh D., and Jozsa, R., (1992), Rapid solution of problems by quantum computation, Proc. R. Soc. London, Ser. A, 400: 73–90.

[12] Deyer, M. and Meggido, N., (1997), Linear programming in low dimensions, Section 38 of Handbook of Discrete and Computational Geometry, CRC Press, 699–710.

[13] Dürr, C. and Høyer, P., A Quantum Algorithm for Finding the Minimum, quant-ph/9607014.

[14] Grover L., (1996), A faster quantum mechanical algorithm for database search, Proc. 28th STOC, 212–219.

[15] Grover, L., (1999), Quantum search on structured problems, Chaos, Solution, & Frctals, 10: 1695–1705.

[16] Grover, L., (2000), Rapid sampling through quantum computing, Proc. 32nd STOC, 618–626.

[17] Megiddo, N., (1984), Linear programming in linear time when dimension is small, SIAM J. Comput., 12: 114–127.

[18] Mihara, T., and Sung, S.-C., (1999), On a quantum algorithm of finding specific members in a database, IEICE Tech. Rep., QIT99-21, 51–56.

[19] Sadakane K., Sugawara, N., and Tokuyama T., Geometric intersection problems in quantum model, working paper.

[20] Salowe, J., (1997), Parametric Search, Section 37 of Handbook of Discrete and Computational Geometry, CRC Press, 683–695.

[21] Seidel, R., (1997), Convex Hull Computations, Section 19 of Handbook of Discrete and Computational Geometry, CRC Press, 361–376.

[22] Sharir, M. and Welzl, E., (1992), A combinatorial bound for linear programming and related problems, Proc. 9th STACS, LNCS 577, 569–579.

[23] Shor, P., (1997), Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput., 26: 1484–1509.

[24] Smith, W., (2001), Sublinear time quantum algorithms in computational geomety and related areas, working paper, NEC Research.

[25] Tokuyama, T., (2001), Minimax parametric optimization problems and multi-dimensional parametric searching, Proc. 33rd STOC, to appear.