Fixed-Dimensional Linear Programming Queries Made Easy*

Timothy M. Chan†

Abstract

We derive two results from Clarkson's randomized algorithm for linear programming in a fixed dimension d. The first is a simple general method that reduces the problem of answering linear programming queries to the problem of answering halfspace range queries. For example, this yields a randomized data structure with $O(n)$ space and $O(n^{1-1/\lfloor d/2 \rfloor} 2^{O(\log^* n)})$ query time for linear programming on n halfspaces ($d > 3$). The second result is a simpler proof of the following: a sequence of q linear programming queries on n halfspaces can be answered in $O(n \log q)$ time, if $q \leq n^{\alpha_d}$ for a certain constant $\alpha_d > 0$. Unlike previous methods, our algorithms do not require parametric searching.

1 Introduction

One of the major discoveries in computational geometry is that fixed-dimensional linear programming can be solved in linear time [Meg84]. It was observed that the introduction of *randomization* leads to considerably simpler solutions [Sei91, Cla95]. The goal of this paper is to extend this observation for the problem of answering fixed-dimensional linear programming queries.

Let H be a set of n (closed) halfspaces in $\mathbb{R}^d$. A *j-dimensional linear programming (LP) query on H* ($0 \leq j \leq d$) is the following:

> Given a linear function $w : \mathbb{R}^d \to \mathbb{R}$ and a j-flat $f \subseteq \mathbb{R}^d$, find a point in $f \cap (\bigcap H)$ that minimizes w.

This definition includes as special cases *membership queries* ($j = 0$; decide if a query point lies in $\bigcap H$), and *ray shooting queries* ($j = 1$; compute the intersection of a query ray with $\bigcap H$). The problem of answering LP queries on a set of halfspaces has numerous applications, ranging from the enumeration of extreme points [Mat93a, Cha95] to the construction of levels in hyperplane arrangements [AM95, Cha95].

For $d = 2$ or 3, it is well known that an LP query can be answered in $O(\log n)$ time if the halfspaces are preprocessed in $O(n \log n)$ time and stored in a data structure of $O(n)$ size [DK90]. Here, we are mainly interested in the case when d is a fixed constant greater than 3. Matoušek [Mat93a] had studied the problem and gave data structures achieving the performance shown in Table 1 (see [MS93] for a small improvement to the third structure). His approach is based on a j-wise application of the *parametric search technique* [Meg83] in d-space, that inductively reduces j-dimensional LP queries to $(j-1)$-dimensional LP queries. His query algorithm thus resembles the LP algorithm by Megiddo [Meg84].

In [Mat94], Matoušek asked whether there is a simpler approach to LP queries, perhaps using randomization. In this paper, we provide a method that answers this question: our query algorithm is based on the randomized LP algorithm by Clarkson [Cla95] and is thus simple and easy to describe. Specifically, we show that any solution to the *halfspace range reporting problem* can directly yield a solution to the LP problem. This transformation increases the preprocessing time and space of the halfspace range reporting data structures by only a constant factor. Moreover, the asymptotic query time remains unchanged with high probability, if it grows faster than some fractional power of n.

While the multidimensional parametric search technique of Matoušek [Mat93a] provides a fairly general reduction of the LP problem to the *halfspace emptiness problem*, our reduction of LP to halfspace range reporting is more general, as Matoušek's technique requires that the halfspace emptiness query algorithm satisfies a certain criterion and that it can be parallelized in a polylogarithmic number of steps.

The parametric search technique has another disad-

---

*A portion of this work was performed at the University of British Columbia. This research was supported by the Army Research Office under grant DAAH04-96-1-0013.

†Center for Geometric Computing, Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218

preprocessing time	space	query time
$n \log n$	n	$n^{1-1/\lfloor d/2 \rfloor} \log^{O(1)} n$
$m \log^{O(1)} n$	$m \log^{O(1)} n$	$\dfrac{n}{m^{1/\lfloor d/2 \rfloor}} \log^{2j+1} n$
$n^{\lfloor d/2 \rfloor} \log^{O(1)} n$	$n^{\lfloor d/2 \rfloor} \log^{O(1)} n$	$\log^{j+1} n$

Table 1: Known (static) data structures for linear programming queries [Mat93a]. The parameter m lies between n and $n^{\lfloor d/2 \rfloor}$ ($d > 3$).

vantage: the algorithms that it produces are complicated and difficult to implement. The only known methods we know of that avoid parametric searching are by Matoušek and Schwarzkopf [MS93] and are for the special case of ray shooting ($j = 1$) only (they actually impose a further restriction that the origins of the query rays lie in $\bigcap H$). However, they only obtain transformations of some *specific* range searching structures to ray shooting structures; with our technique, we can obtain a transformation of an *arbitrary* halfspace range reporting structure to a ray shooting structure.

Yet another disadvantage of parametric search is that each application typically adds an extra polylogarithmic factor to the running time. Thus, the j-wise application of parametric search in Matoušek's approach adds at least an $\Omega(\log^j n)$ factor to the query time. The technique of this paper allows us to remove these extra factors easily in many cases. For example, we obtain a data structure with $O(n^{1+\varepsilon})$ preprocessing time and $O(n)$ space, such that LP queries can be answered in $O(n^{1-1/\lfloor d/2 \rfloor} 2^{O(\log^* n)})$ time with high probability; the best linear-space structure from [Mat93a] has an $O(n^{1-1/\lfloor d/2 \rfloor} \log^{O(1)} n)$ query time. (We use ε to denote an arbitrarily small positive constant and $\log^*$ to denote the iterated logarithm.)

As a second application of Clarkson's LP algorithm, we prove that a sequence of q LP queries on H can be answered in deterministic $O(n \log q)$ time, if $q \leq n^\alpha$ for a certain constant $\alpha > 0$ that depends on d. The previous proof is from [Cha95] and achieves expected $O(n \log q)$ time by a randomized method; $O(n \log q)$-time deterministic methods are obtained only for the $d = 2$ case, the ray shooting case ($j = 1$), and the case when $q = \Omega(\log^\varepsilon n)$. The algorithms there basically involve grouping the halfspaces into subsets of a certain size and building a partition tree for each of these subsets. The method here does not use the grouping technique and is even simpler: we just construct one level of the partition tree. The specialization of this result to ray shooting is worth noting, as it can be used to construct convex hulls with small number of faces [Cha95]. The case $d = 3$ also has an application to the smallest k-enclosing circle problem [Mat95b].

In the next section, we discuss known data structures for the halfspace emptiness and halfspace range reporting problems (for more background information, see the survey by Matoušek [Mat94]). In Section 3, we present our randomized algorithm for LP queries. The deterministic algorithm for answering a sequence of $q \leq n^\alpha$ LP queries is given in Section 4. In the last section, we briefly discuss generalizations as well as the specialization to ray shooting.

2 Preliminaries on halfspace range queries

Let H be a set of n halfspaces in $\mathbb{R}^d$. The most basic queries one can ask about the polytope $\bigcap H$ are *membership queries*: given a point p, decide whether p belongs to $\bigcap H$. Since this membership problem is decomposable, we may assume that the halfspaces in H all contain $(0, \ldots, 0, \infty)$; we can treat the halfspaces containing $(0, \ldots, 0, -\infty)$ in a similar fashion. Using duality to transform H to a set of points $P \in \mathbb{R}^d$, a membership query reduces to the following: given a halfspace γ, decide whether $\gamma \cap P = \emptyset$. This is the so-called *halfspace emptiness problem.*

For the algorithm in the next section, membership queries alone do not suffice. We need the following kind of queries: given a point p, identify all k halfspaces in H that violate p (a point p *violates* a halfspace h if $p \notin h$). The dual problem corresponds to: given a halfspace γ, report all k points in $\gamma \cap P$. This is the *halfspace range reporting problem.*

It turns out that many of the known data structures for halfspace emptiness actually come from data structures for halfspace range reporting. The following was shown by Matoušek [Mat92b]:

Lemma 2.1 *Given an n-point set $P \subseteq \mathbb{R}^d$ and $n \leq m \leq n^{\lfloor d/2 \rfloor}$, there is a data structure with $O(m \log^{O(1)} n)$ preprocessing time and space, such that a halfspace emptiness query can be answered in*

$O((n/m^{1/\lfloor d/2\rfloor})\log n)$ *time and a halfspace range reporting query can be answered in* $O((n/m^{1/\lfloor d/2\rfloor})\log n + k)$ *time.*

For the linear-space case, there is a data structure [Mat92b] that can answer halfspace emptiness queries in $O(n^{1-1/\lfloor d/2\rfloor}2^{O(\log^* n)})$ time; it requires an $O(n^{1+\varepsilon})$ preprocessing time. Matoušek did not give a corresponding result for the halfspace range reporting problem; the closest result for the reporting problem is an $O(n\log\log n)$-space structure with $O(n^{1-1/\lfloor d/2\rfloor}\log^{O(1)} n + k)$ query time. By combining these two structures, one can obtain the following:

Lemma 2.2 *Given an* n*-point set* $P \subseteq \mathbb{R}^d$, *there is a data structure with* $O(n^{1+\varepsilon})$ *preprocessing time and* $O(n)$ *space, such that a halfspace range reporting query can be answered in* $O(n^{1-1/\lfloor d/2\rfloor}2^{O(\log^* n)} + kn^{\varepsilon})$ *time.*

Proof: This proof assumes that the reader is familiar with the *partition trees* in [Mat92b]. We build a constant number of levels of the partition tree corresponding to the $O(n\log\log n)$-space halfspace reporting structure, so that the number of points in each leaf falls below n^{ε}. Then we build the linear-space halfspace emptiness structure at each leaf. To answer a reporting query at the leaf, we first answer the emptiness query; if the answer is no, we carry out the query by examining all n^{ε} points in the leaf. □

We conclude this section by noting that there is a simple general method for converting any halfspace emptiness structure into a halfspace range reporting structure. This method uses the decomposability of the reporting problem and has been known earlier in a different context [CG86, AvKO93].

Observation 2.3 *Suppose that there is a data structure for the halfspace emptiness problem with* $P(n)$ *preprocessing time,* $S(n)$ *space, and* $Q(n)$ *query time, where* $n^{1-\varepsilon}/Q(n)$ *is a nondecreasing function. Then there is a data structure for the halfspace range reporting problem with* $O(P(n)\log n)$ *preprocessing time,* $O(S(n)\log n)$ *space, and* $O(kQ(n/k)\log(n/k))$ *query time* ($k > 0$).

Proof: Given n-point set P, partition P into two subsets P_1, P_2 of equal size. Our reporting structure for P consists of recursively defined reporting structures for P_1 and P_2, together with a halfspace emptiness structures for P. To answer a reporting query on P, we first answer the emptiness query on P; if the answer is no, we then recursively answer the reporting queries on P_1 and P_2. □

The above data structure can also be used for the problem of *halfspace emptiness with witness*: given a halfspace γ, report one point in $\gamma \cap P$ if $\gamma \cap P \neq \emptyset$. The query time we get is $O(Q(n)\log n)$; the logarithmic factor can be removed if $Q(n)/n^{\varepsilon}$ is nondecreasing. Strangely, this almost trivial method was not mentioned in any of the papers [AM93, Mat93a, MS93] discussing the witness problem.

3 A randomized algorithm for LP queries

We now present the main result of the paper. In what follows, a function is $\tilde{O}(f(n))$ if it is $O(f(n))$ with probability at least $1 - O(1/n)$.

Theorem 3.1 *Suppose that there is a data structure for halfspace range reporting queries with* $P(n)$ *preprocessing time,* $S(n)$ *space, and* $Q(n,k)$ *query time, where* $P(n)/n$, $S(n)/n$, *and* $Q(n,k)/n^{\varepsilon}$ *are all nondecreasing functions. Then there is a randomized data structure for LP queries with* $O(P(n))$ *preprocessing time and* $O(S(n))$ *space, such that the maximum query time is* $\tilde{O}(Q(n, C\log n))$ *for some constant* C.

Proof: We follow Clarkson's LP algorithm [Cla95] and use *random sampling*; the main difference is our choice of sample sizes. (Clarkson actually gave two different random sampling algorithms; one is recursive and one is iterative. We follow the former approach.)

Let H be a set of n halfspaces. We choose a random sample $R \subseteq H$ with size n/b, where b is a suitable constant. To build the data structure for H, we recursively build the data structure for R, together with a halfspace range reporting structure for the dual points of H, unless the size of H drops below a constant n_0. Clearly, the preprocessing time $\widehat{P}(n)$ and space $\widehat{S}(n)$ obey the recurrences

$$\widehat{P}(n) = \widehat{P}(n/b) + P(n) \quad \text{and} \quad \widehat{S}(n) = \widehat{S}(n/b) + S(n),$$

which imply that $\widehat{P}(n) = O(P(n))$ and $\widehat{S}(n) = O(S(n))$.

We now give the query answering algorithm. We only discuss d-dimensional LP queries, as these are the most "difficult" of the LP queries. For simplicity, we assume that the optimum to a query is unique and is defined by precisely d halfspaces (this general position assumption can be removed by perturbation arguments as in [Cla95]). Furthermore, we assume that a feasible solution exists (we can verify that the solution produced by the algorithm is correct by answering one membership query; if the answer is no, then the query is infeasible). The algorithm is recursive and has an extra set A as parameter; we can set $A = \emptyset$ initially.

Algorithm Query(H, w, A)

[Given linear function $w : \mathbb{R}^d \to \mathbb{R}$ and a set A of halfspaces in $\mathbb{R}^d$, return a point in $(\bigcap H) \cap (\bigcap A)$ that minimizes w.]

1. if $|H| \leq n_0$ then
2.   return optimum by examining all d-tuples of $H \cup A$
3. for $j = 1, \ldots, d+1$ do
4. $v_j \leftarrow$ Query$(R, w, A \cup V_1 \cup \ldots V_{j-1})$
5. $V_j \leftarrow \{h \in H : v_j \text{ violates } h\}$
6.   if $V_j = \emptyset$ then return v_j

Proof of Correctness. We follow the correctness proof in [Cla95]. Let H^* be the set of d halfspaces in $H \cup A$ that define the optimal vertex. Observe that if v_j is not the optimum, then v_j must violate some halfspace in $H^* \cap H$; so, V_j must contain a halfspace in H^*. If none of $\{v_1, \ldots, v_{d+1}\}$ is the optimum, then the disjointness of the V_j's would imply that $|H^*| \geq d+1$: a contradiction. Thus, v_j is the optimum for some j. Now, by choosing the smallest such j, we see that $V_j = \emptyset$ and line 6 of the algorithm correctly returns v_j.

Analysis of Query Time. Let $\widehat{Q}(n, a)$ denote the maximum running time of Query(H, w, A) with $|H| = n$ and $|A| = a$. Line 2 (the base case) can easily be done in $(n+a)^{O(1)}$ time. We observe that line 5 can be done by answering a halfspace range reporting query on H. The key to the analysis then lies in bounding the number k of halfspaces that the point v_j violates. With high probability, this number turns out to be small, as we know that $v_j \in \bigcap R$.

We say that a subset $R \subseteq H$ is an *ε-net* if any point in $\bigcap R$ violates at most εn of the halfspaces in H. According to a standard result on random sampling (e.g., see [HW87, Mul93]), a random sample $R \subseteq H$ of size r is a $((c \log t)/r)$-net with probability at least $1 - O(1/t^2)$, for any $t \geq r$ and a large enough constant c. Setting $r = n/b$ in our case, we have that any point in $\bigcap R$ violates at most $bc \log t$ of the halfspaces in H with probability $1 - O(1/t^2)$, if $t \geq n$. Thus, with probability at least $1 - O(1/t)$, we can ensure that this property holds for all of the $\Theta(\log n)$ random samples chosen during the preprocessing.

With probability at least $1 - O(1/t)$, we then have the following recurrence for $\widehat{Q}(n, a)$, where C is a constant:

$$\widehat{Q}(n,a) \leq \begin{cases} (n+a)^{O(1)} & \text{if } n \leq n_0 \\ (d+1)\,\widehat{Q}(n/b,\, a + C\log t) & \\ \qquad + O(Q(n, C\log t)) & \text{if } n > n_0 \end{cases}$$

By choosing $b \gg d$ to be a sufficiently large constant, we have $\widehat{Q}(n, 0) = O(Q(n, C \log t))$ with probability $1 - O(1/t)$. Setting $t = n$ yields the theorem. $\square$

From Lemma 2.1, we immediately get the following:

Corollary 3.2 *Let H be a set of n halfspaces in $\mathbb{R}^d$ and $n \leq m \leq n^{\lfloor d/2 \rfloor - \varepsilon}$ $(d > 3)$. There is a data structure with $O(m \log^{O(1)} n)$ preprocessing time and space, such that an LP query on H can be answered in $\tilde{O}((n/m^{1/\lfloor d/2 \rfloor}) \log n)$ time.*

Compared to the second structure in Table 1, this corollary improves the query time by logarithmic factors. However, one can get this improvement in the query time by "hiding" extra logarithmic factors in the $O(m \log^{O(1)} n)$ space bound. In the linear-space case though, we obtain an actual improvement in the query time, as compared to the first structure in Table 1, assuming an $O(n^{1+\varepsilon})$ preprocessing time:

Corollary 3.3 *Let H be a set of n halfspaces in $\mathbb{R}^d$ $(d > 3)$. There is a data structure with $O(n^{1+\varepsilon})$ preprocessing and $O(n)$ space, such that an LP query on H can be answered in $\tilde{O}(n^{1-1/\lfloor d/2 \rfloor} 2^{O(\log^* n)})$ time.*

Remarks:

1. We can make the query algorithm deterministic by applying derandomization methods on the computation of ε-nets [CM93, BCM93]. The space and query time remain the same; however, the cost of preprocessing becomes quite high (but polynomial), since we need to construct $(1/r)$-nets with r close to $n/\log n$. For certain specific halfspace range reporting structures, it may be possible to reduce this preprocessing cost.

2. The parametric search method by Matoušek [Mat93a] allows insertions and deletions of halfspaces. To make our randomized method dynamic, we can use the skip-list approach [Pug90, Mul93]. If the halfspace range reporting structure can be updated in $U(n)$ time, then it can be shown that the expected update time for our LP structure is $O(U(n))$.

3. With our method, we cannot get a structure with a polylogarithmic query time like the third structure in Table 1. Fortunately, Matoušek's parametric search technique is the easiest to apply for this large-space structure, since this case does not need a parallel version of the halfspace emptiness algorithm [Mat93a].

4 A deterministic algorithm for the few-query case

In this section, we discuss a special case in which there are only a few LP queries to be answered. In such a case, we need not spend too much time in the preprocessing. We show that if there are $q \leq n^\alpha$ queries for a certain constant $\alpha > 0$, then the total running time (including preprocessing) that is required to answer all of these queries is $O(n \log q)$.

To prove this, we can follow the approach of the previous section, using Clarkson's LP algorithm. We observe that in this special case, the solution can be simplified and can easily be made deterministic as well as dynamic (assuming fewer than q insertions/deletions).

As before, let H be the given set of n halfspaces in $\mathbb{R}^d$. We assume that these halfspaces all contain $\{(0, \ldots, 0, \infty)\}$, and we let $P \subseteq \mathbb{R}^d$ be the set of points dual to these halfspaces. To remove this assumption, we can keep track of two sets of dual points instead of one, and the algorithm below can be appropriately modified. Again, we assume general position and focus only on d-dimensional LP queries.

For the proof, we need the *Partition Theorem* of Matoušek [Mat92a]. A collection $\{(P_1, \Delta_1), \ldots, (P_m, \Delta_m)\}$ is said to form a *simplicial partition* of P if (i) the P_i's are disjoint subsets whose union is P, (ii) the Δ_i's are relatively open simplices, and (iii) $P_i \subseteq \Delta_i$ for each i. The number m is the *size* of the simplicial partition; the *class sizes* are the cardinality of the P_i's; the *crossing number* is the maximum number of simplices in $\{\Delta_i\}$ that a hyperplane can cross (a hyperplane h *crosses* a simplex Δ if $h \cap \Delta \neq \emptyset$ and $\Delta \not\subseteq h$).

Lemma 4.1 (Partition Theorem [Mat92a]) *Let $1 \leq r \leq n^\beta$ for a certain constant $\beta > 0$. In $O(n \log r)$ time, one can construct a simplicial partition of size $O(r)$, with class sizes between n/r and $2n/r$ and crossing number $O(r^{1-1/d})$.*

This theorem is the basis of many range searching data structures; for example, the linear-space structure from [Mat92b] for halfspace range reporting consists of a tree whose nodes are generated by repeated applications of a variant of the Partition Theorem. Here, we use only one application of the theorem.

Lemma 4.2 *Given a simplicial partition of size $O(r)$, with class sizes between 1 and $O(n/r)$ and crossing number $O(r^{1-\delta})$, an LP query can be answered in $O(n/r^\delta + r)$ time.*

Proof: We follow the algorithm Query() from the previous section, except that the algorithm here is nonrecursive and the set $R \subseteq H$ is obtained by picking one point from each P_i in the simplicial partition and dualizing these $O(r)$ points.

Lines 1–2 can be omitted. Line 4 can be done using a linear-time linear programming algorithm [Meg84, CM93]. We perform line 5 as follows. Let h_j denote the halfspace dual to v_j. Then the halfspaces in H that v_j violates correspond to points in P that violate h_j. Since $v_j \in \bigcap R$, we know that h_j contains a point from each P_i. Thus, if h_j is violated by a point $p \in P_i$, then the bounding hyperplane of h_j must cross Δ_i. Hence, line 5 can be carried out by examining the points in P_i for each Δ_i that the bounding hyperplane of h_j crosses.

We now analyze the running time of this algorithm. Since any hyperplane crosses at most $O(r^{1-\delta})$ of the simplices, line 5 requires the examination of $O((n/r)r^{1-\delta}) = O(n/r^\delta)$ points. In particular, the cardinality of the sets V_j is at most $O(n/r^\delta)$. This implies that line 4 can be done in $O(n/r^\delta + r)$ time (assuming $A = \emptyset$). □

Theorem 4.3 *Let H be a set of n halfspaces in $\mathbb{R}^d$, and $q \leq n^\alpha$ for a certain constant $\alpha > 0$ (depending on $d \geq 3$). Then a sequence of q LP queries and $< q$ insertions/deletions on H can be performed in $O(n \log q)$ time and $O(n)$ space.*

Proof: Set $\alpha = \min\{\beta/d,\, 1/(d+1)\}$. Apply the Partition Theorem (Lemma 4.1) with $r = q^d$. The preprocessing time is thus $O(n \log q)$. Now, by Lemma 4.2, an LP query can be answered in $O(n/r^{1/d} + r) = O(n/q)$, so q queries can be answered in $O(n)$ time. To permit insertions, we add (P_0, Δ_0) to the simplicial partition, where P_0 contains the new points that are inserted and Δ_0 is a sufficiently large simplex containing P_0. Deletions can be handled directly. We see that the class-size condition in Lemma 4.2 remains true if the number of updates is less than $n/r = n/q^d$. □

For 1-dimensional LP queries (i.e., ray shooting queries), Theorem 4.3 can be used to construct an f-face convex hull of n points in $O(n \log f)$ time if $f \leq n^\alpha$; see [Cha95]. For general LP queries, Theorem 4.3 can be used to solve linear programs with few violated constraints [Mat95b, Cha95]; in the case $d = 3$, this includes the *smallest k-enclosing circle problem* [ESZ94, Mat95a]. For this circle problem, we can improve the $O(n \log n + k^3 n^\varepsilon)$ running time of Matoušek's algorithm [Mat95b] to $O(n \log k + k^3 n^\varepsilon)$ for small k. (Note that the circle problem uses a nonlinear convex function for w; it can be checked that the same techniques are still applicable.)

5 Discussion

In this paper, we follow recent trends in demonstrating that algorithms based on parametric search can sometimes be replaced by simpler randomized algorithms [Mat91, DMN92, AS95]. Although we only study the problem of LP queries, the technique we use is quite general. As Clarkson's LP algorithm extends to the general class of *LP-type problems* [SW92], we can answer any LP-type queries, assuming that a data structure is available for reporting all constraints violating a given basis. (To get a high-probability bound as in Theorem 3.1 or apply derandomization techniques, we need an extra assumption concerning the VC dimension of a certain range space [CM93].)

For example, with appropriate range searching structures, our technique can answer linear optimization queries over an intersection of congruent balls in $\mathbb{R}^3$ [AES95]. Matoušek's parametric search technique is not applicable here; in the case of nonlinear constraints, a more sophisticated parametric search technique due to Toledo [Tol92] is required.

The case of ray shooting deserves a few more words due to its various applications. In this case, the algorithm Query() in Section 3 is particularly simple: the for-loop in line 3 needs to be executed twice rather than $d+1$ times, and only one recursive call needs to be made because of the decomposability of the ray shooting problem. In the terminology of [Mul93], this query algorithm is an instance of *bottom-up sampling*.

The same approach in fact works for ray shooting amidst a collection of general objects, assuming that a data structure is available for reporting all objects intersecting a given line segment. By Observation 2.3, such a reporting structure can be derived from a data structure for the *segment emptiness problem*: decide whether a given line segment intersects any object. Previously, Agarwal and Matoušek [AM93] used parametric search to reduce the ray shooting problem to the segment emptiness problem; the bottom-up sampling approach provides a simple alternative. (By modifying the definition of the emptiness problem, this approach can also be applied to the *collision detection problem* studied in [ST95].)

For example, if the objects are hyperplanes in $\mathbb{R}^d$, we get a linear-space data structure that can answer ray shooting queries in $O(n^{1-1/d})$ time, using a range searching structure by Matoušek [Mat93b].

References

[AES95] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. In *Proc. 11th ACM Sympos. Comput. Geom.*, pages 39–50, 1995.

[AM93] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22:764–806, 1993.

[AM95] P. K. Agarwal and J. Matoušek. Dynamic halfspace range reporting and its applications. *Algorithmica*, 13:325–345, 1995.

[AS95] P. K. Agarwal and M. Sharir. Efficient randomized algorithms for some geometric optimization problems. In *Proc. 11th ACM Sympos. Comput. Geom.*, pages 326–335, 1995.

[AvKO93] P. K. Agarwal, M. van Kreveld, and M. Overmars. Intersection queries in curved objects. *J. Algorithms*, 15:229–266, 1993.

[BCM93] H. Brönnimann, B. Chazelle, and J. Matoušek. Product range spaces, sensitive sampling, and derandomization. In *Proc. 34th IEEE Sympos. Found. Comput. Sci.*, pages 400–409, 1993.

[CG86] B. Chazelle and L. J. Guibas. Fractional cascading II: Applications. *Algorithmica*, 1:163–191, 1986.

[Cha95] T. M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. In *Proc. 11th ACM Sympos. Comput. Geom.*, pages 10–19, 1995. *Discrete Comput. Geom.*, to appear.

[Cla95] K. L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42:488–499, 1995.

[CM93] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 281–290, 1993.

[DK90] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra: a unified approach. In *Proc. 17th Int. Colloq. Automata, Languages, and Programming*, Lect. Notes in Comput. Sci., vol. 443, Springer-Verlag, pages 400–413, 1990.

[DMN92] M. Dillencourt, D. Mount, and N. Netanyahu. A randomized algorithm for slope selection. *Int. J. Comput. Geom. Appl.*, 2:1–27, 1992.

[ESZ94] A. Efrat, M. Sharir, and A. Ziv. Computing the smallest k-enclosing circle and related problems. *Comput. Geom. Theory Appl.*, 4:119–136, 1994.

[HW87] D. Haussler and E. Welzl. ε-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987.

[Mat91] J. Matoušek. Randomized optimal algorithm for slope selection. *Inform. Process. Lett.*, 39:183–187, 1991.

[Mat92a] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[Mat92b] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2:169–186, 1992.

[Mat93a] J. Matoušek. Linear optimization queries. *J. Algorithms*, 14:432–448, 1993. Also with O. Schwarzkopf in *Proc. 8th ACM Sympos. Comput. Geom.*, pages 16–25, 1992.

[Mat93b] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10:159–182, 1993.

[Mat94] J. Matoušek. Geometric range searching. *ACM Comput. Surveys*, 26:421–461, 1994.

[Mat95a] J. Matoušek. On enclosing k points by a circle. *Inform. Process. Lett.*, 53:217–221, 1995.

[Mat95b] J. Matoušek. On geometric optimization with few violated constraints. *Discrete Comput. Geom.*, 14:365–384, 1995.

[Meg83] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30:852–865, 1983.

[Meg84] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31:114–127, 1984.

[MS93] J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete Comput. Geom.*, 10:215–232, 1993.

[Mul93] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice-Hall, Englewood Cliffs, N.J., 1993.

[Pug90] W. Pugh. Skip lists: A probabilistic alternative to balanced trees. *Commun. ACM*, 33:668–676, 1990.

[Sei91] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.

[ST95] E. Schömer and C. Thiel. Efficient collision detection for moving polyhedra. In *Proc. 11th ACM Sympos. Comput. Geom.*, pages 51–60, 1995.

[SW92] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *Proc. 9th Sympos. Theoret. Aspects Comput. Sci.*, Lect. Notes in Comput. Sci., vol. 577, Springer-Verlag, pages 569–579, 1992.

[Tol92] S. Toledo. Maximizing non-linear concave functions in fixed dimension. In *Proc. 33rd IEEE Sympos. Found. Comput. Sci.*, pages 696–685, 1992.