Mathematical Programming 44 (1989) 203-212
North-Holland

A RANDOMIZED ALGORITHM FOR FIXED-DIMENSIONAL LINEAR PROGRAMMING

M.E. DYER
University of Leeds, Leeds, UK

A.M. FRIEZE
Carnegie-Mellon University, Pittsburgh, USA, and Queen Mary College, London, UK

Received 25 September 1987
Revised manuscript received 28 March 1988

We give a (Las Vegas) randomized algorithm for linear programming in a fixed dimension d for which the expected computation time is $O(d^{(3+\varepsilon_d)d}n)$, where $\lim_{d\to\infty}\varepsilon_d = 0$. This improves the corresponding worst-case complexity, $O(3^{d^2}n)$. The method is based on a recent idea of Clarkson. Two variations on the algorithm are examined briefly.

Key words: Linear programming, randomized algorithm.

1. Introduction

In [8], Megiddo presented a technique for a multi-dimensional search problem. The technique is a major component of an algorithm for linear programming for which the computational effort grows only linearly with the number of constraints in any fixed dimension d. See also Dyer [3] and Megiddo [7]. The complexity of Megiddo's linear programming algorithm, measured in an operation-count model, is $O(2^{c2^d}n)$, where n is the number of constraints and $c>0$ is constant. Dyer [4] reduced this to $O(3^{d^2}n)$ (as did Clarkson [1] independently), and showed that this approach could not be expected to lead to algorithms with complexity better than $O(d!n)$.

In [2], Clarkson discusses the application of random sampling to some problems in Computational Geometry. (For related work, see also [5].) In this note, we show how the same ideas can be used to obtain a (Las Vegas) randomized algorithm for fixed-dimensional linear programming and related problems with a better constant factor than the deterministic algorithms. (A Las Vegas algorithm always computes the correct answer, but its time-bound is only in expectation.) The (expected) time complexity of our linear programming algorithm is $O(d^{(3+\varepsilon_d)d}n)$ where $\lim_{d\to\infty}\varepsilon_d = 0$.

2. The search problem

The following problem was discussed in [4, 8]. We are given a set S of n affine functions in $\mathbb{R}^d$:

$$h_i(x) = a_i^T x + b_i \quad (i = 1, \ldots, n).$$

The set $\{x \in \mathbb{R}^d : h_i(x) = 0\}$ is the associated hyperplane H_i. Let $x_0 \in \mathbb{R}^d$. We wish to determine the sign of $h_i(x_0)$ for some fixed proportion αn values of i. Here the sign of a number will mean the usual mapping into the set $\{-1, 0, 1\}$. We denote this by $\text{sign}(\cdot)$. We will also abbreviate $\text{sign}(h_i(x_0))$ to $\text{sign}(h_i)$ where no ambiguity arises.

An *enquiry* is an evaluation of $\text{sign}(h)$ for some given affine function h in $\mathbb{R}^d$. (This need not be one of the h_i.) We access information about x_0 through an "oracle" which can answer enquiries. We input any h, and it outputs $\text{sign}(h)$. Given α, the problem is to determine αn values of $\text{sign}(h_i)$ while "minimizing" the number of enquiries. Rather surprisingly, Megiddo showed in [8] that there exists an α for which a number of enquiries suffices which is independent of n. In fact, he showed that if $\alpha = 2^{1-2^d}$, then 2^{d-1} enquiries suffice. Dyer [4] (see also [1]) showed that, in fact, $\alpha = \frac{1}{2}$ is achievable with not too many more enquiries, i.e. at most $9(\sqrt{72})^{d-2}$. This lead to the above-mentioned improvement in the running time for linear programming.

The approach taken here to the search problem is somewhat different from [8] and [4] and is in fact adapted from [2]. We choose a random subset R from the hyperplanes in S and consider the polyhedral decomposition of $\mathbb{R}^d$ induced by these $r = |R|$ hyperplanes. Each cell in the subdivision is then triangulated, i.e. subdivided into simplices. If r is large enough, then, with sufficiently high probability p, each simplex in this decomposition meets at most βn hyperplanes of S, for some fixed $\beta < 1$. If this holds, then all we have to do is to use the oracle to locate x_0 within a simplex of the decomposition. By so doing, we will locate x_0 with respect to at least $(1-\beta)n$ hyperplanes of S, as required. We can in fact achieve $\beta = \frac{1}{2}$ with high probability using $O(d^3 \log d)$ enquiries. It remains to fill in the details. We will describe the algorithm and develop its analysis as we proceed.

Step 0. Randomly choose $R \subseteq S$, $|R| = r = \lceil (3 \log d + 6) d(d+1) \rceil$. (All logarithms will be to base e.)

Step 1. Construct a polyhedron P containing x_0 where

$$P = \{x \in \mathbb{R}^d : \text{sign}(h_i(x)) = \text{sign}(h_i(x_0)),\ i \in R\}.$$

The determination of $h_i(x_0)$, $i \in R$ clearly requires r enquiries of the oracle. If there exists any $i \in R$ such that $h_i(x_0) = 0$, then we can obviously reduce the problem dimension by eliminating a variable. We may therefore assume the worst case, i.e. $\dim(P) = d$.

We want to partition P into simple regions which are completely determined by relatively small sets of hyperplanes in R. The motivating example is where P is

bounded and the decomposition consists of simplices whose vertices are also vertices of P. Unfortunately, P may be unbounded and/or have no vertices. Nevertheless, let us assume

A: P is bounded

and return later to discuss the details of removing this assumption.

There are numerous ways of decomposing P into simplices. The following method (also suggested in [2]) suits us. Choose a vertex z of P, and inductively assume a decomposition of each facet $F_j, j = 1, 2, \ldots, m$ $(m \leq r)$ of P into a set Δ_j of $(d-1)$-dimensional simplices. Take each $\sigma \in \bigcup_{j=1}^{m} \Delta_j$, and, if $z \notin \text{aff}(\sigma)$ then create a d-dimensional simplex σ_z which is the convex hull of $\sigma \cup \{z\}$. Let Δ denote the set of simplices so produced.

Step 2. Locate $\sigma^* \in \Delta$ such that $x_0 \in \sigma^*$.

How long does Step 2 take, and more importantly, how many enquiries are needed? Note first that $O(d^3 r(\frac{r}{d})) = O(r^{d+1})$ operations are ample for determining a vertex of P by examining all possible solutions, i.e. all intersections of d members of R. (Note that, unless otherwise stated, the implied constants in O expressions are true constants, independent of both n and d.) Our next task is to determine j such that $x_0 \in \bigcup_{\sigma \in \Delta_j} \sigma_z$. We start by choosing a pair of hyperplanes H_k, H_l, neither of which contains z. Let H be the hyperplane containing z and $H_k \cap H_l$. Determine the location of x_0 with respect to H by an enquiry. Whatever the outcome, we can eliminate one of k and l as a candidate for j. We take another pair of candidate hyperplanes and repeat. After at most $(r-1)$ enquiries, we will have determined j. Given j, locating a σ_z containing x_0 is essentially a $(d-1)$-dimensional problem of the same form: we must find a $\sigma \in \Delta_j$ such that the point x_0' in H_j on the line through z and x_0 lies in σ. Enquiries about the position of x_0' with respect to a $(d-2)$-dimensional hyperplane H' in $\text{aff}(H_j)$ can be answered by asking for the position of x_0 relative to $\text{aff}(H' \cup \{z\})$. Thus at most $d(r-1)$ enquiries are needed in Step 2. In addition, there will be $O(r^{d+1})$ arithmetic operations.

Step 3. Determine the set $S' \subseteq S$ of hyperplanes which cut σ^*;
if $|S'| \leq \frac{1}{2}n$ **then** SUCCESS **else** FAILURE.

The occurrence of SUCCESS means that we have determined $\text{sign}(h_i(x_0))$ for at least half of the hyperplanes, using at most $r + d(r-1)$ enquiries. The following Lemma is somewhat surprising. The essential idea is due to Clarkson [2].

Lemma 2.1. $\Pr(\text{FAILURE}) \leq (\frac{1}{2})^{d(d+1)} \leq \frac{1}{64}$ *if* $d \geq 2$.

Proof. FAILURE implies the occurrence of the following event:

There exist subsets $X_1, X_2, \ldots, X_{d+1}$ of R, all of size d, such that

(i) the hyperplanes associated with each X_i determine the vertices v_i of a simplex σ,

(ii) at least $\frac{1}{2}n$ members of S cut σ,

(iii) none of the hyperplanes in (ii) are in R.

Hence, if NUM is the number of collections of subsets $X_1, X_2, \ldots, X_{d+1}$ satisfying

(i), (ii), (iii),

$$\Pr(\text{FAILURE}) \leq E(\text{NUM})$$

$$\leq \alpha_d \Pr((\text{i}), (\text{ii}), (\text{iii})) \quad \text{where } \alpha_d = \binom{\binom{r}{d}}{d+1}$$

$$\leq \binom{\lfloor (re/d)^d \rfloor}{d+1} \Pr((\text{iii}) | (\text{ii}))$$

$$\leq ((re/d)^d e/(d+1))^{d+1} \binom{\lfloor \frac{1}{2}n \rfloor - d(d+1)}{r-d(d+1)} \Big/ \binom{n-d(d+1)}{r-d(d+1)}$$

$$\leq ((re/d)^d e/(d+1))^{d+1} (\tfrac{1}{2})^{r-d(d+1)}. \tag{2.1}$$

Here we have used the inequality $\binom{r}{d} \leq (re/d)^d$, and the fact that, for any integers $a \geq b > c \geq d \geq 0$,

$$\binom{b-d}{c-d} \Big/ \binom{a-d}{c-d} = \prod_{i=d}^{c-1} \frac{(b-i)}{(a-i)} \leq \left(\frac{b}{a}\right)^{c-d}$$

since $(b-i)/(a-i) \leq b/a$ for $0 \leq i < a$.

Hence, since $r^{d(d+1)}(\frac{1}{2})^r$ decreases for $r > d(d+1)/\log 2$,

$$\Pr(\text{FAILURE}) \leq \left(\left(\frac{(3 \log d + 6)(d+1)e}{32 d^{3 \log 2}} \right)^d \frac{e}{d+1} \right)^{d+1} \tag{2.2a}$$

$$\leq (\tfrac{1}{2})^{d(d+1)} \quad \text{for } d \geq 2. \tag{2.2b}$$

To see that (2.2b) follows from (2.2a), first note that we can easily show by induction that $3 \log d + 6 < 4.04d$ for all $d \geq 2$. Thus, since $e/(d+1) < 1$ for $d \geq 2$,

$$\Pr(\text{FAILURE}) < \left(\frac{4.04ed(d+1)}{32 d^{3 \log 2}} \right)^{d+1}. \tag{2.3}$$

The bracketed quantity in (2.3) clearly decreases with d since $3 \log 2 > 2$, and for $d = 2$, its value is less than 0.49. □

3. Application to linear programming

Suppose we are given a linear program with constraint set defined by S. Let x_0 be the unknown optimal solution. The technical details concerning infeasibility and unboundedness may be found in references [1, 4, 8].

Once we know $\text{sign}(h_i(x_0))$, we can eliminate this constraint or reduce the dimension of the problem. It is shown in the above references that the oracle in the

search problem has to solve at most three $(d-1)$-dimensional linear programs. One way, therefore, to solve a linear program is

LP Algorithm
repeat
repeat Steps 0 to 3 **until** SUCCESS; remove constraints
until x_0 is determined.

(The one-dimensional problems are solved by computing the minimum or maximum of n numbers.)

Theorem 3.1. *Let $T(n, d)$ be the expected number of operations in the* LP *algorithm for a problem with at most n constraints and dimension at most d. Then*

$$T(n, d) = O((c \log d)^d (d!)^3 n) \quad (d \geq 2) \quad \textit{for some } c > 0.$$

Proof. Let WAIT be the (random) number of repetitions of Steps 0 to 3 until SUCCESS. Clearly $T(n, 1) = an$ for some $a > 0$, and, applying Lemma 2.1 we get

$$T(n, d) \leq E(\text{WAIT})3(r + d(r-1))T(n, d-1) + T(\lfloor \tfrac{1}{2}n \rfloor, d) + O(r^{d+1}) + O(d^2 n)$$

$$\leq A(rdT(n, d-1) + d^2 n + r^{d+1}) + T(\lfloor \tfrac{1}{2}n \rfloor, d) \tag{3.1}$$

for some sufficiently large constant $A > 0$. (The $O(d^2 n)$ term is for checking SUCCESS or FAILURE.)

Now apply Lemma 3.1 below. (The $O(r^{d+1})$, $O(d^2 n)$ terms are absorbed into the $O((2r)^{d+1} n)$ term of the Lemma. This is rather larger than needed here, but is required in later applications of Lemma 3.1.) □

Lemma 3.1. *Let $\tau(n, d)$ satisfy*

(i) $\tau(n, d) \leq A(rd\tau(n, d-1) + (2r)^{d+1} n) + \tau(\lfloor \tfrac{1}{2}n \rfloor, d)$ *for* $n \geq 2, d \geq 2$ *where* $r = r(d)$ *is as defined above.*

(ii) $\tau(n, 1) \leq an$, $\tau(1, d) \leq bd$ *with* $a, b \geq 1$.

Then

$$\tau(n, d) \leq 4(a+b)(K(\log d + 2))^d (d!)^3 n \tag{3.2}$$

where $K = 30(A + 10)$.

Proof. One can easily check (3.2) for the cases $n = 1, d = 1$. Otherwise assume, inductively, that (3.2) holds for $(n, d-1)$ and $(\lfloor \tfrac{1}{2}n \rfloor, d)$. Then

$$\frac{\tau(n, d)}{4(a+b)(K(\log d+2))^d (d!)^3 n} \leq \frac{Ar}{4K(\log d+2)d^2} + \frac{2A(\log d+2)(6d(d+1))^{d+1}}{4K^d (d!)^3} + \tfrac{1}{2}$$

$$< \tfrac{1}{25} + \tfrac{1}{4} + \tfrac{1}{2} < 1.$$

To establish the bounds used here, let the first two terms on the right side of the first inequality be ξ_d, η_d respectively. Now

$$\frac{\xi_d}{\xi_{d-1}} \leqslant \frac{(3\log d+6)d(d+1)+1}{(3\log(d-1)+6)d(d-1)} \cdot \frac{\log(d-1)+2}{\log d+2} \cdot \frac{(d-1)^2}{d^2}$$

$$= \left(d+1+\frac{1}{3d(\log d+2)}\right) \cdot \frac{(d-1)}{d^2} < (d+1+1/d)(d-1)/d^2$$

$$= 1-1/d^3 < 1.$$

Also

$$\frac{\eta_d}{\eta_{d-1}} = \frac{6}{K} \cdot \left(1+\frac{1}{d}\right)^2 \cdot \frac{1}{d-1} \cdot \left(1+\frac{2}{d-1}\right)^{d-1} \cdot \frac{\log d+2}{\log(d-1)+2}$$

$$\leqslant \frac{6}{K} \cdot \tfrac{9}{4} \cdot 1 \cdot e^2 \cdot 2 \quad \text{since } d \geqslant 2$$

$$< \frac{200}{K} \leqslant \tfrac{2}{3} < 1.$$

We have used here the fact that $(\log d+2)/(\log(d-1)+2) < 2$ for all $d \geqslant 2$, the (easy) proof of which we leave to the reader.

Thus both ξ_d and η_d are strictly decreasing for $d \geqslant 2$. Therefore we need only consider the case $d = 2$. But then

$$\xi_2 < \tfrac{49}{43} \cdot \frac{A}{K} = \tfrac{49}{1290} \cdot \frac{A}{(A+10)} < \tfrac{1}{25}.$$

$$\eta_2 < \tfrac{255000}{32} \cdot \frac{A}{K^2} = \tfrac{2550}{288} \cdot \frac{A}{(A+10)^2}$$

$$< \frac{9A}{(A+10)^2} \leqslant \tfrac{9}{40} < \tfrac{1}{4}. \qquad \square$$

We now consider relaxing the boundedness assumption A. In order to prove Theorem 3.1, we must partition P into regions which are determined by at most $d(d+1)$ members of R, and are cut by no other members of R, and so that Pr(SUCCESS) is bounded away from zero.

Now a standard result in polyhedral theory (see for example Schrijver [9]) is that we may write $P = L+Q$ where L is a linear space and Q is a (pointed) polyhedron, such that $Q = P \cap L^{\perp}$. Write $l = \dim(L)$ and $q = \dim(Q)$, so $d = l+q$. Let $P = \{x \in \mathbb{R}^d : Ax \leqslant b\}$, where A is an $r \times d$ matrix with rows a_i^{T}, for $i \in R$, and b is $r \times 1$. Now $L = \{x \in \mathbb{R}^d : Ax = 0\}$. A basis for this can be determined from any set Λ of l linearly independent rows of A in $O(rd^3)$ operations. The $a_i (i \in \Lambda)$ form a basis for $L^{\perp}$. It is now easy to determine Q. We then partition Q into regions $W_1, W_2, \ldots$ and use $W_1+L, W_2+L, \ldots$ as a partition of P. To partition Q, we make

a projective transformation to a bounded polyhedron Q' as follows. Let us assume, for notational convenience, that we still have $Q=\{x\in\mathbb{R}^d: Ax\leq b\}$. Choose any $\mu\in\mathbb{R}^d$ such that $\mu>0, \mu^{\mathrm{T}}b>-1$. (Such a μ clearly always exists, and can be determined in $O(r)$ operations.) Now make the coordinate transformation $f: Q\to\mathbb{R}^d$,

$$y=f(x)=\frac{x}{1+\mu^{\mathrm{T}}(b-Ax)} \quad \text{with inverse } x=f^{-1}(y)=\frac{(1+\mu^{\mathrm{T}}b)y}{1+\mu^{\mathrm{T}}Ay}.$$

Now $Q'=\mathrm{cl}(f(Q))=\{y\in\mathbb{R}^d: (1+\mu^{\mathrm{T}}b)Ay\leq(1+\mu^{\mathrm{T}}Ay)b, 1+\mu^{\mathrm{T}}Ay\geq 0\}$ where cl denotes closure. Q' is defined by $(r+1)$ inequalities. It is straightforward to show that its recession cone [9] is $\{0\}$, and thus it is bounded. We now locate $y_0=f(x_0)$ in a simplex σ of Q'. Observe that enquiries transform to enquiries. Having located y_0, we transform σ back to a region $f^{-1}(\sigma)$ of Q. In making the inverse transformation, vertices of Q' on the facet $1+\mu^{\mathrm{T}}Ay=0$ of Q' transform to extreme rays of Q. Thus we locate x_0 in a region $W+L$, determined by vertices and rays of Q and lines of L. The region is determined by at most $l+1(q+1)\leq d(d+1)$ hyperplanes in R.

We see that even in the general case we can carry out Steps 0 to 3 using $O(r^{d+1})$ operations and $r+dr$ enquiries. Lemma 2.1 still holds, and so does Theorem 3.1.

4. Two variations

The *expected* running time of the algorithm for linear programming in Section 3 is superior to the *worst-case* running time of previous algorithms. However, for any fixed value of d, there remains a significant probability that its running time is ωn with $\omega=\omega(n)\to\infty$ sufficiently slowly. The reason for this is that by the time we realise (in Step 3) that we have not eliminated sufficiently many constraints we may have already done work proportional to n. We can avoid this as follows. Replace Step 0 by

Step 0'. Randomly choose $R\subseteq S; |R|=r$. {Use random sampling to reduce the likelihood of FAILURE}.
$m:=\lceil\sqrt{n}\rceil$ {any $m=o(n/\log n)$ would suffice}.
L1: Randomly choose $M\subseteq S-R, |M|=m$;
for each simplex σ in the decomposition defined by R **do** compute the number m_σ of hyperplanes of M that cut σ;
L2: **if** $\exists\sigma$ such that $m_\sigma\geq 0.49m$ **then**
Reject R and **go to** L1 **else**
Accept R and continue with Steps 1, 2, 3.

For various technical purposes below, we need to bound the number of "simplicial regions", NR say, into which the partition of $\mathbb{R}^d$ determined by R can be triangulated. We may do this as follows.

Lemma 4.1. $NR \leq (2r)^d$.

Proof. If $d = 1$, clearly $NR \leq 2r$. Assume inductively that the result is true for $d-1$. Each hyperplane in R will therefore be triangulated, as a copy of $\mathbb{R}^{d-1}$, into at most $(2(r-1))^{d-1} \leq (2r)^{d-1}$ simplicial regions. There are thus at most $r(2r)^{d-1}$ such $(d-1)$-dimensional regions in total. But each such $(d-1)$-dimensional region is a face of at most two d-dimensional regions. Hence there can be at most $2r(2r)^{d-1} = (2r)^d$ d-dimensional regions. □

Lemma 4.2. *Let E denote the following event:*

R is not rejected by Step 0′, and yet R gives rise to a region which is cut by more than $\frac{1}{2}n$ members of $S - R$.

Then $\Pr(E) \leq (2r)^d e^{-\beta\sqrt{n}}$, *where $\beta > 0$ is some constant.*

Proof. For a fixed region σ, let n_σ be the total number of hyperplanes that cut σ. Then

$$E \subseteq \bigcup_\sigma \{m_\sigma < 0.49m \text{ and } n_\sigma \geq \tfrac{1}{2}n\}.$$

But since $E(m_\sigma) \sim mn_\sigma/n$, it follows that $\Pr(m_\sigma < 0.49m \mid n_\sigma \geq \frac{1}{2}n) \leq e^{-\beta m}$ as may be deduced from Section 6 of Hoeffding [6] (which deals with probability bounds for sampling without replacement). As there are at most $(2r)^d$ regions (Lemma 4.1), the Lemma follows. □

Lemma 4.2 shows that we are very unlikely to accept an R in Step 0′ which would lead to FAILURE in Step 3. From Lemma 4.1, the computation cost of the statements between $L1$ and $L2$ of Step 0′ is clearly $O(d^2(2r)^d\sqrt{n})$, but what of the probability that we accept R in Step 0′?

Lemma 4.3. $\Pr(R \text{ is rejected in Step } 0') \leq (1+o(1))(\frac{3}{5})^{d(d+1)}$ *where the* $o(1)$ *term is valid for* $n \to \infty$ *provided* $d = o(n^{1/4})$.

Proof. In the notation of Lemma 4.2, $\Pr(m_\sigma \geq 0.49m \mid n_\sigma < 0.48n) \leq e^{-\gamma\sqrt{n}}$ for some $\gamma > 0$. This again follows from Hoeffding [6]. Thus, if α_d is as defined in the proof of Lemma 2.1, it follows that

$$\Pr(R \text{ is rejected}) \leq (2r)^d e^{-\gamma\sqrt{n}} + \alpha_d \Pr((\text{i}), (\text{ii})', (\text{iii}))$$

where, in the notation of Lemma 2.1, (ii)′ is (ii) with $\frac{1}{2}n$ replaced by $0.48n$. The Lemma now follows after a similar calculation to that of Lemma 2.1. First observe that the condition on d implies that $(2r)^d e^{-\gamma\sqrt{n}} = o(c^{d(d+1)})$ for every constant c. Also, the effect of replacing (ii) by (ii)′ is that the $\frac{1}{2}$ in (2.1) becomes 0.52, and then the $32\ (=(0.5)^{-5})$ in (2.2a) becomes $26.3\ (<(0.52)^{-5})$. Thus the $\frac{1}{2}$ in (2.2b) becomes $0.49 \times 32/26.3 < \frac{3}{5}$. (Here we have used the value 0.49 given in the proof of Lemma 2.1, rather than the $\frac{1}{2}$ claimed there, since using $\frac{1}{2}$ does not quite establish the bound.) □

It follows that if TIME is the (random) execution time to complete Step 0',

$$E(\text{TIME}) = O(d^2(2r)^d\sqrt{n}). \tag{4.1}$$

We now consider the effect of replacing Step 0 by Step 0' in our linear programming algorithm. We shall assume that the value of m used in Step 0' is constant throughout, i.e. it does not change when the number of constraints is reduced. We choose a random subset of size $\min\{m, |S-R|\}$. Thus the probability bound in Lemma 4.2 will continue to hold as the recursion progresses. The effect of this change is summarized in the following.

Theorem 4.1. *Suppose we replace Step* 0 *by Step* 0' *in our linear programming algorithm. Let* RUNTIME *be the random execution time of the modified algorithm on a linear program with n constraints and d variables. Then, for some constants* $A', c' > 0$,

(a) $T'(n, d) = E(\text{RUNTIME}) \leq A'(c' \log d)^d (d!)^3 n$

and, for any $\varepsilon > 0$,

(b) $\Pr(\text{RUNTIME} \geq A'(c' \log d)^d (d!)^3 n + o(n)) = O(e^{-\delta n^{1/2-\varepsilon}})$

for some constant $\delta > 0$ *and the* o, O *terms are valid as* $n \to \infty$ *provided* $d = o(\log n / \log\log n)$.

Proof (outline). (a) Proceeding as in Theorem 3.1 we find

$$T'(n, d) \leq E(\text{WAIT})\{E(\text{TIME}) + O(rd)T'(n, d-1) + O(d^2 n)\} + T(\lfloor \tfrac{1}{2}n \rfloor, d) + O(r^{d+1}). \tag{4.2}$$

The result follows on using Lemma 3.2, $E(\text{WAIT}) = O(1)$, and (4.1).

(b) Suppose that we execute the algorithm without any occurrences of FAILURE. In this case, the number $\xi(n, d)$ of executions of Step 0' satisfies

$$\xi(n, d) \leq A'' rd\xi(n, d-1) + \xi(\lfloor \tfrac{1}{2}n \rfloor, d) \quad \text{for some constant } A'' > 0.$$

One can easily show from this that $\xi(n, d) = O((A''d^4 \log n)^d)$ for all n, d. (Note that $\xi(n, 0) = O(1)$).

It follows from these remarks and Lemma 4.2 that, except for probability $O((A''d^4 \log n)^4 e^{-\beta\sqrt{n}})$, there are no occurrences of FAILURE in our algorithm. From Lemma 4.3, we can then see that there exists $\theta > 0$ such that, except for probability $O(e^{-\theta d^2 n^{1/2-\varepsilon}})$, a given execution of Step 0' requires at most $n^{1/2-\varepsilon}$ repetitions of loop *L*1 to *L*2. Thus, except for probability $O((A''d^4 \log n)^d e^{-\theta d^2 n^{1/2-\varepsilon}})$, Step 0' requires no more than $O((A''d^4 \log n)^d \times n^{1/2-\varepsilon} \times d^2(2r)^d\sqrt{n}) = o(n)$ (provided $d = o(\log n / \log\log n)$) operations in total. Thus, with the required probability, $\text{RUNTIME} \leq \tau(n, d) + o(n)$ where $\tau(n, d) \leq A'(c' \log d)^d (d!)^3 n$, by comparison with (4.2). □

Note that the condition on d in (b) of Theorem 4.1 is equivalent to requiring that $T'(n, d)$ be "almost linear" in n, i.e. $T'(n, d) = O(n^{1+o(1)})$. This is the region of most interest, though a similar sort of result can be obtained for d growing somewhat faster with n.

Finally let us consider a version of our algorithm in which we omit to check the number n_σ of hyperplanes cutting the region σ in which we have located x_0. Let $T''(n, d)$ denote the maximum expected running time of the linear programming algorithm that uses this strategy for problems with at most n constraints and dimension at most d. Then

$$T''(n, d) \leq O(rd)T''(n, d-1) + O(r^{d+1}) + T''(\lfloor \tfrac{1}{2}n \rfloor, d) \Pr(n_\sigma \leq \lfloor \tfrac{1}{2}n \rfloor) + T''(n, d) \Pr(n_\sigma > \lfloor \tfrac{1}{2}n \rfloor).$$

But Lemma 2.1 implies that $\Pr(n_\sigma > \lfloor \frac{1}{2}n \rfloor) \leq (\frac{1}{2})^{d(d+1)}$ and, since $T''(\lfloor \frac{1}{2}n \rfloor, d) \leq T''(n, d)$, we obtain

$$(1-(\tfrac{1}{2})^{d(d+1)})T''(n, d) \leq O(rd)T''(n, d-1) + O(r^{d+1}) + (1-(\tfrac{1}{2})^{d(d+1)})T''(\lfloor \tfrac{1}{2}n \rfloor, d)$$

and, on dividing through by $(1-(\frac{1}{2})^{d(d+1)})$,

$$T''(n, d) \leq O(rd)T''(n, d-1) + O(r^{d+1}) + T''(\lfloor \tfrac{1}{2}n \rfloor, d). \tag{4.3}$$

Thus we have

Theorem 4.2. *If we omit to check the number of hyperplanes which cut the region containing x_0 (i.e. Step 3 of the search algorithm), then the expected running time of the linear programming algorithm is $O((c'' \log d)^d (d!)^3 n)$ for some $c'' > 0$.*

Proof. Lemma 3.1 and (4.3). □

References

[1] K.L. Clarkson, "Linear programming in $O(n \times 3^{d^2})$ time," *Information Processing Letters* 22 (1986) 21-24.

[2] K.L. Clarkson, "New applications of random sampling to computational geometry," *Journal of Discrete and Computational Geometry* 2 (1987), 195-222.

[3] M.E. Dyer, "Linear time algorithms for two- and three-variable linear programs," *SIAM Journal on Computing* 13 (1984) 31-45.

[4] M.E. Dyer, "On a multidimensional search problem and its application to the Euclidean one-centre problem," *SIAM Journal on Computing* 15 (1986) 725-738.

[5] D. Haussler and E. Welzl, "ε-nets and simplex range queries," *Discrete and Computational Geometry* 2 (1987) 127-151.

[6] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association* 58 (1963) 13-30.

[7] N. Megiddo, "Linear-time algorithms for linear programming $\mathbb{R}^3$ and related problems," *SIAM Journal on Computing* 12 (1983) 759-776.

[8] N. Megiddo, "Linear programming in linear time when the dimension is fixed," *Journal of the Association for Computing Machinery* 31 (1984) 114-127.

[9] A. Schrijver, *Theory of Linear and Integer Programming* (Wiley, Chichester, 1986).