# A Randomized Polynomial-Time Simplex Algorithm for Linear Programming

Jonathan A. Kelner*
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Daniel A. Spielman†
Department of Computer Science
Yale University

## ABSTRACT

We present the first randomized polynomial-time simplex algorithm for linear programming. Like the other known polynomial-time algorithms for linear programming, its running time depends polynomially on the number of bits used to represent its input.

We begin by reducing the input linear program to a special form in which we merely need to certify boundedness. As boundedness does not depend upon the right-hand-side vector, we run the shadow-vertex simplex method with a random right-hand-side vector. Thus, we do not need to bound the diameter of the original polytope.

Our analysis rests on a geometric statement of independent interest: given a polytope $A\boldsymbol{x} \leq \boldsymbol{b}$ in isotropic position, if one makes a polynomially small perturbation to $\boldsymbol{b}$ then the number of edges of the projection of the perturbed polytope onto a random 2-dimensional subspace is expected to be polynomial.

## 1. INTRODUCTION

Linear programming is one of the fundamental problems of optimization. Since Dantzig [3] introduced the simplex method for solving linear programs, linear programming has been applied in a diverse range of fields including economics, operations research, and combinatorial optimization. From a theoretical standpoint, the study of linear programming has motivated major advances in the study of polytopes, convex geometry, combinatorics, and complexity theory.

While the simplex method was the first practically useful approach to solving linear programs and is still one of the most popular, it was unknown whether any variant of the simplex method could be shown to run in polynomial time in the worst case. In fact, most common variants have been shown to have exponential worst-case complexity. In contrast, algorithms have been developed for solving linear programs that do have polynomial worst-case complexity [10, 9, 5, 1]. Most notable among these have been the ellipsoid method [10] and various interior-point methods [9]. All previous polynomial-time algorithms for linear programming of which we are aware differ from simplex methods in that they are fundamentally geometric algorithms: they work either by moving points inside the feasible set, or by enclosing the feasible set in an ellipse. Simplex methods, on the other hand, walk along the vertices and edges defined by the constraints. The question of whether such an algorithm can be designed to run in polynomial time has been open for over fifty years.

We recall that a linear program is a constrained optimization problem of the form:

$$\begin{aligned} &\textbf{maximize} && \boldsymbol{c} \cdot \boldsymbol{x} \\ &\textbf{subject to} && A\boldsymbol{x} \leq \boldsymbol{b},\ \boldsymbol{x} \in \mathbb{R}^d, \end{aligned} \tag{1}$$

where $\boldsymbol{c} \in \mathbb{R}^d$ and $\boldsymbol{b} \in \mathbb{R}^n$ are column vectors, and $A$ is an $n \times d$ matrix. The vector $\boldsymbol{c}$ is the *objective function*, and the set $P := \{\boldsymbol{x} \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$ is the set of *feasible* points. If it is non-empty, $P$ is a convex polyhedron, and each of its extreme vertices will be determined by $d$ constraints of the form $\boldsymbol{a}_i \cdot \boldsymbol{x} = b_i$, where $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ are the rows of $A$. It is not difficult to show that the objective function is always maximized at an extreme vertex, if this maximum is finite.

The first simplex methods used heuristics to guide a walk on the graph of vertices and edges of $P$ in search of one that maximizes the objective function. In order to show that any such method runs in worst-case polynomial time, one must prove a polynomial upper bound on the diameter of polytope graphs. Unfortunately, the existence of such a bound is a wide-open question: the famous Hirsch Conjecture asserts that the graph of vertices and edges of $P$ has diameter at most $n-d$, whereas the best known bound for this diameter is superpolynomial in $n$ and $d$ [8].

Later simplex methods, such as the self-dual simplex method and the criss-cross method, avoided this obstacle by considering more general graphs for which diameter bounds were known. However, even though these graphs have polynomial diameters, they have exponentially many vertices, and nobody had been able to design a polynomial-time algorithm that provably finds the optimum after following a polynomial number of edges. In fact, essentially every such algorithm has well-known counterexamples on which the walk takes exponentially many steps.

In this paper, we present the first randomized polynomial-time simplex method. Like the other known polynomial-

†Partially supported by NSF grant CCR-0324914.
*Partially supported by an NSF Graduate Fellowship and NSF grant CCR-0324914.

time algorithms for linear programming, the running time of our algorithm depends polynomially on the bit-length of the input. We do *not* prove an upper bound on the diameter of polytopes. Rather we reduce the linear programming problem to the problem of determining whether a set of linear constraints defines an unbounded polyhedron. We then randomly perturb the right-hand sides of these constraints, observing that this does not change the answer, and we then use a shadow-vertex simplex method to try solve the perturbed problem. When the shadow-vertex method fails, it suggests a way to alter the distributions of the perturbations, after which we apply the method again. We prove that the number of iterations of this loop is polynomial with high probability.

It is important to note that the vertices considered during the course of the algorithm may not all appear on a single polytope. Rather, they may be viewed as appearing on the convex hulls of polytopes with different $\boldsymbol{b}$-vectors. It is well-known that the graph of all of these "potential" vertices has small diameter. However, there was previously no way to guide a walk among these potential vertices to one optimizing any particular objective function. Our algorithm uses the graphs of polytopes "near" $P$ to impose structure on this graph and to help to guide our walk.

Perhaps the message to take away from this is that instead of worrying about the combinatorics of the natural polytope $P$, one can reduce the linear programming problem to one whose polytope is more tractable. The first result of our paper, and the inspiration for the algorithm, captures this idea by showing that if one slightly perturbs the $\boldsymbol{b}$-vector of a polytope in near-isotropic position, then there will be a polynomial-step path from the vertex minimizing to the vertex maximizing a random objective function. Moreover, this path may be found by the shadow-vertex simplex method.

We stress that while our algorithm involves a perturbation, it is intrinsically different from previous papers that have provided average-case or smoothed analyses of linear programming. In those papers, one shows that, given some linear program, one can probably use the simplex method to solve a nearby but different linear program; the perturbation actually modified the input. In the present paper, our perturbation is used to inform the walk that we take on the (feasible or infeasible) vertices of our linear program; however, we actually solve the exact instance that we are given. We believe that ours is the first simplex algorithm to achieve this, and we hope that our results will be a useful step on the path to a strongly polynomial-time algorithm for linear programming.

We note that no effort has been made to optimize our bounds; we shall include more precise bounds in a later version of this paper.

## 2. THE SHADOW-VERTEX METHOD

Let $P$ be a convex polyhedron, and let $S$ be a two-dimensional subspace. The *shadow* of $P$ onto $S$ is simply the projection of $P$ onto $S$. The shadow is a polygon, and every vertex (edge) of the polygon is the image of some vertex (edge) of $P$. One can show that the set of vertices of $P$ that project onto the boundary of the shadow polygon are exactly the vertices of $P$ that optimize objective functions in $S$ [2, 6].

These observations are the inspiration for the shadow-vertex simplex method, which lifts the simplicity of linear programming in two dimensions to the general case [2, 6]. To start, the shadow-vertex method requires as input a vertex $\boldsymbol{v}_0$ of $P$. It then chooses some objective function optimized at $\boldsymbol{v}_0$, say $\boldsymbol{f}$, sets $S = \text{span}(\boldsymbol{c}, \boldsymbol{f})$, and considers the shadow of $P$ onto $S$. If no degeneracies occur, then for each vertex $\boldsymbol{y}$ of $P$ that projects onto the boundary of the shadow, there is a unique neighbor of $\boldsymbol{y}$ on $P$ that projects onto the next vertex of the shadow in clockwise-order. Thus, by tracing the vertices of $P$ that map to the boundary of the shadow, the shadow-vertex method can move from the vertex it knows that optimizes $\boldsymbol{f}$ to the vertex that optimizes $\boldsymbol{c}$. The number of steps that the method takes will be bounded by the number of edges of the shadow polygon. For future reference, we call the shadow-vertex simplex method by

$$\textsc{ShadowVertex}(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n, \boldsymbol{b}, \boldsymbol{c}, S, \boldsymbol{v}_0, s),$$

where $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n, \boldsymbol{b}$, and $\boldsymbol{c}$ specify a linear program of form (1), $S$ is a two-dimensional subspace containing $\boldsymbol{c}$, and $\boldsymbol{v}_0$ is the start vertex, which must optimize some objective function in $S$. We allow the method to run for at most $s$ steps. If it has not found the vertex optimizing $\boldsymbol{c}$ within that time, it should return $(\textbf{fail}, \boldsymbol{y})$, where $\boldsymbol{y}$ is its current vertex. If it has solved the linear program, it either returns $(\textbf{opt}, \boldsymbol{x})$, where $\boldsymbol{x}$ is the solution, or **unbounded** if it was unbounded.

In the next section, we will show that if the polytope is in isotropic position and the distances of the facets from the origin are randomly perturbed, then the number of edges of the shadow onto a random $S$ is expected to be polynomial. The one geometric fact that we will require in our analysis is that if an edge of $P$ is tight for inequalities $\boldsymbol{a}_i \cdot \boldsymbol{x} = b_i$, for $i \in I$, then the edge projects to an edge in the shadow if and only if $S$ intersects the convex hull of $\{\boldsymbol{a}_i\}_{i \in I}$. Below, we will often abuse notation by identifying an edge with the set of constraints $I$ for which it is tight.

## 3. THE SHADOW SIZE IN THE $k$-ROUND CASE

Definition 3.1. *We say that a polytope $P$ is $k$-round if*

$$B(\boldsymbol{0}, 1) \subseteq P \subseteq B(\boldsymbol{0}, k),$$

*where $B(\boldsymbol{0}, r)$ is the ball of radius $r$ centered at the origin.*

In this section, we will consider a polytope $P$ defined by

$$\left\{ \boldsymbol{x} | \forall i,\ \boldsymbol{a}_i^T \boldsymbol{x} \leq 1 \right\},$$

in the case that $P$ is $k$-round. Note that the condition $B(\boldsymbol{0}, 1) \subseteq P$ implies $\|\boldsymbol{a}_i\| \leq 1$.

We will then consider the polytope we get by perturbing the right-hand sides,

$$Q = \left\{ \boldsymbol{x} | \forall i,\ \boldsymbol{a}_i^T \boldsymbol{x} \leq 1 + r_i \right\},$$

where each $r_i$ is an independent exponentially distributed random variable with expectation $\lambda$. That is,

$$\Pr[r_i \geq t] = e^{-t/\lambda}$$

for all $t \geq 0$.

We will prove that the expected number of edges of the projection of $Q$ onto a random 2-plane is polynomial in $n$, $k$ and $1/\lambda$. In particular, this will imply that for a random objective function, the shortest path from the minimum vertex to the maximum vertex is expected to have a number of steps polynomial in $n$, $k$ and $1/\lambda$.

Our proof will proceed by analyzing the expected length of edges that appear on the boundary of the projection. We shall show that the total length of all such edges is expected to be bounded above. However, we shall also show that our perturbation will cause the expected length of each edge to be reasonably large. Combining these two statements will provide a bound on the expected number of edges that appear.

THEOREM 3.2. *Let $\boldsymbol{v}$ and $\boldsymbol{w}$ be uniformly random unit vectors, and let $V$ be their span. Then, the expectation over $\boldsymbol{v}$, $\boldsymbol{w}$, and the $r_i$s of the number of facets of the projection of $Q$ onto $V$ is at most*

$$\frac{12\pi k(1+\lambda\ln(ne))\sqrt{d}n}{\lambda}.$$

PROOF. We first observe that the perimeter of the shadow of $P$ onto $V$ is at most $2\pi k$. Let $r = \max_i r_i$. Then, as

$$Q \subseteq \left\{\boldsymbol{x} | \forall i,\ \boldsymbol{a}_i^T \boldsymbol{x} \leq 1 + r\right\} = (1+r)P,$$

the perimeter of the shadow of $Q$ onto $V$ is at most $2\pi k(1+r)$. As we shall show in Proposition 3.3, the expectation of $r$ is at most $\lambda\ln(ne)$, so the expected perimeter of the shadow of $Q$ on $V$ is at most $2\pi k(1+\lambda\ln(ne))$.

Now, each edge of $Q$ is determined by the subset of $d-1$ of the constraints that are tight on that edge. For each $I \in \binom{[n]}{d-1}$, let $S_I(V)$ be the event that edge $I$ appears in the shadow, and let $\ell(I)$ denote the length of that edge in the shadow. We now know

$$\begin{aligned} 2\pi k(1+\lambda\ln(ne)) &\geq \sum_{I\in\binom{[n]}{d-1}} \mathbf{E}\left[\ell(I)\right] \\ &= \sum_{I\in\binom{[n]}{d-1}} \mathbf{E}\left[\ell(I)|S_I(V)\right]\Pr\left[S_I(V)\right]. \end{aligned}$$

Below, in Lemma 3.9, we will prove that

$$\mathbf{E}\left[\ell(I)|S_I(V)\right] \geq \frac{\lambda}{6\sqrt{d}n}.$$

From this, we conclude that

$$\begin{aligned} \mathbf{E}\left[\text{number of edges}\right] &= \sum_{I\in\binom{[n]}{d-1}} \Pr\left[S_I(V)\right] \\ &\leq \frac{12\pi k(1+\lambda\ln(ne))\sqrt{d}n}{\lambda}, \end{aligned}$$

as desired. □

We now prove the various lemmas used in the proof of Theorem 3.2. Our first is a straightforward statement about exponential random variables.

PROPOSITION 3.3. *Let $r_1,\ldots,r_n$ be independent exponentially distributed random variables of expectation $\lambda$. Then,*

$$\mathbf{E}\left[\max r_i\right] \leq \lambda\ln(ne).$$

PROOF. This follows by a simple calculation, in which the first inequality follows from a union bound:

$$\begin{aligned} \mathbf{E}\left[\max r_i\right] &= \int_{t=0}^{\infty} \Pr\left[\max r_i \geq t\right] \\ &\leq \int_{t=0}^{\infty} \Pr\left[\min(1, ne^{-t/\lambda})\right] \\ &= \int_{t=0}^{\lambda\ln n} 1 + \int_{\lambda\ln n}^{\infty} ne^{-t/\lambda} \\ &= (\lambda\ln n) + \lambda \\ &= \lambda\ln(ne), \end{aligned}$$

as desired. □

We shall now prove the lemmas necessary for Lemma 3.9, which bounds the expected length of an edge, given that it appears in the shadow. Our proof of Lemma 3.9 will have two parts. In Lemma 3.7, we will show that it is unlikely that the edge indexed by $I$ is short, given that it appears on the convex hull of $Q$. We will then use Lemma 3.8 to show that, given that it appears in the shadow, it is unlikely that its projection onto the shadow plane is much shorter. To facilitate the proofs of these lemmas, we shall prove some auxiliary lemmas about *shifted exponential random variables*.

DEFINITION 3.4. *We say that $r$ is a shifted exponential random variable with parameter $\lambda$ if there exists a $t \in \mathbb{R}$ such that $r = s - t$, where $s$ is an exponential random variable with expectation $\lambda$.*

PROPOSITION 3.5. *Let $r$ be a shifted exponential random variable of parameter $\lambda$. Then, for all $q \in \mathbb{R}$ and $\epsilon \geq 0$,*

$$\Pr\left[r \leq q+\epsilon \middle| r \geq q\right] \leq \epsilon/\lambda.$$

PROOF. As $r-q$ is a shifted exponential random variable, it suffices to consider the case in which $q = 0$. So, assume $q = 0$ and $r = s - t$, where $s$ is an exponential random variable of expectation $\lambda$. We now need to compute

$$\Pr\left[s \leq t+\epsilon \middle| s \geq t\right]. \tag{2}$$

We only need to consider the case $\epsilon < \lambda$, as the proposition is trivially true otherwise. We first consider the case in which $t \geq 0$. In this case, we have

$$\begin{aligned} (2) = \Pr\left[s \leq t+\epsilon \middle| s \geq t\right] &= \frac{\frac{1}{\lambda}\int_{s=t}^{t+\epsilon} e^{-s/\lambda} ds}{\frac{1}{\lambda}\int_{s=t}^{\infty} (1/\lambda) e^{-s/\lambda} ds} \\ &= \frac{e^{-t/\lambda} - e^{-t/\lambda+\epsilon/\lambda}}{e^{-t/\lambda}} \\ &= 1 - e^{\epsilon/\lambda} \leq \epsilon/\lambda, \end{aligned}$$

for $\epsilon/\lambda \leq 1$.

Finally, the case when $t \leq 0$ follows from the analysis in the case $t = 0$. □

LEMMA 3.6. *For $N$ and $P$ disjoint subsets of $\{1,\ldots,n\}$, let $\{r_i\}_{i\in P}$ and $\{r_j\}_{j\in N}$ be independent random variables, each of which is a shifted exponential random variable with parameter at least $\lambda$. Then*

$$\Pr\left[\min_{i\in P}(r_i) + \min_{j\in N}(r_j) < \epsilon \middle| \min_{i\in P}(r_i) + \min_{j\in N}(r_j) \geq 0\right] \leq n\epsilon/2\lambda.$$

PROOF. Assume without loss of generality that $|P| \leq |N|$, so $|P| \leq n/2$.

Set $r^+ = \min_{i \in P} r_i$ and $r^- = \min_{j \in N} r_j$. Sample $r^-$ according to the distribution induced by the requirement that $r^+ + r^- \geq 0$. Given the sampled value for $r^-$, the induced distribution on $r^+$ is simply the base distribution restricted to the space where $r^+ \geq -r^-$. So, it suffices to bound

$$\begin{aligned}
&\max_{r^-} \Pr_{r^+} \left[ r^+ < \epsilon - r^- \middle| r^+ \geq -r^- \right] \\
&= \max_{r^-} \Pr_{r^+} \left[ \min_{i \in P}(r_i) < \epsilon - r^- \middle| \min_{i \in P}(r_i) \geq -r^- \right] \\
&\leq \max_{r^-} \sum_{i \in P} \Pr_{r^+} \left[ r_i < \epsilon - r^- \middle| \min_{i \in P}(r_i) \geq -r^- \right] \\
&= \sum_{i \in P} \max_{r^-} \Pr_{r^+} \left[ r_i < \epsilon - r^- \middle| \min_{i \in P}(r_i) \geq -r^- \right] \\
&= \sum_{i \in P} \max_{r^-} \Pr_{r^+} \left[ r_i < \epsilon - r^- \middle| r_i \geq -r^- \right] \\
&\leq |P| \, (\epsilon/\lambda),
\end{aligned}$$

where the last equality follows from the independence of the $r_i$'s, and the last inequality follows from Proposition 3.5. □

LEMMA 3.7. *Let $I \in \binom{[n]}{d-1}$, and let $A(I)$ be the event that $I$ appears on the convex hull of $Q$. Let $\delta(I)$ denote the length of the edge $I$ on $Q$. Then,*

$$\Pr\left[\delta(I) < \epsilon | A(I)\right] \leq \frac{n\epsilon}{2\lambda}.$$

PROOF. Without loss of generality, we set $I = \{1, \ldots, d-1\}$. As our proof will not depend upon the values of $r_1, \ldots, r_{d-1}$, assume that they have been set arbitrarily. Now, parameterize the line of points satisfying

$$\boldsymbol{a}_i^T \boldsymbol{x} = 1 + r_i, \quad \text{for } i \in I,$$

by

$$l(t) := \boldsymbol{p} + t\boldsymbol{q},$$

where $\boldsymbol{p}$ is the point on the line closest to the origin, and $\boldsymbol{q}$ is a unit vector orthogonal to $\boldsymbol{p}$. For each $i \geq d$, let $t_i$ index the point where the $i^{\text{th}}$ constraint intersects the line, i.e.,

$$\boldsymbol{a}_i^T l(t_i) = 1 + r_i. \tag{3}$$

Now, divide the constraints indexed by $i \notin I$ into a positive set, $P = \left\{ i \geq d | \boldsymbol{a}_i^T \boldsymbol{q} \geq 0 \right\}$, and a negative set $N = \left\{ i \geq d | \boldsymbol{a}_i^T \boldsymbol{q} < 0 \right\}$. Note that each constraint in the positive set is satisfied by $l(-\infty)$ and each constraint in the negative set is satisfied by $l(\infty)$. The edge $I$ appears in the convex hull if and only if for each $i \in P$ and $j \in N$, $t_j < t_i$. When the edge $I$ appears, its length is

$$\min_{i \in P,\, j \in N} t_i - t_j.$$

Solving (3) for $i \in P$, we find $t_i = \frac{1}{\boldsymbol{a}_i^T \boldsymbol{q}} \left(1 - \boldsymbol{a}_i^T \boldsymbol{p} + r_i\right)$. Similarly, for $j \in N$, we find $t_j = \frac{1}{|\boldsymbol{a}_j^T \boldsymbol{q}|} \left(-1 + \boldsymbol{a}_j^T \boldsymbol{p} - r_j\right)$. Thus, $t_i$ for $i \in P$ and $-t_j$ for $j \in N$ are both shifted exponential random variables with parameter at least $\lambda$. So, by Lemma 3.6,

$$\Pr_{\{r_i | i \notin I\}} \left[ \min_{i \in P,\, j \in N} t_i - t_j < \epsilon | A(I) \right] < n\epsilon/2\lambda. \quad \square$$

LEMMA 3.8. *Let $Q$ be an arbitrary polytope, and let $I$ index an edge of $Q$. Let $\boldsymbol{v}$ and $\boldsymbol{w}$ be random unit vectors, and let $V$ be their span. Let $S_I(V)$ be the event that the edge $I$ appears on the convex hull of the projection of $Q$ onto $V$. Let $\theta_I(V)$ denote the angle of the edge $I$ to $V$. Then*

$$\Pr_{\boldsymbol{v}, \boldsymbol{w}} \left[ \cos(\theta_I(V)) < \epsilon | S_I(V) \right] \leq d\epsilon^2.$$

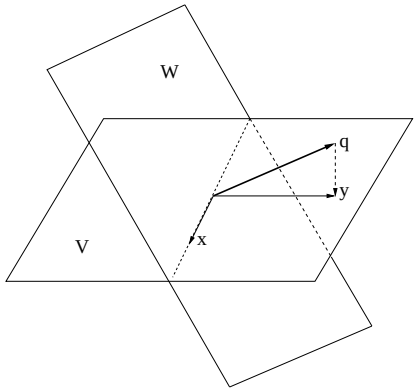


**Figure 1: The points $x$, $y$ and $q$.**

PROOF. As in the proof of Lemma 3.7, parameterize the edge by

$$l(t) := \boldsymbol{p} + t\boldsymbol{q},$$

where $\boldsymbol{q}$ is a unit vector. Observe that $S_I(V)$ holds if and only if $V$ non-trivially intersects the cone $\left\{ \sum_{i \in I} \alpha_i \boldsymbol{a}_i | \alpha_i \geq 0 \right\}$, which we denote $C$. To evaluate the probability, we will perform a change of variables that will both enable us to easily evaluate the angle between $\boldsymbol{q}$ and $V$ and to determine whether $S_I(V)$ holds. Some of the new variables that we introduce are shown in Figure 1.

First, let $W$ be the span of $\{\boldsymbol{a}_i | i \in I\}$, and note that $W$ is also the subspace orthogonal to $\boldsymbol{q}$. The angle of $\boldsymbol{q}$ to $V$ is determined by the angle of $\boldsymbol{q}$ to the unit vector through the projection of $\boldsymbol{q}$ onto $V$, which we will call $\boldsymbol{y}$. Fix any vector $\boldsymbol{c} \in C$, and let $\boldsymbol{x}$ be the unique unit vector in $V$ that is orthogonal to $\boldsymbol{y}$ and has positive inner product with $\boldsymbol{c}$. Note that $\boldsymbol{x}$ is also orthogonal to $\boldsymbol{q}$, and so $\boldsymbol{x} \in V \cap W$. Also note that $S_I(V)$ holds if and only if $\boldsymbol{x} \in C$.

Instead of expressing $V$ as the span of $\boldsymbol{v}$ and $\boldsymbol{w}$, we will express it as the span of $\boldsymbol{x}$ and $\boldsymbol{y}$, which are much more useful vectors. In particular, we need to express $\boldsymbol{v}$ and $\boldsymbol{w}$ in terms of $\boldsymbol{x}$ and $\boldsymbol{y}$, which we do by introducing two more variables, $\alpha$ and $\beta$, so that

$$\begin{aligned}
\boldsymbol{v} &= \boldsymbol{x} \cos\alpha + \boldsymbol{y} \sin\alpha, \text{and} \\
\boldsymbol{w} &= \boldsymbol{x} \cos\beta + \boldsymbol{y} \sin\beta.
\end{aligned}$$

Note that number of degrees of freedom has not changed: $\boldsymbol{v}$ and $\boldsymbol{w}$ each had $d-1$ degrees of freedom, while $\boldsymbol{x}$ only has $d-2$ degrees of freedom since it is restricted to be orthogonal to $\boldsymbol{q}$, and given $\boldsymbol{x}$, $\boldsymbol{y}$ only has $d-2$ degrees of freedom since it is restricted to be orthogonal to $\boldsymbol{x}$.

We now make one more change of variables so that the angle between $\boldsymbol{q}$ and $\boldsymbol{y}$ becomes a variable. To do this, we let $\theta = \theta_I(V)$ be the angle between $\boldsymbol{y}$ and $\boldsymbol{q}$, and note that once $\theta$ and $\boldsymbol{x}$ have been specified, $\boldsymbol{y}$ is constrained to lie on

a $d-2$ dimensional sphere. We let $\boldsymbol{z}$ denote the particular point on that sphere.

Deshpande and Spielman [4, Full version] prove that the Jacobian of this change of variables from $\alpha$, $\beta$, $\boldsymbol{x}$, $\theta$, $\boldsymbol{z}$ to $\boldsymbol{v}$ and $\boldsymbol{w}$ is

$$c(\cos\theta)(\sin\theta)^{d-3}\sin(\alpha-\beta)^{d-2},$$

where $c$ is a constant depending only on the dimension.

Now, to compute the probability, we will fix $\alpha$ and $\beta$ arbitrarily and integrate over $\boldsymbol{x} \in C$.

$$\begin{aligned}
&\Pr_V\left[\cos(\theta_I(V)) < \epsilon | S_I(V)\right] \\
&= \frac{\int_{\boldsymbol{v},\boldsymbol{w}\in S^{n-1}:V\cap C\neq\emptyset \text{ and } \theta_I(V)\leq\epsilon} 1\, d\boldsymbol{v}\; d\boldsymbol{w}}{\int_{\boldsymbol{v},\boldsymbol{w}\in S^{n-1}:\mathrm{Span}(\boldsymbol{v},\boldsymbol{w})\cap C\neq\emptyset} 1\, d\boldsymbol{v}\; d\boldsymbol{w}} \\
&= \frac{\int_{\boldsymbol{x}\in C,\boldsymbol{z}}\int_{\theta>\arccos(\epsilon)} c(\cos\theta)(\sin\theta)^{d-3}\, d\boldsymbol{x}\; d\boldsymbol{z}\; d\theta}{\int_{\boldsymbol{x}\in C,\boldsymbol{z},\theta} c(\cos\theta)(\sin\theta)^{d-3}\, d\boldsymbol{x}\; d\boldsymbol{z}\; d\theta} \\
&= \frac{\int_{\theta=\arccos(\epsilon)}^{\pi/2}(\cos\theta)(\sin\theta)^{d-3}\, d\theta}{\int_{\theta=0}^{\pi/2}(\cos\theta)(\sin\theta)^{d-3}\, d\theta} \\
&= \frac{(\sin\theta)^{d-2}\big|_{\arccos(\epsilon)}^{\pi/2}}{(\sin\theta)^{d-2}\big|_0^{\pi/2}} \\
&\leq 1-(\sin(\arccos(\epsilon)))^{d-2} \\
&\leq 1-(1-\epsilon^2)^{d-2} \;\leq\; (d-2)\epsilon^2. \quad \square
\end{aligned}$$

LEMMA 3.9. *For all $I \in \binom{[n]}{d-1}$, $\mathbf{E}_{V,r_1,\ldots,r_n}\left[\ell(I)|S_I(V)\right] \geq \frac{\lambda}{6\sqrt{dn}}$.*

PROOF. For each edge $I$, $\ell(I) = \delta(I)\cos(\theta_I(V))$. By Lemma 3.7,

$$\Pr\left[\delta(I) \geq \frac{\lambda}{n} \;\middle|\; A(I)\right] \geq 1/2.$$

By Lemma 3.8,

$$\Pr_V\left[cos(\theta_I(V)) \geq 1/\sqrt{2d} \;\middle|\; S_I(V)\right] \geq 1/2.$$

So, given that edge $I$ appears on the shadow, $\ell(I) > (1/\sqrt{2d})\left(\frac{\lambda}{n}\right)$ with probability at least 1/4. Thus, its expected length when it appears is at least $\frac{\lambda}{6\sqrt{dn}}$. $\square$

# 4. THE SHADOW SIZE IN THE GENERAL CASE

In this section, we present an extension of Theorem 3.2 that we will require in the analysis of our simplex algorithm. We extend the theorem in two ways. First of all, we examine what happens when $P$ is not near isotropic position. In this case, we just show that the shadow of the convex hull of the vertices of bounded norm probably has few edges. As such, if we take a polynomial number of steps around the shadow, we should either come back to where we started or find a vertex far from the origin. Secondly, we consider the shadow onto random planes that come close to a particular vector, rather than just onto uniformly random planes.

DEFINITION 4.1. *For a unit vector $\boldsymbol{u}$ and a $\rho > 0$, we define the $\rho$-perturbation of $\boldsymbol{u}$ to be the random unit vector $\boldsymbol{v}$ chosen by*

1. *choosing a $\theta \in [0,\pi]$ according to the restriction of the exponential distribution of expectation $\rho$ to the range $[0,\pi]$, and*
2. *setting $\boldsymbol{v}$ to be a uniformly chosen unit vector of angle $\theta$ to $\boldsymbol{u}$.*

THEOREM 4.2. *Let $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n$ be vectors of norm at most 1. Let $r_1,\ldots,r_n$ be independent exponentially distributed random variables with expectation $\lambda$. Let $Q$ be the polytope given by*

$$Q = \left\{\boldsymbol{x} | \forall i,\ \boldsymbol{a}_i^T\boldsymbol{x} \leq 1 + r_i\right\}.$$

*Let $\boldsymbol{u}$ be an arbitrary unit vector, $\rho < 1/\sqrt{d}$, and let $\boldsymbol{v}$ be a random $\rho$ perturbation of $\boldsymbol{u}$. Let $\boldsymbol{w}$ be a uniformly chosen random unit vector. Then, for all $t > 1$,*

$$\begin{aligned}
&\mathbf{E}_{r_1,\ldots,r_n,\boldsymbol{v},\boldsymbol{w}}\left[\mathrm{ShadowSize}_{\mathrm{span}(\boldsymbol{v},\boldsymbol{w})}(Q\cap B(\boldsymbol{0},t))\right] \\
&\qquad\qquad \leq \frac{42\pi t(1+\lambda\log n)\sqrt{d}n}{\lambda\rho}.
\end{aligned}$$

PROOF. The proof of Theorem 4.2 is almost identical to that of Theorem 3.2, except that we substitute Lemma 4.3 for Lemma 3.7, and we substitute Lemma 4.4 for Lemma 3.8. $\square$

LEMMA 4.3. *For $I \subseteq \binom{[n]}{d-1}$ and $t > 0$,*

$$\Pr\left[\delta(I) < \epsilon \middle| A(I) \text{ and } I\cap B(\boldsymbol{0},t) \neq \emptyset\right] \leq \frac{n\epsilon}{2\lambda}.$$

PROOF. The proof is identical to the proof of Lemma 3.7, except that in the proof of Lemma 3.6 we must condition upon the events that

$$r^+ \geq -\sqrt{t-\|\boldsymbol{p}\|} \quad \text{and} \quad r^- \leq \sqrt{t-\|\boldsymbol{p}\|}.$$

These conditions have no impact on any part of the proof. $\square$

LEMMA 4.4. *Let $Q$ be an arbitrary polytope, and let $I$ index an edge of $Q$. Let $\boldsymbol{u}$ be any unit vector, let $\rho < 1/\sqrt{d}$, and let $\boldsymbol{v}$ be a random $\rho$ perturbation of $\boldsymbol{u}$. Let $\boldsymbol{w}$ be a uniformly chosen random unit vector, and let $V = \mathrm{span}(\boldsymbol{u},\boldsymbol{v})$. Then*

$$\Pr_{\boldsymbol{v},\boldsymbol{w}}\left[\cos(\theta_I(V)) < \epsilon | S_I(V)\right] \leq 3.5\epsilon^2/\rho^2.$$

PROOF. We perform the same change of variables as in Lemma 3.8.

To bound the probability that $\cos\theta < \epsilon$, we will allow the variables $\boldsymbol{x}$, $\boldsymbol{z}$, $\alpha$ and $\beta$ to be fixed arbitrarily, and just consider what happens as we vary $\theta$. To facilitate writing the resulting probability, let $\mu$ denote the density function on $\boldsymbol{v}$. If we fix $\boldsymbol{x}$, $\boldsymbol{z}$, $\alpha$ and $\beta$, then we can write $\boldsymbol{v}$ as a function of $\theta$. Moreover, as we vary $\theta$ by $\phi$, $\boldsymbol{v}$ moves through an angle of at most $\phi$. So, for all $\phi < \rho$ and $\theta$,

$$\mu(\boldsymbol{v}(\theta)) < \mu(\boldsymbol{v}(\theta+\phi))/e. \tag{4}$$

With this fact in mind, we compute the probability to be

$$\begin{aligned}
&\frac{\int_{\boldsymbol{v},\boldsymbol{w}\in S^{n-1}:V\cap C\neq\emptyset \text{ and } \theta_I(V)\leq\epsilon} \mu(\boldsymbol{v})\,d\boldsymbol{v}\;d\boldsymbol{w}}{\int_{\boldsymbol{v},\boldsymbol{w}\in S^{n-1}:V\cap C\neq\emptyset} \mu(\boldsymbol{v})\,d\boldsymbol{v}\;d\boldsymbol{w}} \\
&= \frac{\int_{\theta=\arccos(\epsilon)}^{\pi/2}(\cos\theta)(\sin\theta)^{d-3}\mu(\boldsymbol{v}(\theta))\,d\theta}{\int_{\theta=0}^{\pi/2}(\cos\theta)(\sin\theta)^{d-3}\mu(\boldsymbol{v}(\theta))\,d\theta} \\
&\leq \frac{\int_{\theta=\arccos(\epsilon)}^{\pi/2}(\cos\theta)(\sin\theta)^{d-3}\mu(\theta)\,d\theta}{\int_{\theta=\pi/2-\rho}^{\pi/2}(\cos\theta)(\sin\theta)^{d-3}\mu(\theta)\,d\theta} \\
&\leq \frac{e\int_{\theta=\arccos(\epsilon)}^{\pi/2}(\cos\theta)(\sin\theta)^{d-3}\,d\theta}{\int_{\theta=\pi/2-\rho}^{\pi/2}(\cos\theta)(\sin\theta)^{d-3}\,d\theta}, \text{by (4)} \\
&= e\frac{(\sin\theta)^{d-2}\big|_{\arccos(\epsilon)}^{\pi/2}}{(\sin\theta)^{d-2}\big|_{\pi/2-\rho}^{\pi/2}} \\
&= e\frac{1-(\sin(\arccos(\epsilon))^{d-2}}{1-(\sin(\rho)^{d-2}} \\
&\leq e\frac{1-(1-\epsilon^2)^{d-2}}{1-(1-\rho^2/2)^{d-2}} \\
&\leq e\frac{(d-2)\epsilon^2}{(d-2)2(1-1/\sqrt{e})\rho^2}, \text{ as } \rho<1/\sqrt{d}, \\
&\leq 3.5(\epsilon/\rho)^2. \quad \square
\end{aligned}$$

# 5. REDUCTION OF LINEAR PROGRAMMING TO CERTIFYING BOUNDEDNESS

We now recall an old trick [12, p. 125] for reducing the problem of solving a linear program in form (1) to a different form that will be more useful for our purposes. We recall that the dual of such a linear program is given by

$$\begin{aligned}
\textbf{minimize} \quad & \boldsymbol{b}\cdot\boldsymbol{y} \\
\textbf{subject to} \quad & A^T\boldsymbol{y}=\boldsymbol{c},\ \boldsymbol{y}\geq 0,
\end{aligned} \tag{5}$$

and that when the programs are both feasible and bounded, they have the same solution. Thus, any feasible solution to the system of constraints

$$\begin{aligned}
A\boldsymbol{x}\leq\boldsymbol{b},\ \boldsymbol{x}\in\mathbb{R}^d, \\
A^T\boldsymbol{y}=\boldsymbol{c},\ \boldsymbol{y}\geq\boldsymbol{0}, \\
\boldsymbol{c}\cdot\boldsymbol{x}=\boldsymbol{b}\cdot\boldsymbol{y}
\end{aligned} \tag{6}$$

provides a solution to both the linear program and its dual. Using standard techniques, one can reduce the solution of a feasibility problem in this form to a feasibility problem of the form

$$\begin{aligned}
A_1^T\boldsymbol{z}=\boldsymbol{0} \\
\boldsymbol{z}\geq\boldsymbol{0},\ \boldsymbol{z}\neq\boldsymbol{0},
\end{aligned} \tag{7}$$

where $A_1$ is a matrix constructed from $A$, $\boldsymbol{b}$ and $\boldsymbol{c}$. When the system (7) is non-degenerate, a solution to the system is equivalent to a certificate that a system of form

$$A_1\boldsymbol{w}\leq\boldsymbol{b}_1 \tag{8}$$

is bounded, where the choice of the vector $\boldsymbol{b}_1$ does not matter as it does not affect boundedness. However, our reduction produces a system that is degenerate. This is easily remedied, in strongly-polynomial time, by applying the $\epsilon$-perturbation technique of Megiddo and Chandrasekaran [11], which produces a non-degenerate system that is solvable if and only if the original is, and from whose solution one can obtain the solution to the original[1]. By solving this system with a randomly chosen right-hand side vector we can solve system (1) while avoiding the combinatorial complications of the feasible set of (1).

In our algorithm, we will certify boundedness of (1) by finding the vertices minimizing and maximizing some objective function. Provided that the system is non-degenerate, which it is with high probability under our choice of right-hand sides, this can be converted into a solution to (7).

# 6. OUR ALGORITHM

Our bound from Theorem 3.2 suggests a natural algorithm for certifying the boundedness of a linear program of the form given in (8): set each $b_i$ to be $1+r_i$, where $r_i$ is an exponential random variable, pick a random objective function $c$ and a random two-dimensional subspace containing it, and then use the shadow-vertex method with the given subspace to maximize and minimize $c$.

In order to make this approach into a polynomial-time algorithm, there are two difficulties that we must surmount:

1. To use the shadow-vertex method, we need to start with some vertex that appears on the boundary of the shadow. If we just pick an arbitrary shadow plane, there is no obvious way to find such a vertex.

2. Theorem 3.2 bounds the expected shadow size of the vertices of bounded norm in polytopes with perturbed right-hand sides, whereas the polytope that we are given may have vertices of exponentially large norm. If we naively choose our perturbations, objective function, and shadow plane as if we were in a coordinate system in which all of our vertices had bounded norm, the distribution of vertices that appear on the shadow may be very different, and we have no guarantees about the expected shadow size.

We address the first difficulty by constructing an artificial vertex at which to start our simplex algorithm. To address the second difficulty, we start out by choosing our random variables from the naive distributions. If this doesn't work, we iteratively use information about how it failed to improve the probability distributions from which we sample and try again.

## 6.1 Constructing a Starting Vertex

In order to use the shadow-vertex method on a polytope $P$, we need a shadow plane $\boldsymbol{S}$ and a vertex $\boldsymbol{v}$ that appears on the boundary of the shadow. One way to obtain such a pair is to pick any vertex $\boldsymbol{v}$, randomly choose (from some probability distribution) an objective function $\boldsymbol{c}$ optimized by $\boldsymbol{v}$, let $\boldsymbol{u}$ be a uniformly random unit vector, and set $\boldsymbol{S}=\text{span}(\boldsymbol{c},\boldsymbol{u})$.

However, to apply the bound on the shadow size given by Theorem 4.2, we need to choose $\boldsymbol{c}$ to be a $\rho$-perturbation of some vector. For such a $\boldsymbol{c}$ to be likely to be optimized

[1]If one views the problem as asking if the origin is contained inside the convex hull of the rows of $A_1$, then the perturbation pushes the origin slightly towards the average of the rows. The magnitude of the perturbation is small enough that it cannot make an infeasible system feasible, but it only increases the size of the input by a polynomial factor in $n$ and $d$.

by $\boldsymbol{v}$, we need $\boldsymbol{v}$ to optimize a reasonably large ball of objective functions. To guarantee that we can find such a $\boldsymbol{v}$, we create one. That is, we add constraints to our polytope to explicitly construct an artificial vertex with the desired properties. (This is similar to the "Phase I" approaches that have appeared in some other simplex algorithms.)

Suppose for now that the polytope $\{\boldsymbol{x}|A\boldsymbol{x} \leq \mathbf{1}\}$ is $k$-round. Construct a modified polytope $P'$ by adding $d$ new constraints, $\left\{\overline{\boldsymbol{w}}_i^T \boldsymbol{x} \leq 1, i = 1, \ldots, d\right\}$, where

$$\boldsymbol{w}_i = -\Big(\sum_j \boldsymbol{e}_j\Big) + \sqrt{d}\boldsymbol{e}_i/3k^2,$$

and $\overline{\boldsymbol{w}}_i = \boldsymbol{w}_i/(2\,||\boldsymbol{w}_i||)$. Let $\boldsymbol{x}_0$ be the vertex at which $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_d$ are all tight. Furthermore, let $\boldsymbol{c}$ be a $\rho$-perturbation of the vector $\mathbf{1}/\sqrt{d}$, with $\rho = 1/6dk^2$, and let $\boldsymbol{x}_1$ be the vertex at which $\boldsymbol{c}$ is maximized. We can prove:

LEMMA 6.1. *The following three properties hold with high probability. Furthermore, they remain true with probability* $1 - (d+2)e^{-n}$ *if we perturb all of the right-hand sides of the constraints in* $P'$ *by an exponential random variable of expectation* $\lambda = 1/n$.

1. *The vertex* $\boldsymbol{x}_0$ *appears on* $P'$,
2. $-\boldsymbol{c}$ *is maximized at* $\boldsymbol{x}_0$, *and*
3. *None of the constraints* $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_d$ *is tight at* $\boldsymbol{x}_1$.

PROOF. Follows from Lemma 7.1 and bounds on tails of exponential random variables. □

Set

$$k := 16d + 1 \quad \text{and} \quad s := 4 \cdot 10^7\ d^{9/2}n.$$

Let $\boldsymbol{S} = \text{span}(\boldsymbol{c}, \boldsymbol{u})$, where $u$ is a uniform random unit vector. If $P$ is $k$-round, then by Lemma 6.1 and Theorem 3.2 we can run the shadow vertex method on $P'$ with shadow plane $\boldsymbol{S}$ and starting at vertex $\boldsymbol{x}_0$, and we will find the vertex $\boldsymbol{x}_1$ that maximizes $\boldsymbol{c}$ within $s$ steps, with probability at least 1/2. Since none of the $\boldsymbol{w}_i$ are tight at $\boldsymbol{x}_1$, $\boldsymbol{x}_1$ will also be the vertex of the original polytope $P$ that maximizes $\boldsymbol{c}$.

This gives us the vertex $\boldsymbol{x}_1$ of $P$ that maximizes $\boldsymbol{c}$. We can now run the shadow vertex method again on $P$ using the same shadow plane. This time, we *start* at $\boldsymbol{x}_1$ and find the vertex that maximizes $-\boldsymbol{c}$. We are again guaranteed to have an expected polynomial-sized shadow, so this will again succeed with high probability. This will give us a pair of vertices that optimize $\boldsymbol{c}$ and $-\boldsymbol{c}$, from which we can compute our desired certificate of boundedness. It just remains to deal with polytopes that are not near isotropic position.

## 6.2 Polytopes Far from Isotropic Position

We first observe that for every polytope there exists an affine change of coordinates (i.e., a translation composed with a change of basis) that makes it $k$-round. An affine change of coordinates does not change the combinatorial structure of a polytope, so this means that there exists *some* probability distribution on $\boldsymbol{b}$ and $\boldsymbol{S}$ for which the shadow has polynomial expected size. We would like to sample $\boldsymbol{b}$ and $\boldsymbol{S}$ from these probability distributions and then pull the result back along the change of coordinates. Unfortunately, we don't know an affine transformation that makes our polytope $k$-round, so we are unable to sample from these distributions.

**Algorithm 6.1:** CHECKBOUNDEDNESS$(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n)$

**Require** each $\boldsymbol{a}_i$ has norm at most 1.
Set $k = 16d + 1$, $\lambda = 1/n$, $\rho = 1/6dk^2$
  $s = 4 \cdot 10^7\ d^{9/2}n$, and $\overline{\boldsymbol{w}}_i$ as described in text;
Initialize $\boldsymbol{Q} := \text{Id}_n$, $\boldsymbol{r} := \mathbf{0}$;
**Repeat** until you return an answer
  Construct constraints for starting corner:
    $\boldsymbol{a}_{n+i} := \boldsymbol{Q}^T\overline{\boldsymbol{w}_i}/(1 - \overline{\boldsymbol{w}_i} \cdot (\boldsymbol{Q}\boldsymbol{r}))$ for $i = 1, \ldots, d$;
  $b_i := (1 + \beta_i)(1 + \boldsymbol{a}_i^T\boldsymbol{r})$ for $i = 1, \ldots n + d$, (1)
    $\beta_i$ exponential random vars with expectation $\lambda$;
  Set starting corner $\boldsymbol{x}_0 :=$ point where $\boldsymbol{a}_i^T\boldsymbol{x}_0 = b_i$
    for $i = n + 1, \ldots, n + d$;
  **If** $\boldsymbol{x}_0$ violates $\boldsymbol{a}_i^T\boldsymbol{x}_0 \leq b_i$ for any $i$, go back to (1)
    and generate new random variables;
  $c := \boldsymbol{Q}^T\gamma$, with $\gamma$ a $\rho$-perturbation of $\mathbf{1}/\sqrt{d}$;
  Shadow plane $\boldsymbol{S} := \text{span}(\boldsymbol{c}, \boldsymbol{Q}^T\boldsymbol{u})$,
    with $\boldsymbol{u}$ a uniformly random unit vector;
  Run SHADOWVERTEX$((\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{n+d}), \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{S}, \boldsymbol{x}_0, s)$ :
    **If** returns **unbounded then**
      **return** (**unbounded**);
    **If** returns (**fail**, $\boldsymbol{y}_0$) **then**
      set $\boldsymbol{y} := \boldsymbol{y}_0$ and go to (3);
    **If** returns (**opt**, $\boldsymbol{v}_0$) **then**
      set $\boldsymbol{v} := \boldsymbol{v}_0$ and continue to (2);
  Run SHADOWVERTEX$((\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n), \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{S}, \boldsymbol{v}, s)$ : (2)
    **If** returns **unbounded then**
      **return** (**unbounded**);
    **If** returns (**fail**, $\boldsymbol{y}_0$) **then**
      set $\boldsymbol{y} := \boldsymbol{y}_0$ and go to (3);
    **If** returns (**opt**, $\boldsymbol{v}_0$) **then**
      set $\boldsymbol{v}' := \boldsymbol{v}_0$ and **return** $(\boldsymbol{v}, \boldsymbol{v}')$;
  Update $\boldsymbol{Q}$ and $\boldsymbol{r}$: (3)
    **If** $||\boldsymbol{Q}(\boldsymbol{y} + \boldsymbol{r})|| \leq 2k$ **then**
      don't change $\boldsymbol{Q}$ or $\boldsymbol{r}$
    **else**
      Set $\boldsymbol{M} :=$ the matrix that scales down
        $Q(\boldsymbol{y} + \boldsymbol{r})$ by factor of 8 and scales
        vectors in orthogonal complement up
        by factor of $1 - 1/d$;
      $\boldsymbol{Q} := \boldsymbol{M}\,\boldsymbol{Q}$;
      $\boldsymbol{r} := \boldsymbol{r} + 7Q(\boldsymbol{y} + \boldsymbol{r})/||Q(\boldsymbol{y} + \boldsymbol{r})||$;

Instead, we shall start out as we would in the $k$-round case, adding in artificial constraints $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_d$, and choosing an objective function and shadow plane as in Section 6.1. By Theorem 4.2, running the shadow-vertex method for $s$ steps will yield one of two results with probability at least 1/2:

1. It will find the optimal vertex $\boldsymbol{x}_1$, or
2. It will find a vertex $\boldsymbol{y}$ of norm at least $2k$.

In the first case, we can proceed just as in the $k$-round case and run the shadow-vertex method a second time to optimize $-\boldsymbol{c}$, for which we will have the same two cases.

In the second case, we have not found the optimal vertex, but we have with high probability learned a point of large norm inside our polytope. We can use this point to change the probability distributions from which we draw our random variables and then start over. This changes our randomized pivot rule on the graph of potential vertices of our polytope, hopefully putting more probability mass on

short paths from the starting vertex to the optimum. We shall show that, with high probability, we need only repeat this process a polynomial number of times before we find a right-hand side and shadow plane for which the shadow-vertex method finds the optimum.

Our analysis rest upon the following geometric lemma, proved in Section 7:

LEMMA 6.2. *Let $B \subseteq \mathbb{R}^d$ be the unit ball, let $P$ be a point at distance $S$ from the origin, and let $C = \text{conv}(B, P)$ be their convex hull. If $S \geq 16d+1$, then $C$ contains an ellipse of volume at least twice that of $B$, having $d-1$ semi-axes[2] of length $1-1/d$ and one semi-axis of length at least 8 centered at the point of distance 7 from the origin in the direction of $P$.*

We remark that the number of times that we have to change probability distributions depends on the bit-length of the inputs, and that this is the only part of our algorithm in which this is a factor. Otherwise, the execution of our algorithm is totally independent of the bit-length of the inputs.

THEOREM 6.3. *If each entry of the vectors $\boldsymbol{a}_i$ is specified using $L$ bits, then* CHECKBOUNDEDNESS() *either produces a certificate that its input is bounded or that it is unbounded within $O(n^3L)$ iterations, with high probability.*

PROOF. It will be helpful to think of the input to CHECKBOUNDEDNESS() as being the polytope $\{\boldsymbol{x} \,|\, \boldsymbol{a}_i^T x \leq 1 \,\forall i\}$ instead of just the vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$. We can then talk about running this algorithm on an arbitrary polytope $\{\boldsymbol{x} \,|\, \alpha_i^T x \leq \tau_i \,\forall i\}$ by rewriting this polytope as $\{\boldsymbol{x} \,|\, (\alpha_i/\tau_i)^T x \leq 1 \,\forall i\}$.

With this notation, it is easy to check that running an iteration of the **Repeat** loop on a polytope $P$ with $\boldsymbol{Q} = \boldsymbol{Q}_0$ and $\boldsymbol{r} = \boldsymbol{r}_0$ is equivalent to running the same code on the polytope $\boldsymbol{Q}_0(P + \boldsymbol{r}_0)$ with $\boldsymbol{Q} = \text{Id}$ and $\boldsymbol{r} = 0$. The update step at the end of the algorithm can therefore be thought of as applying an affine change of coordinates to the input and then restarting the algorithm.

If $\boldsymbol{Q} = \text{Id}_n$ and $\boldsymbol{r} = 0$, the argument from Section 6.1 proves that the first iteration of the **Repeat** loop will either prove boundedness, prove unboundedness, or find a point with norm at least $k$ with probability at least $1/2$. In either of the first two cases, the algorithm will have succeeded, so it suffices to consider the third.

If a point $\boldsymbol{y}$ is in the polytope $P' = \{\boldsymbol{x} \,|\, A\boldsymbol{x} \leq \boldsymbol{b}\}$, the point $\boldsymbol{y}/2$ will be in the polytope $P = \{x \,|\, A\boldsymbol{x} \leq \boldsymbol{1}\}$ with probability at least $1 - ne^{-n}$. This guarantees that $P$ contains a point of norm at least $k$. Since $P$ contains the unit ball, Lemma 6.2 implies that $P$ contains an ellipse of volume at least twice that of the unit ball. The update step of our algorithm identifies such an ellipse and scales and translates so that it becomes the unit ball, and it then restarts with this new polytope as its input. This new polytope has at most half the volume of the original polytope.

All the vertices of the original polyhedron are contained in a ball of radius $2^{O(n^2L)}$, where $L$ is the maximum bit-length of any number in the input, and so their convex hull has volume at most $2^{O(n^3L)}$ times that of the unit ball [7]. Each iteration of the algorithm that finds a point of norm at least $k$ decreases the volume of $P$ by a factor of at least 2. All of the polytopes that we construct contain the unit ball, so this can occur at most $O(n^3L)$ times. This guarantees that the **Repeat** loop finds an answer after a $O(n^3L)$ iterations with high probability, as desired. □

While the algorithm requires samples from the exponential distribution and uniform random points on the unit sphere, it is not difficult to show that it suffices to use standard discretizations of these distributions of bit-length polynomial in $n$ and $d$.

# 7. GEOMETRIC LEMMAS FOR ALGORITHM'S CORRECTNESS

LEMMA 7.1. *Let $P$ be a $k$-round polytope, let $c$ and $q$ be unit vectors, and let*

$$v = \underset{x \in P}{\text{argmax}}\, c \cdot x$$

*be the vertex of $P$ at which $c \cdot x$ is maximized. If $c \cdot q \leq -(2k^2-1)/2k^2$, then $v \cdot q \leq 0$.*

PROOF. We first note that

$$||q+c||^2 \;=\; ||q||^2 + ||c||^2 + 2(c \cdot q) \leq 2 - \frac{2k^2-1}{k^2} = \frac{1}{k^2},$$

so $||q+c|| \leq 1/k$. The fact that $P$ is contained in $B(0,k)$ implies that $||v|| \leq k$, and the fact that $P$ contains the unit ball implies that

$$v \cdot c = \max_{x \in P} c \cdot x \geq 1.$$

We therefore have

$$q \cdot v = -c \cdot v + (q+c) \cdot v \leq -1 + ||q+c||||v|| \leq 0,$$

as desired. □

We now prove some geometric facts that will be necessary for the analysis of our algorithm. We first prove a two-dimensional geometric lemma. We then use this to prove a higher-dimensional analogue, which is the version that we shall actually use to analyze our algorithm.

## 7.1 2-Dimensional Geometry Lemma

In this section, we prove a lemma about the two-dimensional objects shown in Figure 2. In this picture, $C$ is the center of a circle $\mathbf{C}$ of radius 1. $P$ is a point somewhere along the positive $x$-axis, and we have drawn the two lines tangent to the circle through $P$, the top one of which we have labeled $L$. $E$ is the center of an axis-parallel ellipse $\mathbf{E}$ with horizontal semi-axis $M \geq 1$ and vertical semi-axis $m \leq 1$. The ellipse is chosen to be a maximal ellipse contained in the convex hull of the circle and $P$. Furthermore, let $S$ be the distance from $C$ to $P$, and let $Q = (1-m^2)/2$.

LEMMA 7.2. *With the definitions above,*

$$M = Q(S-1) + 1.$$

[2] If an ellipsoid $E$ is given as the set $E = \{x \,|\, x^T Q^{-1} x \leq 1\}$, where $Q$ is a symmetric, positive definite matrix, then the *semi-axes* of $E$ have lengths equal to the the eigenvalues of $Q$. For example, the semi-axes of the sphere are all of length 1.

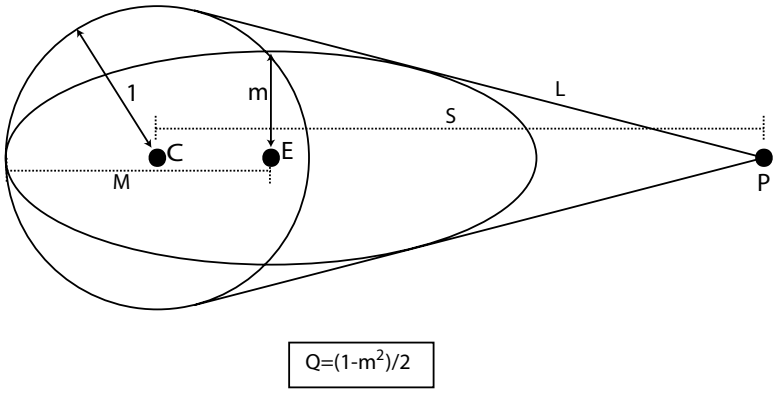


**Figure 2: The geometric objects considered in Lemma 7.2**

PROOF. Without loss of generality, let $E$ be the origin. The circle and ellipse are mutually tangent at their leftmost points on the $x$-axis, so $\mathbf{C}$ is at $(-M+1, 0)$, and $P$ is therefore at $(S-M+1, 0)$. Let

$$\ell = \left(\frac{1}{S}, \sqrt{1-\frac{1}{S^2}}\right),$$

and let $L$ be the line given by

$$L = \left\{(x,y) \,|\, \ell \cdot (x,y) = \frac{S-M+1}{S}\right\}.$$

We claim that $L$ has the following three properties, as shown in Figure 2:

1. $L$ passes through $P$.
2. $L$ is tangent to $\mathbf{C}$.
3. If we take the major semi-axis $M$ of the ellipse $\mathbf{E}$ to be $Q(S-1)+1$, then $L$ is tangent to $\mathbf{E}$.

Establishing these properties would immediately imply Lemma 7.2, so it suffices to check them one by one.

1. This follows by direct computation—we simply note that the point $P = (S-M+1, 0)$ satisfies the equation for $L$.
2. It suffices to show that the distance from the point $C$ to the line $L$ is exactly 1. Since $\mathcal{L}$ is the unit normal to $L$, it suffices to check that
$$\ell \cdot C = \left(\frac{S-M+1}{S}\right) - 1 = \frac{-M+1}{S},$$
which again follows by direct computation.
3. Let
$$\mathcal{L} = (\mathcal{L}_x, \mathcal{L}_y) = \frac{S}{S-M+1}\,\ell = \left(\frac{1}{S-M+1}, \frac{\sqrt{S^2-1}}{S-M+1}\right),$$
so that $L = \{(x,y) \,|\, \mathcal{L} \cdot (x,y) = 1\}$. When expressed in this form, $L$ will be tangent to $E$ if and only if $\mathcal{L}_x^2 M^2 + \mathcal{L}_y^2 m^2 = 1$. This can be verified by plugging in $M = Q(S-1)+1$ and $Q = (1-m^2)/2$, and then expanding the left-hand side of the equation. □

## 7.2 High-Dimensional Geometry Lemma

LEMMA 7.3. *Let $B \subseteq \mathbb{R}^d$ be the unit ball, let $P$ be a point at distance $S$ from the origin, and let $C = \mathrm{conv}(B, P)$ be their convex hull. For any $m \leq 1$, $C$ contains an ellipsoid with $(d-1)$ semi-axes of length $m$ and one semi-axis of length $\left(1-m^2\right)(S-1)/2+1$.*

PROOF. Without loss of generality, take $P = (S, 0, \ldots, 0)$. Consider an axis-parallel ellipsoid $E$ with the axes described in the above theorem, with its distinct axis parallel to $e_1$, and translated so that it is tangent to $B$ at $(-1, 0, \ldots, 0)$.

We assert that $E$ is contained in $C$. It suffices to check the containment when we intersect with an arbitrary 2-dimensional subspace containing 0 and $P$. In this case, we have exactly the setup of Lemma 7.2, and our result follows immediately. □

PROOF PROOF OF LEMMA 6.2. If we set $m = 1 - 1/d$, then Lemma 7.3 guarantees that the length of the longer semi-axis of the ellipse will be at least

$$\left(1 - \left(1-\frac{1}{d}\right)^2\right)\frac{16d}{2} \geq 8.$$

So, the ratio of the volume of the unit ball to the ellipse is at least

$$\begin{aligned} V/\mathrm{vol}(B) &\geq \left(1-\frac{1}{d}\right)^{d-1} 8 \\ &\geq \frac{8}{4} = 2. \quad \square \end{aligned}$$

# 8. REFERENCES


[1] D. Bertsimas and S. Vempala. Solving convex programs by random walks. *J. ACM*, 51(4):540–556, 2004.

[2] K. H. Borgwardt. *The Simplex Method: a probabilistic analysis*. Number 1 in Algorithms and Combinatorics. Springer-Verlag, 1980.

[3] G. B. Dantzig. Maximization of linear function of variables subject to linear inequalities. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347. 1951.

[4] A. Deshpande and D. A. Spielman. Improved smoothed analysis of the shadow vertex simplex method. preliminary version appeared in FOCS '05, 2005.

[5] J. Dunagan and S. Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. In *Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing (STOC-04)*, pages 315–320, New York, June 13–15 2004. ACM Press.

[6] S. Gass and T. Saaty. The computational algorithm for the parametric objective function. *Naval Research Logistics Quarterly*, 2:39–45, 1955.

[7] M. Grotschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1991.

[8] G. Kalai and D. J. Kleitman. A quasi-polynomial bound for the diameter of graphs of polyhedra. *Bulletin Amer. Math. Soc.*, 26:315–316, 1992.

[9] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[10] L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademia Nauk SSSR*, pages 1093–1096, 1979.

[11] N. Megiddo and R. Chandrasekaran. On the epsilon-perturbation method for avoiding degeneracy. *Operations Research Letters*, 8:305–308, 1989.

[12] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.