Solving Totally Unimodular LPs with the Shadow Vertex Algorithm*

Tobias Brunsch, Anna Großwendt, and Heiko Röglin

Department of Computer Science
University of Bonn, Germany
{brunsch,grosswen,roeglin}@cs.uni-bonn.de

Abstract

We show that the shadow vertex simplex algorithm can be used to solve linear programs in strongly polynomial time with respect to the number n of variables, the number m of constraints, and $1/\delta$, where δ is a parameter that measures the flatness of the vertices of the polyhedron. This extends our recent result that the shadow vertex algorithm finds paths of polynomial length (w.r.t. n, m, and $1/\delta$) between two given vertices of a polyhedron [4].

Our result also complements a recent result due to Eisenbrand and Vempala [6] who have shown that a certain version of the random edge pivot rule solves linear programs with a running time that is strongly polynomial in the number of variables n and $1/\delta$, but independent of the number m of constraints. Even though the running time of our algorithm depends on m, it is significantly faster for the important special case of totally unimodular linear programs, for which $1/\delta \leq n$ and which have only $O(n^2)$ constraints.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases linear optimization, simplex algorithm, shadow vertex method

Digital Object Identifier 10.4230/LIPIcs.STACS.2015.171

1 Introduction

The shadow vertex algorithm is a well-known pivoting rule for the simplex method that has gained attention in recent years because it was shown to have polynomial running time in the model of smoothed analysis [8]. Recently we have observed that it can also be used to find short paths between given vertices of a polyhedron [4]. Here short means that the path length is $O(\frac{mn^2}{\delta^2})$, where n denotes the number of variables, m denotes the number of constraints, and δ is a parameter of the polyhedron that we will define shortly.

Our result left open the question whether or not it is also possible to solve linear programs in polynomial time with respect to n, m, and $1/\delta$ by the shadow vertex simplex algorithm. In this article we resolve this question and introduce a variant of the shadow vertex simplex algorithm that solves linear programs in strongly polynomial time with respect to these parameters.

For a given matrix $A = [a_1, \dots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n}$ and vectors $b \in \mathbb{R}^m$ and $c_0 \in \mathbb{R}^n$ our goal is to solve the linear program $\max\{{c_0}^{\mathrm{T}} x \mid Ax \leq b\}$. We assume without loss of generality that $\|c_0\| = 1$ and $\|a_i\| = 1$ for every row a_i of the constraint matrix.

▶ **Definition 1.** The matrix A satisfies the *δ-distance property* if the following condition holds: For any $I \subseteq \{1, \dots, m\}$ and any $j \in \{1, \dots, m\}$, if $a_j \notin \operatorname{span}\{a_i \mid i \in I\}$

* This research was supported by ERC Starting Grant 306465 (BeyondWorstCase).

32nd Symposium on Theoretical Aspects of Computer Science (STACS 2015).
Editors: Ernst W. Mayr and Nicolas Ollinger; pp. 171–183
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

then $\operatorname{dist}(a_j, \operatorname{span}\{a_i \mid i \in I\}) \geq \delta$. In other words, if a_j does not lie in the subspace spanned by the a_i, $i \in I$, then its distance to this subspace is at least δ.

We present a variant of the shadow vertex simplex algorithm that solves linear programs in strongly polynomial time with respect to n, m, and $1/\delta$, where δ denotes the largest δ' for which the constraint matrix of the linear program satisfies the δ'-distance property. (In the following theorems, we assume $m \geq n$. If this is not the case, we use a method described in [3] to add irrelevant constraints so that A has rank n. Hence, for instances that have fewer constraints than variables, the parameter m should be replaced by n in all bounds.)

▶ **Theorem 2.** *There exists a randomized variant of the shadow vertex simplex algorithm (described in Section 2) that solves linear programs with n variables and m constraints satisfying the δ-distance property using $O\big(\frac{mn^3}{\delta^2} \cdot \log\big(\frac{1}{\delta}\big)\big)$ pivots in expectation if a basic feasible solution is given. A basic feasible solution can be found using $O\big(\frac{m^5}{\delta^2} \cdot \log\big(\frac{1}{\delta}\big)\big)$ pivots in expectation.*

We stress that the algorithm can be implemented without knowing the parameter δ. From the theorem it follows that the running time of the algorithm is strongly polynomial with respect to the number n of variables, the number m of constraints, and $1/\delta$ because every pivot can be performed in time $O(mn)$ in the arithmetic model of computation (see Section 2.2).[1]

Let $A \in \mathbb{Z}^{m \times n}$ be an integer matrix and let $A' \in \mathbb{R}^{m \times n}$ be the matrix that arises from A by scaling each row such that its norm equals 1. If Δ denotes an upper bound for the absolute value of any sub-determinant of A, then A' satisfies the δ-distance property for $\delta = 1/(\Delta^2 n)$ [4]. For such matrices A Phase 1 of the simplex method can be implemented more efficiently and we obtain the following result.

▶ **Theorem 3.** *For integer matrices $A \in \mathbb{Z}^{m \times n}$, there exists a randomized variant of the shadow vertex simplex algorithm (described in Section 2) that solves linear programs with n variables and m constraints using $O\big(mn^5\Delta^4 \log(\Delta + 1)\big)$ pivots in expectation if a basic feasible solution is given, where Δ denotes an upper bound for the absolute value of any sub-determinant of A. A basic feasible solution can be found using $O\big(m^6\Delta^4 \log(\Delta + 1)\big)$ pivots in expectation.*

Theorem 3 implies in particular that totally unimodular linear programs can be solved by our algorithm with $O\big(mn^5\big)$ pivots in expectation if a basic feasible solution is given and with $O\big(m^6\big)$ pivots in expectation otherwise.

Besides totally unimodular matrices there are also other classes of matrices for which $1/\delta$ is polynomially bounded in n. Eisenbrand and Vempala [6] observed, for example, that $\delta = \Omega(1/\sqrt{n})$ for edge-node incidence matrices of undirected graphs with n vertices. One can also argue that δ can be interpreted as a condition number of the matrix A in the following sense: If $1/\delta$ is large then there must be an $(n \times n)$-submatrix of A of rank n that is almost singular.

1.1 Related Work

Shadow vertex simplex algorithm

We will briefly explain the geometric intuition behind the shadow vertex simplex algorithm. For a complete and more formal description, we refer the reader to [2] or [8]. Let us consider

[1] By *strongly polynomial with respect to n, m, and $1/\delta$* we mean that the number of steps in the arithmetic model of computation is bounded polynomially in n, m, and $1/\delta$ and the size of the numbers occurring during the algorithm is polynomially bounded in the encoding size of the input.

the linear program $\max\{{c_0}^{\mathrm{T}} x \mid Ax \leq b\}$ and let $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ denote the polyhedron of feasible solutions. Assume that an initial vertex x_1 of P is known and assume, for the sake of simplicity, that there is a unique optimal vertex $x^\star$ of P that maximizes the objective function ${c_0}^{\mathrm{T}} x$. The shadow vertex pivot rule first computes a vector $w \in \mathbb{R}^n$ such that the vertex x_1 minimizes the objective function $w^{\mathrm{T}} x$ subject to $x \in P$. Again for the sake of simplicity, let us assume that the vectors c_0 and w are linearly independent.

In the second step, the polyhedron P is projected onto the plane spanned by the vectors c_0 and w. The resulting projection is a (possibly open) polygon P' and one can show that the projections of both the initial vertex x_1 and the optimal vertex $x^\star$ are vertices of this polygon. Additionally, every edge between two vertices x and y of P' corresponds to an edge of P between two vertices that are projected onto x and y, respectively. Due to these properties a path from the projection of x_1 to the projection of $x^\star$ along the edges of P' corresponds to a path from x_1 to $x^\star$ along the edges of P.

This way, the problem of finding a path from x_1 to $x^\star$ on the polyhedron P is reduced to finding a path between two vertices of a polygon. There are at most two such paths and the shadow vertex pivot rule chooses the one along which the objective ${c_0}^{\mathrm{T}} x$ improves.

Finding short paths

In [4] we considered the problem of finding a short path between two given vertices x_1 and x_2 of the polyhedron P along the edges of P. Our algorithm is the following variant of the shadow vertex algorithm: Choose two vectors $w_1, w_2 \in \mathbb{R}^n$ such that x_1 uniquely minimizes ${w_1}^{\mathrm{T}} x$ subject to $x \in P$ and x_2 uniquely maximizes ${w_2}^{\mathrm{T}} x$ subject to $x \in P$. Then project the polyhedron P onto the plane spanned by w_1 and w_2 in order to obtain a polygon P'. Let us call the projection π. By the same arguments as above, it follows that $\pi(x_1)$ and $\pi(x_2)$ are vertices of P' and that a path from $\pi(x_1)$ to $\pi(x_2)$ along the edges of P' can be translated into a path from x_1 to x_2 along the edges of P. Hence, it suffices to compute such a path to solve the problem. Again computing such a path is easy because P' is a two-dimensional polygon.

The vectors w_1 and w_2 are not uniquely determined, but they can be chosen from cones that are determined by the vertices x_1 and x_2 and the polyhedron P. We proved in [4] that the expected path length is $O(\frac{mn^2}{\delta^2})$ if w_1 and w_2 are chosen randomly from these cones. For totally unimodular matrices this implies that the diameter of the polyhedron is bounded by $O(mn^4)$, which improved a previous result by Dyer and Frieze [5] who showed that for this special case paths of length $O(m^3 n^{16} \log(mn))$ can be computed efficiently.

Additionally, Bonifas et al. [1] proved that in a polyhedron defined by an integer matrix A between any pair of vertices there exists a path of length $O(\Delta^2 n^4 \log(n\Delta))$ where Δ is the largest absolute value of any sub-determinant of A. For the special case that A is a totally unimodular matrix, this bound simplifies to $O(n^4 \log n)$. Their proof is non-constructive, however.

Geometric random edge

Eisenbrand and Vempala [6] have presented an algorithm that solves a linear program $\max\{{c_0}^{\mathrm{T}} x | Ax \leq b\}$ in strongly polynomial time with respect to the parameters n and $1/\delta$. Remarkably the running time of their algorithm does not depend on the number m of constraints. Their algorithm is based on a variant of the random edge pivoting rule. The algorithm performs a random walk on the vertices of the polyhedron whose transition probabilities are chosen such that it quickly attains a distribution close to its stationary distribution.

In the stationary distribution the random walk is likely at a vertex x_c that optimizes an objective function $c^{\mathrm{T}}x$ with $\|c_0 - c\| < \frac{\delta}{2n}$. The δ-distance property guarantees that x_c and the optimal vertex $x^\star$ with respect to the objective function ${c_0}^{\mathrm{T}}x$ lie on a common facet. This facet is then identified and the algorithm is run again in one dimension lower. This is repeated at most n times until all facets of the optimal vertex $x^\star$ are identified. The number of pivots to identify one facet of $x^\star$ is proven to be $O(n^{10}/\delta^8)$. A single pivot can be performed in polynomial time but determining the right transition probabilities is rather sophisticated and requires to approximately integrate a certain function over a convex body.

Let us point out that the number of pivots of our algorithm depends on the number m of constraints. However, Heller showed that for the important special case of totally unimodular linear programs $m = O(n^2)$ [7]. Using this observation we also obtain a bound that depends polynomially only on n for totally unimodular matrices.

Combinatorial linear programs

Éva Tardos has proved in 1986 that combinatorial linear programs can be solved in strongly polynomial time [9]. Here combinatorial means that A is an integer matrix whose largest entry is polynomially bounded in n. Her result implies in particular that totally unimodular linear programs can be solved in strongly polynomial time, which is also implied by Theorem 3. However, the proof and the techniques used to prove Theorem 3 are completely different from those in [9].

1.2 Our Contribution

We replace the random walk in the algorithm of Eisenbrand and Vempala by the shadow vertex algorithm. Given a vertex x_0 of the polyhedron P we choose an objective function $w^{\mathrm{T}}x$ for which x_0 is an optimal solution. As in [4] we choose w uniformly at random from the cone determined by x_0. Then we randomly perturb each coefficient in the given objective function ${c_0}^{\mathrm{T}}x$ by a small amount. We denote by $c^{\mathrm{T}}x$ the perturbed objective function. As in [4] we prove that the projection of the polyhedron P onto the plane spanned by w and c has $O\big(\frac{mn^2}{\delta^2}\big)$ edges in expectation. If the perturbation is so small that $\|c_0 - c\| < \frac{\delta}{2n}$, then the shadow vertex algorithm yields with $O\big(\frac{mn^2}{\delta^2}\big)$ pivots a solution that has a common facet with the optimal solution $x^\star$. We follow the same approach as Eisenbrand and Vempala and identify the facets of $x^\star$ one by one with at most n calls of the shadow vertex algorithm.

The analysis in [4] exploits that the two objective functions possess the same type of randomness (both are chosen uniformly at random from some cones). This is not the case anymore because every component of c is chosen independently uniformly at random from some interval. This changes the analysis significantly and introduces technical difficulties that we address in this article.

The problem when running the simplex method is that a feasible solution needs to be given upfront. Usually, such a solution is determined in Phase 1 by solving a modified linear program with a constraint matrix A' for which a feasible solution is known and whose optimal solution is feasible for the linear program one actually wants to solve. There are several common constructions for this modified linear program, it is, however, not clear how the parameter δ is affected by modifying the linear program. To solve this problem, Eisenbrand and Vempala [6] have suggested a method for Phase 1 for which the modified constraint matrix A' satisfies the δ-distance property for the same δ as the matrix A. However, their method is very different from usual textbook methods and needs to solve m different linear programs to find an initial feasible solution for the given linear program. We show that also

one of the usual textbook methods can be applied. We argue that $1/\delta$ increases by a factor of at most $\sqrt{m}$ and that Δ, the absolute value of any sub-determinant of A, does not change at all in case one considers integer matrices. In this construction, the number of variables increases from n to $n+m$.

1.3 Outline and Notation

In the following we assume that we are given a linear program $\max\{{c_0}^{\mathrm{T}} x \mid Ax \leq b\}$ with vectors $b \in \mathbb{R}^m$ and $c_0 \in \mathbb{R}^n$ and a matrix $A = [a_1, \ldots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n}$. Moreover, we assume that $\|c_0\| = \|a_i\| = 1$ for all $i \in [m]$, where $[m] := \{1, \ldots, m\}$ and $\|\cdot\|$ denotes the Euclidean norm. This entails no loss of generality since any linear program can be brought into this form by scaling the objective function and the constraints appropriately. For a vector $x \in \mathbb{R}^n \setminus \{0^n\}$ we denote by $\mathcal{N}(x) = \frac{1}{\|x\|} \cdot x$ the normalization of vector x.

For a vertex v of the polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ we call the set of row indices $B_v = \{i \in \{1, \ldots, m\} \mid a_i \cdot v = b_i\}$ *basis* of v. Then the *normal cone* C_v of v is given by the set

$$C_v = \left\{ \sum_{i \in B_v} \lambda_i a_i \mid \lambda_i \geq 0 \right\}.$$

We will describe our algorithm in Section 2.1 where we assume that the linear program in non-degenerate, that A has full rank n, and that the polyhedron P is bounded. We have already described in Section 3 of [4] that the linear program can be made non-degenerate by slightly perturbing the vector b. This does not affect the parameter δ because δ depends only on the matrix A. In Section 3 we analyze our algorithm and prove Theorem 2. In the full version of this article [3] we discuss why we can assume that A has full rank and why P is bounded. There are, of course, textbook methods to transform a linear program into this form. However, we need to be careful that this transformation does not change δ. Moreover in [3] we discuss how Phase 1 of the simplex method can be implemented and we give an alternative definition of δ and discuss some properties of this parameter. Due to space limitations most proofs are omitted. They can also be found in [3].

2 Algorithm

Given a linear program $\max\{{c_0}^{\mathrm{T}} x \mid Ax \leq b\}$ and a basic feasible solution x_0, our algorithm randomly perturbs each coefficient of the vector c_0 by at most $1/\phi$ for some parameter ϕ to be determined later. Let us call the resulting vector c. The next step is then to use the shadow vertex algorithm to compute a path from x_0 to a vertex x_c which maximizes the function $c^{\mathrm{T}} x$ for $x \in P$. For $\phi > \frac{2n^{3/2}}{\delta}$ one can argue that the solution x has a facet in common with the optimal solution $x^\star$ of the given linear program with objective function ${c_0}^{\mathrm{T}} x$. Then the algorithm is run again on this facet one dimension lower until all facets that define $x^\star$ are identified.

Section 2.1 presents the shadow vertex algorithm, the main building block of our algorithm. Details of the identification and reduction to an optimal facet are provided in the full version of this paper. In Section 2.2 we discuss the running time of a single pivot step of the shadow vertex algorithm.

2.1 The Shadow Vertex Method

In this section we assume that we are given a linear program of the form $\max\{{c_0}^{\mathrm{T}} x \mid x \in P\}$, where $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is a bounded polyhedron (i.e., a polytope), and a basic feasible solution $x_0 \in P$. We assume $\|c_0\| = \|a_i\| = 1$ for all rows a_i of A. Furthermore, we assume that the linear program is non-degenerate.

Due to the assumption $\|c_0\| = 1$ it holds $c_0 \in [-1, 1]^n$. Our algorithm slightly perturbs the given objective function ${c_0}^{\mathrm{T}} x$ at random. For each component $(c_0)_i$ of c_0 it chooses an arbitrary interval $I_i \subseteq [-1, 1]$ of length $1/\phi$ with $(c_0)_i \in I_i$, where ϕ denotes a parameter that will be given to the algorithm. Then a random vector $c \in [-1, 1]^n$ is drawn as follows: Each component c_i of c is chosen independently uniformly at random from the interval I_i. We denote the resulting random vector by $\mathrm{pert}(c_0, \phi)$. Note that we can bound the norm of the difference $\|c_0 - c\|$ between the vectors c_0 and c from above by $\frac{\sqrt{n}}{\phi}$.

The shadow vertex algorithm is given as Algorithm 1. It is assumed that ϕ is given to the algorithm as a parameter. We will discuss later how we can run the algorithm without knowing this parameter. Let us remark that the Steps 5 and 6 in Algorithm 1 are actually not executed separately. Instead of computing the whole projection P' in advance, the edges of P' are computed on the fly one after another.

Algorithm 1 Shadow Vertex Algorithm

1: Generate a random perturbation $c = \mathrm{pert}(c_0, \phi)$ of c_0.
2: Determine n linearly independent rows ${u_k}^{\mathrm{T}}$ of A for which ${u_k}^{\mathrm{T}} x_0 = b_k$.
3: Draw a vector $\lambda \in (0, 1]^n$ uniformly at random.
4: Set $w = -[u_1, \ldots, u_n] \cdot \lambda$.
5: Use the function $\pi : x \mapsto \left(c^{\mathrm{T}} x, w^{\mathrm{T}} x\right)$ to project P onto the Euclidean plane and obtain the shadow vertex polygon $P' = \pi(P)$.
6: Walk from $\pi(x_0)$ along the edges of P' in increasing direction of the first coordinate until a rightmost vertex $\tilde{x}_c$ of P' is found.
7: Output the vertex x_c of P that is projected onto $\tilde{x}_c$.

Note that

$$\|w\| \leq \sum_{k=1}^{n} \lambda_k \cdot \|u_k\| \leq \sum_{k=1}^{n} \lambda_k \leq n,$$

where the second inequality follows because all rows of A are assumed to have norm 1.

The Shadow Vertex Algorithm yields a path from the vertex x_0 to a vertex x_c that is optimal for the linear program $\max\{c^{\mathrm{T}} x \mid x \in P\}$ where $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. The following theorem (whose proof can be found in Section 3) bounds the expected length of this path, i.e., the number of pivots.

▶ **Theorem 4.** *For any $\phi \geq \sqrt{n}$ the expected number of edges on the path output by Algorithm 1 is $O\left(\frac{mn^2}{\delta^2} + \frac{m\sqrt{n}\phi}{\delta}\right)$.*

Since $\|c_0 - c\| \leq \frac{\sqrt{n}}{\phi}$ choosing $\phi > \frac{2n^{3/2}}{\delta}$ suffices to ensure $\|c_0 - c\| < \frac{\delta}{2n}$. This implies (see [3]) that, for such a choice of ϕ, the vertex x_c has a facet in common with the optimal solution of the linear program $\max\{{c_0}^{\mathrm{T}} x \mid x \in P\}$ and we can reduce the dimension of the linear program as discussed in [3]. This step is repeated at most n times. It is important that we can start each repetition with a known feasible solution because the transformation in [3] maps the optimal solution of the linear program of repetition i onto a feasible solution with

which repetition $i+1$ can be initialized. Together with Theorem 4 this implies that an optimal solution of the linear program can be found by performing in expectation $O\left(\frac{mn^3}{\delta^2} + \frac{mn^{3/2}\phi}{\delta}\right)$ pivots if a basic feasible solution x_0 and the right choice of ϕ are given. We will refer to this algorithm as *repeated shadow vertex algorithm*.

Since δ is not known to the algorithm, the right choice for ϕ cannot easily be computed. Instead we will try values for ϕ until an optimal solution is found. For $i \in \mathbb{N}$ let $\phi_i = 2^i n^{3/2}$. First we run the repeated shadow vertex algorithm with $\phi = \phi_0$ and check whether the returned solution is an optimal solution for the linear program $\max\{{c_0}^{\mathrm{T}} x \mid x \in P\}$. If this is not the case, we run the repeated shadow vertex algorithm with $\phi = \phi_1$, and so on. We continue until an optimal solution is found. For $\phi = \phi_{i^\star}$ with $i^\star = \left\lceil \log_2\left(1/\delta\right)\right\rceil + 2$ this is the case because $\phi_{i^\star} > \frac{2n^{3/2}}{\delta}$.

Since $\phi_{i^\star} \leq \frac{8n^{3/2}}{\delta}$, in accordance with Theorem 4, each of the at most $i^\star = O(\log(1/\delta))$ calls of the repeated shadow vertex algorithm uses in expectation

$$O\left(\frac{mn^3}{\delta^2} + \frac{mn^{3/2}\phi_{i^\star}}{\delta}\right) = O\left(\frac{mn^3}{\delta^2}\right).$$

pivots. Together this proves the first part of Theorem 2. The second part follows from [3], where it is proven that Phase 1 can be realized with increasing $1/\delta$ by at most $\sqrt{m}$ and increasing the number of variables from n to $n + m \leq 2m$. This implies that the expected number of pivots of each call of the repeated shadow vertex algorithm in Phase 1 is $O(m(n+m)^3\sqrt{m}^2/\delta^2) = O(m^5/\delta^2)$. Since $1/\delta$ can increase by a factor of $\sqrt{m}$, the argument above yields that we need to run the repeated shadow vertex algorithm at most $i^\star = O(\log(\sqrt{m}/\delta))$ times in Phase 1 to find a basic feasible solution. By setting $\phi_i = 2^i\sqrt{m}(n+m)^{3/2}$ instead of $\phi_i = 2^i(n+m)^{3/2}$ this number can be reduced to $i^\star = O(\log(1/\delta))$ again.

Theorem 3 follows from Theorem 2 using the following fact from [4]: Let $A \in \mathbb{Z}^{m\times n}$ be an integer matrix and let $A' \in \mathbb{R}^{m\times n}$ be the matrix that arises from A by scaling each row such that its norm equals 1. If Δ denotes an upper bound for the absolute value of any sub-determinant of A, then A' satisfies the δ-distance property for $\delta = 1/(\Delta^2 n)$. Additionally in [3] it is proven that Phase 1 can be realized without increasing Δ but with increasing the number of variables from n to $n + m \leq 2m$. Substituting $1/\delta = \Delta^2 n$ in Theorem 2 almost yields Theorem 3 except for a factor $O(\log(\Delta^2 n))$ instead of $O(\log(\Delta + 1))$. This factor results from the number $i^\star$ of calls of the repeated shadow vertex algorithm. The desired factor of $O(\log(\Delta + 1))$ can be achieved by setting $\phi_i = 2^i n^{5/2}$ if a basic feasible solution is known and $\phi_i = 2^i(n+m)^{5/2}$ in Phase 1.

2.2 Running Time of the Repeated Shadow Vertex Algorithm

So far we have only discussed the number of pivots. Let us now calculate the actual running time of our algorithm. For an initial basic feasible solution x_0 the repeated shadow vertex algorithm repeats the following three steps until an optimal solution is found. Initially let $P' = P$.

Step 1: Run the shadow vertex algorithm for the linear program $\max\{c^{\mathrm{T}} x \mid x \in P'\}$, where $c = \mathrm{pert}(c_0, \phi)$. We will denote this linear program by LP'.

Step 2: Let x_c denote the returned vertex in Step 1, which is optimal for the objective function $c^{\mathrm{T}} x$. Identify an element a'_i of x_c that is in common with the optimal basis.

Step 3: Calculate an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ that rotates a'_i into the first unit vector e_1 and set LP' to the projection of the current LP' onto the orthogonal complement. Let P' denote the polyhedron of feasible solutions of LP'.

First note that the three steps are repeated at most n times during the algorithm. In Step 1 the shadow vertex algorithm is run once. Step 1 to Step 4 of Algorithm 1 can be performed in time $O(m)$ as we assumed P to be non-degenerate (this implies P' to be non-degenerate in each further step). Step 5 and Step 6 can be implemented with strongly polynomial running time in a tableau form, described in [2]. The tableau can be set up in time $O((m-d)d^3) = O(mn^3)$ where d is the dimension of P'. By Theorem 1 of [2] we can identify for a vertex on a path the row which leaves the basis and the row which is added to the basis in order to move to the next vertex in time $O(m)$ using the tableau. After that, the tableau has to be updated. This can be done in $O((m-d)d) = O(mn)$ steps. Using this and Theorem 4 we can compute the path from x_0 to x_c in expected time $O\big(mn^3 + mn \cdot \big(\frac{mn^2}{\delta^2} + \frac{m\sqrt{n}\phi}{\delta}\big)\big) = O\big(\frac{m^2n^3}{\delta^2} + \frac{m^2n^{3/2}\phi}{\delta}\big)$. Using that $\phi \leq \frac{8n^{3/2}}{\delta}$, as discussed above, yields a running time of $O\big(\frac{m^2n^3}{\delta^2}\big)$.

Once we have calculated the basis of x_c we can easily compute the element a_i that is also an element of the optimal basis. Assume the rows $a'_1, \ldots, a'_n$ are the basis of x_c. Eisenbrand and Vempala discuss in [6] that we can solve the system of linear equations $[a'_1, \ldots, a'_n]\mu = c$ and choose the row for which the coefficient μ_i is maximal. Then a'_i is part of the optimal basis. As a consequence, Step 2 can be performed in time $O(n^3)$. Moreover solving a system of linear equations is possible in strongly polynomial time using Gaussian elimination.

In Step 3, we compute an orthogonal matrix $Q \in \mathbb{R}^{d\times d}$ such that $e_1 Q = a_i$. Since Q is orthogonal we obtain $e_1 = a_i Q^{\mathrm{T}}$ and thus, the first row of Q is given by a_i. Hence, it is sufficient to compute an orthonormal basis including a_i. This is possible in strongly polynomial time $O(d^3) = O(n^3)$ using the Gram-Schmidt process.

Since all Steps are repeated in this order at most n times we obtain a running time $O(\frac{m^2n^4}{\delta^2})$ for the repeated shadow vertex algorithm.

▶ **Theorem 5.** *The repeated shadow vertex algorithm has a running time of* $O(\frac{m^2n^4}{\delta^2})$.

The entries of both c and λ in Algorithm 1 are continuous random variables. In practice it is, however, more realistic to assume that we can draw a finite number of random bits. In the full version of this paper we show that our algorithm only needs to draw $\mathrm{poly}(\log m, n, \log(1/\delta))$ random bits in order to obtain the expected running time stated in Theorem 2 if δ (or a good lower bound for it) is known. However, if the parameter δ is not known upfront and only discrete random variables with a finite precision can be drawn, we have to modify the shadow vertex algorithm. This will give us an additional factor of $O(n)$ in the expected running time.

3 Analysis of the Shadow Vertex Algorithm

For given linear functions $L_1 \colon \mathbb{R}^n \to \mathbb{R}$ and $L_2 \colon \mathbb{R}^n \to \mathbb{R}$ we denote by $\pi = \pi_{L_1,L_2}$ the function $\pi \colon \mathbb{R}^n \to \mathbb{R}^2$, given by $\pi(x) = (L_1(x), L_2(x))$. Note that n-dimensional vectors can be treated as linear functions. By $P' = P'_{L_1,L_2}$ we denote the projection $\pi(P)$ of the polytope P onto the Euclidean plane, and by $R = R_{L_1,L_2}$ we denote the path from the bottommost vertex of P' to the rightmost vertex of P' along the edges of the lower envelope of P'.

Our goal is to bound the expected number of edges of the path $R = R_{c,w}$, which is random since c and w are random. Each edge of R corresponds to a slope in $(0, \infty)$. These slopes are pairwise distinct with probability one (see Lemma 7). Hence, the number of edges of R equals the number of distinct slopes of R.

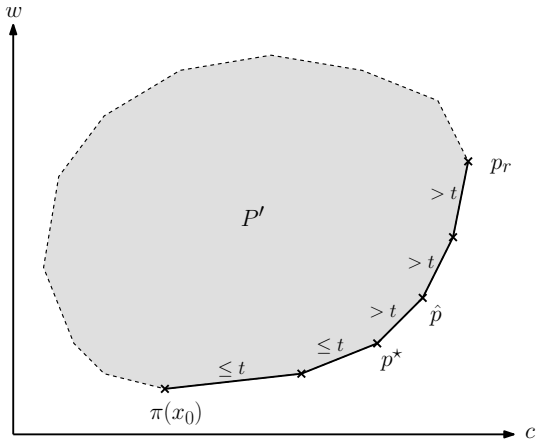

Figure 1 Slopes of the vertices of R.

▶ **Definition 6.** For a real $\varepsilon > 0$ let $\mathcal{F}_\varepsilon$ denote the event that there are three pairwise distinct vertices z_1, z_2, z_3 of P such that z_1 and z_3 are neighbors of z_2 and such that

$$\left| \frac{w^{\mathrm{T}} \cdot (z_2 - z_1)}{c^{\mathrm{T}} \cdot (z_2 - z_1)} - \frac{w^{\mathrm{T}} \cdot (z_3 - z_2)}{c^{\mathrm{T}} \cdot (z_3 - z_2)} \right| \leq \varepsilon \,.$$

Note that if event $\mathcal{F}_\varepsilon$ does not occur, then all slopes of R differ by more than ε. Particularly, all slopes are pairwise distinct. First of all we show that event $\mathcal{F}_\varepsilon$ is very unlikely to occur if ε is chosen sufficiently small. The proof of the following lemma is almost identical to the corresponding proof in [4] except that we need to adapt it to the different random model of c.

▶ **Lemma 7.** *The probability of event $\mathcal{F}_\varepsilon$ tends to 0 for $\varepsilon \to 0$.*

Let p be a vertex of R, but not the bottommost vertex $\pi(x_0)$. We call the slope s of the edge incident to p to the left of p *the slope of* p. As a convention, we set the slope of $\pi(x_0)$ to 0 which is smaller than the slope of any other vertex p of R.

Let $t \geq 0$ be an arbitrary real, let $p^\star$ be the rightmost vertex of R whose slope is at most t, and let $\hat{p}$ be the right neighbor of $p^\star$, i.e., $\hat{p}$ is the leftmost vertex of R whose slope exceeds t (see Figure 1). Let $x^\star$ and $\hat{x}$ be the neighboring vertices of P with $\pi(x^\star) = p^\star$ and $\pi(\hat{x}) = \hat{p}$. Now let $i = i(x^\star, \hat{x}) \in [m]$ be the index for which ${a_i}^{\mathrm{T}} x^\star = b_i$ and for which $\hat{x}$ is the (unique) neighbor x of $x^\star$ for which ${a_i}^{\mathrm{T}} x < b_i$. This index is unique due to the non-degeneracy of the polytope P. For an arbitrary real $\gamma \geq 0$ we consider the vector $\tilde{w} := w - \gamma \cdot a_i$.

▶ **Lemma 8** (Lemma 9 of [4])**.** *Let $\tilde{\pi} = \pi_{c,\tilde{w}}$ and let $\tilde{R} = R_{c,\tilde{w}}$ be the path from $\tilde{\pi}(x_0)$ to the rightmost vertex $\tilde{p}_r$ of the projection $\tilde{\pi}(P)$ of polytope P. Furthermore, let $\tilde{p}^\star$ be the rightmost vertex of $\tilde{R}$ whose slope does not exceed t. Then $\tilde{p}^\star = \tilde{\pi}(x^\star)$.*

Let us reformulate the statement of Lemma 8 as follows: The vertex $\tilde{p}^\star$ is defined for the path $\tilde{R}$ of polygon $\tilde{\pi}(R)$ with the same rules as used to define the vertex $p^\star$ of the original path R of polygon $\pi(P)$. Even though R and $\tilde{R}$ can be very different in shape, both vertices, $p^\star$ and $\tilde{p}^\star$, correspond to the same solution $x^\star$ in the polytope P, that is, $p^\star = \pi(x^\star)$ and $\tilde{p}^\star = \tilde{\pi}(x^\star)$.

Lemma 8 holds for any vector $\tilde{w}$ on the ray $\vec{r} = \{w - \gamma \cdot a_i \,|\, \gamma \geq 0\}$. As $\|w\| \leq n$ (see Section 2.1), we have $w \in [-n, n]^n$. Hence, ray $\vec{r}$ intersects the boundary of $[-n, n]^n$ in a unique point z. We choose $\tilde{w} = \tilde{w}(w, i) := z$ and obtain the following result.

▶ **Corollary 9.** *Let $\tilde{\pi} = \pi_{c,\tilde{w}(w,i)}$ and let $\tilde{p}^\star$ be the rightmost vertex of path $\tilde{R} = R_{c,\tilde{w}(w,i)}$ whose slope does not exceed t. Then $\tilde{p}^\star = \tilde{\pi}(x^\star)$.*

Note that Corollary 9 only holds for the right choice of index $i = i(x^\star, \hat{x})$. However, the vector $\tilde{w}(w,i)$ can be defined for any vector $w \in [-n,n]^n$ and any index $i \in [m]$. In the remainder, index i is an arbitrary index from $[m]$.

We can now define the following event that is parameterized in i, t, and a real $\varepsilon > 0$ and that depends on c and w.

▶ **Definition 10.** For an index $i \in [m]$ and a real $t \geq 0$ let $\tilde{p}^\star$ be the rightmost vertex of $\tilde{R} = R_{c,\tilde{w}(w,i)}$ whose slope does not exceed t and let $y^\star$ be the corresponding vertex of P. For a real $\varepsilon > 0$ we denote by $E_{i,t,\varepsilon}$ the event that the conditions

- ${a_i}^{\mathrm{T}} y^\star = b_i$ and
- $\frac{w^{\mathrm{T}}(\hat{y}-y^\star)}{c^{\mathrm{T}}(\hat{y}-y^\star)} \in (t, t+\varepsilon]$, where $\hat{y}$ is the neighbor y of $y^\star$ for which ${a_i}^{\mathrm{T}} y < b_i$,

are met. Note that the vertex $\hat{y}$ always exists and that it is unique since the polytope P is non-degenerate.

Let us remark that the vertices $y^\star$ and $\hat{y}$, which depend on the index i, equal $x^\star$ and $\hat{x}$ if we choose $i = i(x^\star, \hat{x})$. For other choices of i, this is, in general, not the case.

Observe that all possible realizations of w from the line $L := \{w + x \cdot a_i \,|\, x \in \mathbb{R}\}$ are mapped to the same vector $\tilde{w}(w,i)$. Consequently, if c is fixed and if we only consider realizations of λ for which $w \in L$, then vertex $\tilde{p}^\star$ and, hence, vertex $y^\star$ from Definition 10 are already determined. However, since w is not completely specified, we have some randomness left for event $E_{i,t,\varepsilon}$ to occur. This allows us to bound the probability of event $E_{i,t,\varepsilon}$ from above (see proof of Lemma 12). The next lemma shows why this probability matters.

▶ **Lemma 11** (Lemma 12 from [4]). *For any $t \geq 0$ and $\varepsilon > 0$ let $A_{t,\varepsilon}$ denote the event that the path $R = R_{c,w}$ has a slope in $(t, t+\varepsilon]$. Then, $A_{t,\varepsilon} \subseteq \bigcup_{i=1}^{m} E_{i,t,\varepsilon}$.*

With Lemma 11 we can now bound the probability of event $A_{t,\varepsilon}$. The proof of the next lemma is almost identical to the proof of Lemma 13 from [4]. The only differences to Lemma 13 from [4] are that we can now use the stronger upper bound $\|c\| \leq 2$ instead of $\|c\| \leq n$ and that we have more carefully analyzed the case of large t.

▶ **Lemma 12.** *For any $\phi \geq \sqrt{n}$, any $t \geq 0$, and any $\varepsilon > 0$ the probability of event $A_{t,\varepsilon}$ is bounded by*

$$\mathbf{Pr}\left[A_{t,\varepsilon}\right] \leq \frac{2mn^2\varepsilon}{\max\left\{\frac{n}{2}, t\right\} \cdot \delta^2} \leq \frac{4mn\varepsilon}{\delta^2}\,.$$

▶ **Lemma 13.** *For any interval I let X_I denote the number of slopes of $R = R_{c,w}$ that lie in the interval I. Then, for any $\phi \geq \sqrt{n}$,*

$$\mathbf{E}\left[X_{(0,n]}\right] \leq \frac{4mn^2}{\delta^2}$$

Proof. For a real $\varepsilon > 0$ let $\mathcal{F}_\varepsilon$ denote the event from Definition 6. Recall that all slopes of R differ by more than ε if $\mathcal{F}_\varepsilon$ does not occur. For $t \in \mathbb{R}$ and $\varepsilon > 0$ let $Z_{t,\varepsilon}$ be the random variable that indicates whether R has a slope in the interval $(t, t+\varepsilon]$ or not, i.e., $Z_{t,\varepsilon} = 1$ if $X_{(t,t+\varepsilon]} > 0$ and $Z_{t,\varepsilon} = 0$ if $X_{(t,t+\varepsilon]} = 0$.

Let $k \geq 1$ be an arbitrary integer. We subdivide the interval $(0,n]$ into k subintervals. If none of them contains more than one slope then the number $X_{(0,n]}$ of slopes in the

interval $(0, n]$ equals the number of subintervals for which the corresponding Z-variable equals 1. Formally

$$X_{(0,n]} \leq \begin{cases} \sum_{i=0}^{k-1} Z_{i\cdot\frac{n}{k},\frac{n}{k}} & \text{if } \mathcal{F}_{\frac{n}{k}} \text{ does not occur}\,, \\ m^n & \text{otherwise}\,. \end{cases}$$

This is true because $\binom{m}{n-1} \leq m^n$ is a worst-case bound on the number of edges of P and, hence, of the number of slopes of R. Consequently,

$$\begin{aligned} \mathbf{E}\left[X_{(0,n]}\right] &\leq \sum_{i=0}^{k-1} \mathbf{E}\left[Z_{i\cdot\frac{n}{k},\frac{n}{k}}\right] + \mathbf{Pr}\left[\mathcal{F}_{\frac{n}{k}}\right] \cdot m^n = \sum_{i=0}^{k-1} \mathbf{Pr}\left[A_{i\cdot\frac{n}{k},\frac{n}{k}}\right] + \mathbf{Pr}\left[\mathcal{F}_{\frac{n}{k}}\right] \cdot m^n \\ &\leq \sum_{i=0}^{k-1} \frac{2mn^2 \cdot \frac{n}{k}}{\frac{n}{2}\delta^2} + \mathbf{Pr}\left[\mathcal{F}_{\frac{n}{k}}\right] \cdot m^n = \frac{4mn^2}{\delta^2} + \mathbf{Pr}\left[\mathcal{F}_{\frac{n}{k}}\right] \cdot m^n\,. \end{aligned}$$

The second inequality stems from Lemma 12. Now the lemma follows because the bound on $\mathbf{E}\left[X_{(0,n]}\right]$ holds for any integer $k \geq 1$ and since $\mathbf{Pr}\left[\mathcal{F}_\varepsilon\right] \to 0$ for $\varepsilon \to 0$ in accordance with Lemma 7. ◀

In [4] we only computed an upper bound for the expected value of $X_{(0,1]}$. Then we argued that the same upper bound also holds for the expected value of $X_{(1,\infty)}$. In order to see this, we simply exchanged the order of the objective functions in the projection π. Then any edge with a slope of $s > 1$ becomes an edge with slope $\frac{1}{s} < 1$. Hence the number of slopes in $[1, \infty)$ equals the number of slopes in $(0, 1]$ in the scenario in which the objective functions are exchanged. Due to the symmetry in the choice of the objective functions in [4] the same analysis as before applies also to that scenario.

We will now also exchange the order of the objective functions $w^{\mathrm{T}}x$ and $c^{\mathrm{T}}x$ in the projection. Since these objective functions are not anymore generated by the same random experiment, a simple argument as in [4] is not possible anymore. Instead we have to go through the whole analysis again. We will use the superscript -1 to indicate that we are referring to the scenario in which the order of the objective functions is exchanged. In particular, we consider the events $\mathcal{F}_\varepsilon^{-1}$, $A_{t,\varepsilon}^{-1}$, and $E_{i,t,\varepsilon}^{-1}$ that are defined analogously to their counterparts without superscript except that the order of the objective functions is exchanged. The proof of the following lemma is analogous to the proof of Lemma 7.

▶ **Lemma 14.** *The probability of event $\mathcal{F}_\varepsilon^{-1}$ tends to 0 for $\varepsilon \to 0$.*

▶ **Lemma 15.** *For any $\phi \geq \sqrt{n}$, any $t \geq 0$, and any $\varepsilon > 0$ the probability of event $A_{t,\varepsilon}^{-1}$ is bounded by*

$$\mathbf{Pr}\left[A_{t,\varepsilon}^{-1}\right] \leq \frac{2mn^{3/2}\varepsilon\phi}{\max\left\{1, \frac{nt}{2}\right\} \cdot \delta} \leq \frac{2mn^{3/2}\varepsilon\phi}{\delta}\,.$$

▶ **Lemma 16.** *For any interval I let X_I^{-1} denote the number of slopes of $R_{w,c}$ that lie in the interval I. Then*

$$\mathbf{E}\left[X_{(0,1/n]}^{-1}\right] \leq \frac{2m\sqrt{n}\phi}{\delta}\,.$$

Proof. As in the proof of Lemma 13 we define for $t \in \mathbb{R}$ and $\varepsilon > 0$ the random variable $Z_{t,\varepsilon}^{-1}$ that indicates whether $R_{w,c}$ has a slope in the interval $(t, t + \varepsilon]$ or not. For any integer $k \geq 1$

we obtain

$$\begin{aligned}\mathbf{E}\left[X^{-1}_{\left(0,\frac{1}{n}\right]}\right] &\leq \sum_{i=0}^{k-1} \mathbf{E}\left[Z^{-1}_{i\cdot\frac{1}{kn},\frac{1}{kn}}\right] + \mathbf{Pr}\left[\mathcal{F}^{-1}_{\frac{1}{kn}}\right]\cdot m^n \\ &= \sum_{i=0}^{k-1} \mathbf{Pr}\left[A^{-1}_{i\cdot\frac{1}{kn},\frac{1}{kn}}\right] + \mathbf{Pr}\left[\mathcal{F}^{-1}_{\frac{1}{kn}}\right]\cdot m^n \\ &\leq \sum_{i=0}^{k-1} \frac{2mn^{3/2}\phi}{kn\delta} + \mathbf{Pr}\left[\mathcal{F}^{-1}_{\frac{1}{k2^\ell\sqrt{n}}}\right]\cdot m^n = \frac{2m\sqrt{n}\phi}{\delta} + \mathbf{Pr}\left[\mathcal{F}^{-1}_{\frac{1}{k2^\ell\sqrt{n}}}\right]\cdot m^n\,.\end{aligned}$$

The second inequality stems from Lemma 15. Now the lemma follows because the bound holds for any integer $k \geq 1$ and $\mathbf{Pr}\left[\mathcal{F}^{-1}_{\varepsilon}\right] \to 0$ for $\varepsilon \to 0$ in accordance with Lemma 14. ◀

The following corollary directly implies Theorem 4.

▶ **Corollary 17.** *The expected number of slopes of $R = R_{c,w}$ is*

$$\mathbf{E}\left[X_{(0,\infty)}\right] = \frac{4mn^2}{\delta^2} + \frac{2m\sqrt{n}\phi}{\delta}\,.$$

Proof. We divide the interval $(0,\infty)$ into the subintervals $(0,n]$ and (n,∞). Using Lemma 13, Lemma 16, and linearity of expectation we obtain

$$\begin{aligned}\mathbf{E}\left[X_{(0,\infty)}\right] &= \mathbf{E}\left[X_{(0,n]}\right] + \mathbf{E}\left[X_{(n,\infty)}\right] = \mathbf{E}\left[X_{(0,n]}\right] + \mathbf{E}\left[X^{-1}_{\left(0,\frac{1}{n}\right]}\right] \\ &\leq \frac{4mn^2}{\delta^2} + \frac{2m\sqrt{n}\phi}{\delta}\,.\end{aligned}$$

In the second step we have exploited that by definition $X_{(a,b)} = X^{-1}_{(1/b,1/a)}$ for any interval (a,b). ◀

4 Conclusions

We have shown that the shadow vertex algorithm can be used to solve linear programs possessing the δ-distance property in strongly polynomial time with respect to n, m, and $1/\delta$. The bound we obtained in Theorem 2 depends quadratically on $1/\delta$. Roughly speaking, one term $1/\delta$ is due to the fact that the smaller δ the less random is the objective function $w^{\mathrm{T}}x$. This term could in fact be replaced by $1/\delta(B)$ where B is the matrix that contains only the rows that are tight for x. The other term $1/\delta$ is due to our application of the principle of deferred decisions in the proof of Lemma 12. The smaller δ the less random is $w(Z)$.

For packing linear programs, in which all coefficients of A and b are non-negative and one has $x \geq 0$ as additional constraint, it is, for example, clear that $x = 0^n$ is a basic feasible solution. That is, one does not need to run Phase 1. Furthermore as in this solution without loss of generality exactly the constraints $x \geq 0$ are tight, $\delta(B) = 1$ and one occurrence of $1/\delta$ in Theorem 2 can be removed.

Acknowledgments The authors would like to thank Friedrich Eisenbrand and Santosh Vempala for providing detailed explanations of their paper and the anonymous reviewers for valuable suggestions how to improve the presentation.

References

1 Nicolas Bonifas, Marco Di Summa, Friedrich Eisenbrand, Nicolai Hähnle, and Martin Niemeier. On sub-determinants and the diameter of polyhedra. In *Proceedings of the 28th ACM Symposium on Computational Geometry (SoCG)*, pages 357–362, 2012.

2 Karl Heinz Borgwardt. *A probabilistic analysis of the simplex method*. Springer-Verlag New York, Inc., New York, NY, USA, 1986.

3 Tobias Brunsch, Anna Großwendt, and Heiko Röglin. Solving totally unimodular LPs with the shadow vertex algorithm. *CoRR*, abs/1412.5381, 2014.

4 Tobias Brunsch and Heiko Röglin. Finding short paths on polytopes by the shadow vertex algorithm. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 279–290, 2013.

5 Martin E. Dyer and Alan M. Frieze. Random walks, totally unimodular matrices, and a randomised dual simplex algorithm. *Mathematical Programming*, 64:1–16, 1994.

6 Friedrich Eisenbrand and Santosh Vempala. Geometric random edge. *CoRR*, abs/1404.1568, 2014.

7 I. Heller. On linear systems with integral valued solutions. *Pacific Journal of Mathematics*, 7(3):1351–1364, 1957.

8 Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

9 Éva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.