An Unconditional Lower Bound
for the Active-Set Method
on the Hypercube

Yann Disser and Nils Mosis$^{(\boxtimes)}$

TU Darmstadt, Darmstadt, Germany
{disser,mosis}@mathematik.tu-darmstadt.de

Abstract. The existence of a polynomial-time pivot rule for the simplex method is a fundamental open question in optimization. While many super-polynomial lower bounds exist for individual or very restricted classes of pivot rules, there currently is little hope for an unconditional lower bound that addresses all pivot rules. We approach this question by considering the active-set method as a natural generalization of the simplex method to non-linear objectives. This generalization allows us to prove the first unconditional lower bound *for all pivot rules*. More precisely, we construct a multivariate polynomial of degree linear in the number of dimensions such that the active-set method started in the origin visits all vertices of the hypercube. We hope that our framework serves as a starting point for a new angle of approach to understanding the complexity of the simplex method.

Keywords: simplex method · active-set method · lower bound · linear programming · non-linear programming

1 Introduction

The simplex method [12] is widely regarded as one of the most natural and important practical algorithms for solving linear programming problems. However, its theoretical complexity remains an open question to this day. This has to do with the fact that the simplex method's behavior is highly sensitive to the choice of pivot rule, which determines how ties between improving edges at a vertex are broken and significantly influences the method's behavior, such as whether it guarantees termination.

The discovery of a polynomial-time pivot rule for the simplex method would have profound implications. In particular, it would most likely provide a strongly polynomial algorithm for linear programming, thereby resolving the ninth open problem on Smale's list of mathematical problems for the 21st century [35]. While weakly polynomial algorithms for linear optimization are known, namely the ellipsoid and the interior-point method [27,38], and while the latter has recently been adapted to a strongly polynomial algorithm for linear programs

with at most two variables per constraint [2,10], barrier methods cannot yield strongly polynomial guarantees [3]. For linear programs with zero-one vertices strongly polynomial simplex pivot rules are known [8]. A general polynomial-time pivot rule for the simplex method would also imply a polynomial bound on the combinatorial diameter of polytopes, thus resolving the long-standing polynomial Hirsch conjecture [11,33].

Over the years, super-polynomial lower running time bounds were established for various natural pivot rules. Bounds for straightforward rules were found early-on by utilizing distorted hypercubes [5,7,24,26,29,31]. More recently, linear programs derived from Markov Decision Processes have been used to handle more sophisticated, memory-based rules [6,14,20–22]. In some cases, the exponential worst-case behavior of the simplex method could be attributed to its mightiness [1,17,18], which can be seen as a conditional lower bound. Nevertheless, results from smoothed complexity [9,13,25,36] suggest that the constructions underlying the above bounds are fragile, which indicates that fundamentally different approaches may be necessary to achieve an unconditional lower bound that applies universally across all pivot rules. While existing hypercube constructions were generalized to small classes of pivot rules [4,15], the existence of an unconditional super-polynomial lower bound remains wide open.

We approach this question by considering the active-set method [19,32] as a natural generalization of the simplex method to non-linear programs. A case of particular importance for the active-set method are quadratic, concave programs for which, much like in the linear setting, only weakly polynomial algorithms are known [38,39], and the active-set method is a promising candidate for the first strongly polynomial algorithm. Similarly to the simplex method, the active-set method is governed by a pivot rule that determines the improving direction in each step in case there is more than one option. Again, no general running time bounds for all pivot rules are known.

Our Results. We prove the first unconditional super-polynomial lower bounds on the running time of the active-set method *for all pivot rules*. More precisely, we show the following.

Theorem 1. *For all $n \in \mathbb{N}_{>2}$, there is a multivariate polynomial $\mathcal{F}_n$ of degree n such that the active-set method started in $\mathbf{0}$ needs $2^n - 1$ iterations to optimize $\mathcal{F}_n$ over the n-dimensional hypercube $[0, 1]^n$, irrespective of the pivot rule.*

In particular, our construction forces the active-set method to visit all vertices of the hypercube while traversing along its edges in a simplex-like fashion.

This result yields a new method of approach for unconditional lower bounds on the running time of the simplex method: If we could lower the degree of the polynomial to 1 by considering more general polytopes, we would obtain a super-polynomial (even exponential) lower bound for the simplex method using any pivot rule.

As a first step in this direction, we can lower the degree of the polynomial, while maintaining a super-polynomial bound, simply by neglecting most of the input dimensions.

Corollary 1. *For every $d(n) = \omega(\log n)$ with $d(n) \leq n$, there are polynomials $(\mathcal{F}_n)_{n \in \mathbb{N}}$ of degrees $\mathcal{O}(d(n))$ such that the active-set method started in $\mathbf{0}$ needs $2^{\omega(\log n)}$ iterations to optimize $\mathcal{F}_n$ over the n-dimensional hypercube $[0, 1]^n$, irrespective of the pivot rule.*

Importantly, we provide a single construction that addresses all pivot rules simultaneously (there is a unique improving direction in each step). This means that lowering the degree of the polynomial for general polytopes would constitute progress towards disproving the monotone polynomial Hirsch conjecture [40].

We observe that it is NP-complete to optimize polynomials of degree 3 over the hypercube (Proposition 5). This already implies a super-polynomial lower bound on the running time of the active-set method, assuming P ≠ NP. The value of our contribution lies in the fact that we provide an *unconditional* lower bound.

In fact, it turns out that the polynomials of Theorem 1 and Corollary 1 induce decomposable unique sink orientations of the hypercube (Proposition 6) that can be solved in linear time [34]. In particular, the polynomials in our construction do not give rise to NP-hard optimization problems (unless P = NP).

Notation. We write all vectors in boldface and denote the i-th unit vector by $\mathbf{e^i}$. For $x, y \in \mathbb{N} \cup \{0\}$, we write $x \equiv_2 y$ if both x and y are either even or odd. For $n \in \mathbb{N}$, we write $[n] =: \{1, \ldots, n\}$. We denote the i-th row of a matrix $A \in \mathbb{R}^{m \times n}$ by $\mathbf{A_i}$ and, for an index set $\mathcal{A} \subseteq [m]$, we write $A_{\mathcal{A}}$ to denote the matrix that consists of the rows $\mathbf{A_i}$ with $i \in \mathcal{A}$.

2   Simplex and Active-Set Method

The aim of this section is to illustrate that the active-set method is a natural generalization of the simplex method. This connection is well-known.

Proposition 1 (Exercise 8.17 in [19]). *The active-set method, when applied to linear objectives, is equivalent to the simplex method, i.e., the same intermediate solutions are computed during the application of both algorithms.*

Consider Algorithms 1 & 2 for a side-by-side comparison of SIMPLEX and ACTIVESET. Both algorithms are formulated in a way that makes their connection evident. We now describe the behavior of both algorithms in more detail.

Algorithm 1: SIMPLEX(c, A, b, x)

input: $c \in \mathbb{Q}^n$, vertex x of polytope $\mathcal{P} := \{y: Ay \leq b\}$, system $Ay \leq b$ non-degenerate

output: optimum solution x of $\max\{c^\top y: y \in \mathcal{P}\}$

$\mathcal{B} \leftarrow \mathrm{eq}(x)$ (basis)

while $\mathcal{D} := \{d: A_{\mathrm{eq}(x)}.d \leq 0,\ c^\top d > 0\} \neq \emptyset$:

   take $d \in \mathcal{D}$ with $|\{i \in \mathcal{B}: A_i.d = 0\}| = n - 1$

   if $A_{\mathcal{B}}.d \neq 0$:   (always true)

      $\mathcal{B} \leftarrow \mathcal{B} \setminus \{\ell\}$ for the unique $\ell \in \mathcal{B}$ with $A_\ell.d < 0$

   if $A_{\mathcal{B}}.d = 0$:   (always true)

      $\mu \leftarrow \inf\{\mu \geq 0: x + \mu d \notin \mathcal{P}\}$

      $x \leftarrow x + \mu d$

      if $c^\top d > 0$:   (always true)

         $\mathcal{B} \leftarrow \mathcal{B} \cup \{j\}$ for the unique $j \in \mathrm{eq}(x) \setminus \mathcal{B}$

Algorithm 2: ACTIVESET(f, A, b, x)

input: $f: \mathbb{R}^n \to \mathbb{R}$ continuously differentiable, point x of polytope $\mathcal{P} := \{y: Ay \leq b\}$

output: critical point x of $\max\{f(y): y \in \mathcal{P}\}$

$\mathcal{A} \leftarrow \mathrm{eq}(x)$ (active set)

while $\mathcal{D} := \{d: A_{\mathrm{eq}(x)}.d \leq 0,\ \nabla f(x)^\top d > 0\} \neq \emptyset$:

   take $d \in \mathcal{D}$ maximizing $|\{i \in \mathcal{A}: A_i.d = 0\}|$

   if $A_{\mathcal{A}}.d \neq 0$:

      $\mathcal{A} \leftarrow \mathcal{A} \setminus \{i\}$ for some $i \in \mathcal{A}$ with $A_i.d < 0$

   if $A_{\mathcal{A}}.d = 0$:

      $\mu \leftarrow \inf\{\mu \geq 0: x + \mu d \notin \mathcal{P}$ or $\nabla f(x + \mu d)^\top d \leq 0\}$

      $x \leftarrow x + \mu d$

      if $\nabla f(x)^\top d > 0$:

         $\mathcal{A} \leftarrow \mathcal{A} \cup \{j\}$ for some $j \in \mathrm{eq}(x) \setminus \mathcal{A}$

The simplex method solves arbitrary linear programs of the form

$$\max \quad \boldsymbol{c}^\top \boldsymbol{x}$$
$$\text{s.t.} \quad A\boldsymbol{x} \leq \boldsymbol{b}, \tag{LP}$$

where $\boldsymbol{c} \in \mathbb{Q}^n$, $A \in \mathbb{Q}^{m \times n}$, and $\boldsymbol{b} \in \mathbb{Q}^m$. For simplicity, we assume that the feasible region $\mathcal{P} := \{\boldsymbol{x} \colon A\boldsymbol{x} \leq \boldsymbol{b}\}$ is bounded, and that the constraints are non-degenerate, i.e., we assume that every vertex $\boldsymbol{x}$ of $\mathcal{P}$ satisfies $|\operatorname{eq}(\boldsymbol{x})| = n$, where we write $\operatorname{eq}(\boldsymbol{x}) := \{i \colon \boldsymbol{A_i}.\boldsymbol{x} = b_i\}$. The simplex method can be understood combinatorially as a traversal of $\mathcal{P}$ along its edges.

More precisely, it maintains a subset $\mathcal{B}$, called a *basis*, of n constraint indices such that the system $A_\mathcal{B}.\boldsymbol{x} = \boldsymbol{b}_\mathcal{B}$ uniquely determines the current vertex $\boldsymbol{x}$. By non-degeneracy, this is equivalent to $\mathcal{B} = \operatorname{eq}(\boldsymbol{x})$. As long as $\boldsymbol{x}$ is not already an optimum, the simplex method moves along edges of $\mathcal{P}$ to neighboring vertices with better objective function values. We now describe how our formulation of SIMPLEX reflects this behavior.

A vertex $\boldsymbol{x}$ of $\mathcal{P}$ is not optimal if and only if there is a direction $\boldsymbol{d}$ that is feasible and improving, i.e., that satisfies $A_{\operatorname{eq}(\boldsymbol{x})}.\boldsymbol{d} \leq \boldsymbol{0}$ and $\boldsymbol{c}^\top \boldsymbol{d} > 0$. In particular, there is at least one improving edge direction[1] in each vertex $\boldsymbol{x}$ that is not optimal. Due to non-degeneracy, every edge direction $\boldsymbol{d}$ corresponds to a unique index $\ell \in \mathcal{B}$ with $\boldsymbol{A_\ell}.\boldsymbol{d} < 0$ and $A_{\mathcal{B} \setminus \{\ell\}}.\boldsymbol{d} = \boldsymbol{0}$. In fact, a feasible direction $\boldsymbol{d}$ is an edge direction in $\boldsymbol{x}$ if and only if

$$|\{i \in \mathcal{B} \colon \boldsymbol{A_i}.\boldsymbol{d} = 0\}| = n - 1 = |\mathcal{B}| - 1.$$

Now, if the current vertex $\boldsymbol{x}$ is not optimal, SIMPLEX chooses an improving edge direction $\boldsymbol{d}$, deletes the corresponding index ℓ from $\mathcal{B}$, and moves along $\boldsymbol{d}$ as far as possible without becoming infeasible. Note that, by boundedness of $\mathcal{P}$, this transitions to another vertex $\boldsymbol{y}$ of $\mathcal{P}$. The method then adds a new active constraint index $j \in \operatorname{eq}(\boldsymbol{y})$ to $\mathcal{B}$ to obtain a basis uniquely identifying the new vertex. By non-degeneracy, this index is uniquely determined by $j \in \operatorname{eq}(\boldsymbol{y}) \setminus \mathcal{B}$.

The number of iterations of the simplex method depends on the *pivot rule* that determines the improving edge direction $\boldsymbol{d}$ chosen in each step in case there is more than one. The question of whether or not there exists a pivot rule guaranteeing a polynomial running time of the simplex method is arguably one of the most famous open problems in (linear) optimization. We refer to [11] for more details on the simplex method.

The active-set method can be applied to non-linear programs with linear constraints of the form

$$\max \quad f(\boldsymbol{x})$$
$$\text{s.t.} \quad A\boldsymbol{x} \leq \boldsymbol{b}, \tag{NLP}$$

where $f \colon \mathbb{R}^n \to \mathbb{R}$ is continously differentiable, $A \in \mathbb{Q}^{m \times n}$, and $\boldsymbol{b} \in \mathbb{Q}^m$. As before, we denote the feasible region by $\mathcal{P}$, and the set of indices of the constraints

---

[1] An edge direction in vertex $\boldsymbol{x}$ is a vector of the form $\boldsymbol{d} := \lambda(\boldsymbol{y} - \boldsymbol{x})$ where $\boldsymbol{y}$ is a neighboring vertex of $\boldsymbol{x}$ and $\lambda > 0$.

that are active in some $\boldsymbol{x} \in \mathcal{P}$ by eq($\boldsymbol{x}$). For simplicity, we again assume that $\mathcal{P}$ is bounded.

If $\boldsymbol{x}^*$ is a local optimum of (NLP), then there are no feasible improving directions in $\boldsymbol{x}^*$. That is, for all $\boldsymbol{d}$ with $A_{\mathrm{eq}(\boldsymbol{x}^*).}\boldsymbol{d} \leq \boldsymbol{0}$, we have $\boldsymbol{\nabla}\boldsymbol{f}(\boldsymbol{x}^*)^{\top}\boldsymbol{d} \leq 0$. The active-set method aims to find a *critical point*, i.e., a point satisfying this necessary optimality condition, by determining the set of constraints that are active in such a point.

More precisely, the active-set method maintains a subset $\mathcal{A} \subseteq$ eq($\boldsymbol{x}$), called *active set*, of constraints that are active in the current solution $\boldsymbol{x} \in \mathcal{P}$. As long as $\boldsymbol{x}$ is not already a critical point, the method tries to move along an improving feasible direction $\boldsymbol{d}$ that respects the active set, i.e., that satisfies $A_{\mathcal{A}.}\boldsymbol{d} = \boldsymbol{0}$. If there is no such direction, the method removes as few indices as possible from $\mathcal{A}$ until such a direction $\boldsymbol{d}$ can be found. Now it moves as far as possible along $\boldsymbol{d}$ without becoming infeasible, or until reaching a point $\boldsymbol{y}$ with $\boldsymbol{\nabla}\boldsymbol{f}(\boldsymbol{y})^{\top}\boldsymbol{d} \leq 0$. If this movement is stopped by hitting the boundary of $\mathcal{P}$, the method adds a new active constraint to $\mathcal{A}$. This procedure is iterated until a critical point for (NLP) is found. We now describe how our formulation of ACTIVESET allows this behavior.

A feasible point $\boldsymbol{x}$ is *not critical* if and only if there is a feasible improving direction $\boldsymbol{d}$, i.e., a direction $\boldsymbol{d}$ with $A_{\mathrm{eq}(\boldsymbol{x}).}\boldsymbol{d} \leq \boldsymbol{0}$ and $\boldsymbol{\nabla}\boldsymbol{f}(\boldsymbol{x})^{\top}\boldsymbol{d} > 0$. In ACTIVESET, we take such a direction $\boldsymbol{d}$ which also maximizes the number of indices $j \in \mathcal{A}$ with $\boldsymbol{A}_{j.}\boldsymbol{d} = 0$, and can then remove indices i from $\mathcal{A}$ with $\boldsymbol{A}_{i.}\boldsymbol{d} < 0$ (in separate iterations). This corresponds to deleting a smallest set $\mathcal{I} \subseteq \mathcal{A}$ from $\mathcal{A}$ that ensures the existence of a feasible improving direction $\boldsymbol{d}$ with $A_{\mathcal{A}\setminus\mathcal{I}.}\boldsymbol{d} = \boldsymbol{0}$. Now the method moves along the improving direction $\boldsymbol{d}$ as far as possible without becoming infeasible or reaching a point where the derivative of the objective in direction $\boldsymbol{d}$ is non-positive. If the directional derivative is still positive in the new point, the movement was stopped by hitting the boundary of $\mathcal{P}$, so the method adds a new active constraint to $\mathcal{A}$.

Implementations of the active-set method are usually formulated for strictly concave, quadratic objective functions (see e.g. [19, 32]), where they compute Karush-Kuhn-Tucker (KKT) points [28, 30] of equality constrained subproblems until reaching a KKT point, that is, the unique global optimum solution, of (NLP). More precisely, given a feasible solution $\boldsymbol{x}$ of (NLP) and an active set $\mathcal{A} \subseteq \{i \colon \boldsymbol{A}_{i.}\boldsymbol{x} = b_i\}$ of constraint indices, the method computes a KKT point $\boldsymbol{x}^*$ of

$$\begin{aligned} \max\ & f(\boldsymbol{y}) \\ \text{s.t.}\ & A_{\mathcal{A}.}\boldsymbol{y} = \boldsymbol{b}_{\mathcal{A}}. \end{aligned} \tag{NLP$_{\mathcal{A}}$}$$

Then, it moves from $\boldsymbol{x}$ towards $\boldsymbol{x}^*$ as far as possible without becoming infeasible. If it reaches $\boldsymbol{x}^*$, the method removes a constraint from the active set to enable further progress. Otherwise, it adds a new constraint to the active set.

Note that our formulation of ACTIVESET allows this behavior for strictly concave, quadratic functions. If the current solution $\boldsymbol{x}$ is not already the KKT point of (NLP$_{\mathcal{A}}$), the direction $\boldsymbol{d} := \boldsymbol{x}^* - \boldsymbol{x}$ is improving (by concavity of the

objective) and satisfies $A_{\mathcal{A}}.\boldsymbol{d} = \boldsymbol{0}$, i.e., maximizes $|\{i \in \mathcal{A} : \boldsymbol{A_{i.}d} = 0\}|$. Further, we have $\boldsymbol{x}^* = \boldsymbol{x} + \bar{\mu}\boldsymbol{d}$ for

$$\bar{\mu} := \inf\{\mu \geq 0 : \boldsymbol{\nabla f}(\boldsymbol{x} + \mu\boldsymbol{d})^\top \boldsymbol{d} \leq 0\}.$$

If $\boldsymbol{x}$ is the KKT point of $(\text{NLP}_{\mathcal{A}})$, then there is no feasible improving direction $\boldsymbol{d}$ with $A_{\mathcal{A}}.\boldsymbol{d} = \boldsymbol{0}$, so ACTIVESET removes constraints from $\mathcal{A}$ preventing progress.

While concave quadratic functions allow for an efficient implementation, the computations in an iteration of ACTIVESET are non-trivial for general functions. In the following, we assume that ACTIVESET has access to an oracle, which finds $\boldsymbol{d}$ and computes μ in each step.

The running time of the algorithm, i.e., the number of iterations, highly depends on the *pivot rule* that determines the direction $\boldsymbol{d}$ and the indices $i \in \mathcal{A}$ and $j \in \text{eq}(\boldsymbol{x}) \setminus \mathcal{A}$ in each iteration in case there is more than one choice. It is an open problem whether there is a polynomial time pivot rule for the active-set method for (strictly) concave, quadratic objective functions.

3   Unconditional Super-Polynomial Bounds for Active-Set

In Sect. 3.1, we define multivariate polynomials $\mathcal{F}_n : \mathbb{R}^n \to \mathbb{R}$ of degree n for all $n > 2$. We then see in Sect. 3.2 that there is exactly one improving edge in every vertex of the boolean hypercube $[0, 1]^n$ which is not optimal; and that the value of $\mathcal{F}_n$ increases when moving along this edge to the neighboring vertex. The path π obtained by starting in $\boldsymbol{0}$ and iteratively moving along unique improving edges to neighboring vertices visits all 2^n vertices of the hypercube, see Sect. 3.3. In Sect. 3.4, we obtain our main results by showing that ACTIVESET follows the path π. Finally, we make some remarks on the complexity of the optimization problem induced by our polynomials in Sect. 3.5.

3.1   Polynomials of Linear Degree

For each $n \in \mathbb{N}$, we define a multivariate polynomial $\mathcal{F}_n : \mathbb{R}^n \to \mathbb{R}$ by

$$\mathcal{F}_n(\boldsymbol{x}) = \sum_{i=1}^{n} \left(2^{i-1}\alpha_{n,i}(\boldsymbol{x}) - \beta_{n,i}(\boldsymbol{x})\right),$$

with $\boldsymbol{x} = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$ and $\alpha_{n,i}, \beta_{n,i} : \mathbb{R}^n \to \mathbb{R}$ given by $\alpha_{n,n+1}(\boldsymbol{x}) = 0$ and

$$\alpha_{n,i}(\boldsymbol{x}) = x_i + (1 - 2x_i)\alpha_{n,i+1}(\boldsymbol{x}), \qquad\qquad \forall\, i \in [n], \qquad (1)$$

$$\beta_{n,i}(\boldsymbol{x}) = 2^i \left(x_i - x_i^2\right)\left(1 - x_{i-1} + \sum_{j=1}^{i-2} x_j\right), \qquad \forall\, i \in [n],$$

where we write $x_0 := 1$, i.e., we have $\beta_{n,1}(\boldsymbol{x}) = 0$ for all $\boldsymbol{x}$. Whenever $\boldsymbol{x}$ and its dimension are clear from the context, we write $\alpha_i := \alpha_{n,i}(\boldsymbol{x})$ and $\beta_i := \beta_{n,i}(\boldsymbol{x})$. Note that $\mathcal{F}_n$ is a multivariate polynomial of degree n – except for $\mathcal{F}_2$ which is of degree 3.

We will analyze the behavior of ACTIVESET optimizing $\mathcal{F}_n$ on the hypercube $[0,1]^n$ starting in $\mathbf{0}$. As a preparation, we now compute the maximum of $\mathcal{F}_n$ on $[0,1]^n$ as well as the gradients of $\mathcal{F}_n$ in the vertices of the hypercube.[2]

Lemma 1. *For all $n \in \mathbb{N}$, the optimum solution of*

$$\begin{aligned} \max \quad & \mathcal{F}_n(\mathbf{x}) \\ s.t. \quad & \mathbf{x} \in [0,1]^n \end{aligned}$$

is attained by the unit vector $\mathbf{e}^n$.

Lemma 2. *For all $n \in \mathbb{N}$, $k \in [n]$, and $\mathbf{x} \in \{0,1\}^n$, the k-th partial derivative of $\mathcal{F}_n$ is given by*

$$\partial_k \mathcal{F}_n(\mathbf{x}) = (1 - 2\alpha_{k+1}) \sum_{i=1}^{k} 2^{i-1} \prod_{j=i}^{k-1} (1 - 2x_j) - 2^k (1 - 2x_k) \left(1 - x_{k-1} + \sum_{i=1}^{k-2} x_i \right),$$

where we write $x_0 := 1$, i.e., we have $\partial_1 \mathcal{F}_n(\mathbf{x}) = 1 - 2\alpha_2$.

3.2   Uniqueness of Improving Edges

We now show that in every vertex of the hypercube that is not optimal, there is exactly one improving edge direction.

Proposition 2. *For all $n \in \mathbb{N}$ and all $\mathbf{x} \in \{0,1\}^n \setminus \{\mathbf{e}^n\}$, there is exactly one $k \in [n]$ such that $\partial_k \mathcal{F}_n(\mathbf{x}) > 0$ and $x_k = 0$, or $\partial_k \mathcal{F}_n(\mathbf{x}) < 0$ and $x_k = 1$.*

Before giving the proof, we introduce notation tightening our further analysis.

Definition 1. *Given some $\mathbf{x} = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$ and $k \in [n]$, we write*

$$\mathcal{P}_n(\mathbf{x}, k) = x_{k-1} \prod_{j=1}^{k-2} (1 - x_j) \quad and \quad \mathcal{S}_n(\mathbf{x}, k) = \sum_{j=k+1}^{n} x_j,$$

where we set $x_0 := 1$, i.e., we have $\mathcal{P}_n(\mathbf{x}, 1) = 1$ for all $\mathbf{x}$. Whenever n and $\mathbf{x}$ are clear from the context, we write $\mathcal{P}(k) := \mathcal{P}_n(\mathbf{x}, k)$ and $\mathcal{S}(k) := \mathcal{S}_n(\mathbf{x}, k)$.

We now observe how the dimension k of Proposition 2 is determined by the values of $\mathcal{P}_n$ and $\mathcal{S}_n$.(see footnote 2)

Lemma 3. *The following are equivalent for all $\mathbf{x} \in \{0,1\}^n$ and $k \in [n]$:*

(i) $\partial_k \mathcal{F}_n(\mathbf{x}) > 0$ and $x_k = 0$, or $\partial_k \mathcal{F}_n(\mathbf{x}) < 0$ and $x_k = 1$,
(ii) $\mathcal{S}(k) \equiv_2 x_k$ and $\mathcal{P}(k) = 1$.

---

[2] All missing proofs are provided in the full version [16].

We obtain our proposition by showing that there is a unique dimension k that satisfies condition (ii) from Lemma 3 in each vertex that is not optimal.

Proof of Proposition 2. We assume $n \geq 3$, the cases $n \in \{1, 2\}$ can be checked directly. We start with two technical observations. First, fix some $\boldsymbol{x} \in \{0, 1\}^n$ and $1 < k_1 < k_2 \leq n$. We can easily see that $\mathcal{P}(k_1) = 1$ implies $x_{k_1 - 1} = 1$, while $\mathcal{P}(k_2) = 1$ implies $x_{k_1 - 1} = 0$. Since $\mathcal{P}(k) \in \{0, 1\}$ for all $k \in [n]$, this yields the following:

$$\forall \boldsymbol{x} \in \{0, 1\}^n, 1 < k_1 < k_2 \leq n \colon \mathcal{P}(k_1) = 0 \vee \mathcal{P}(k_2) = 0. \qquad (2)$$

Now fix some $\boldsymbol{x} \in \{0, 1\}^n$ with $\mathcal{S}(1) \equiv_2 x_1$. Assume that $\mathcal{P}(k) = 1$ holds for some fixed $k \in \{2, \ldots, n\}$. Then, we have $x_{k-1} = 1$ and $x_j = 0$ holds for all $j \in [k - 2]$. Thus, $\mathcal{S}(1) \equiv_2 x_1$ implies $\mathcal{S}(k) \equiv_2 x_k + 1$, which gives us the following:

$$\forall \boldsymbol{x} \in \{0, 1\}^n \text{ with } \mathcal{S}(1) \equiv_2 x_1, \ \forall k \in \{2, \ldots, n\} \colon \mathcal{P}(k) = 0 \vee \mathcal{S}(k) \equiv_2 x_k + 1. \quad (3)$$

For all $\boldsymbol{x} \in \{0, 1\}^n$, Eqs. (2) & (3) yield that we have $\mathcal{S}(k) \equiv_2 x_k$ and $\mathcal{P}(k) = 1$ for at most one $k \in [n]$.

Now consider some $\boldsymbol{x} \in \{0, 1\}^n \setminus \{\boldsymbol{e^n}\}$. It remains to argue that there exists some $k \in [n]$ with $\mathcal{S}(k) \equiv_2 x_k$ and $\mathcal{P}(k) = 1$. We may assume $\mathcal{S}(1) \equiv_2 x_1 + 1$ since $\mathcal{P}(1) = 1$. This yields the existence of some $j \in [n]$ with $x_j = 1$. Further, we have $k := \min\{j \in [n] \colon x_j = 1\} < n$ as $\boldsymbol{x} \neq \boldsymbol{e^n}$, hence $\mathcal{P}(k + 1) = 1$ and $\mathcal{S}(k + 1) \equiv_2 x_{k+1}$. The statement now follows directly by Lemma 3. □

Let k be the unique dimension from Proposition 2 for some vertex $\boldsymbol{x}$ that is not optimal. We show that the value of $\partial_k \mathcal{F}_n(\boldsymbol{x})$ does not depend on x_k.(see footnote 2)

Proposition 3. *Let $n \in \mathbb{N}$ and $\boldsymbol{x} \in \{0, 1\}^n \setminus \{\boldsymbol{e^n}\}$. Let $k \in [n]$ be the unique dimension from Proposition 2. Then, we have*

$$\partial_k \mathcal{F}_n(\boldsymbol{x}) = \partial_k \mathcal{F}_n(\boldsymbol{x} + \mu \boldsymbol{e^k})$$

for all $\mu \in \mathbb{R}$.

3.3   Visiting all Vertices of the Hypercube

We will see that ACTIVESET applied to $\mathcal{F}_n$ over the hypercube and starting in $\boldsymbol{0}$ moves from one vertex to another along the unique improving edges we identified in Sect. 3.2. This trajectory results in a Hamiltonian path in the polyhedral graph of $[0, 1]^n$, i.e., in a path along the edges of the hypercube that visits all of its 2^n vertices exactly once.

Proposition 4. *Let $n \in \mathbb{N}$. The sequence $(\boldsymbol{x^i})_{i \in [2^n]} \subseteq \{0,1\}^n$ given by $\boldsymbol{x^1} := \boldsymbol{0}$ and*

$$\boldsymbol{x^{i+1}} := \boldsymbol{x^i} + (1 - 2(x^i)_{k_i})\boldsymbol{e^{k_i}}, \quad \forall i \in [2^n - 1], \tag{4}$$

where $k_i \in [n]$ is the unique dimension of Proposition 2 in $\boldsymbol{x^i}$, forms a Hamiltonian path in the polyhedral graph of $[0,1]^n$ ending in $\boldsymbol{e^n}$.

Proof. By Lemma 3 and Proposition 2, the choice of $k_i \in [n]$ is indeed unique for $\boldsymbol{x^i} \in \{0,1\}^n \setminus \{\boldsymbol{e^n}\}$ and we have $\mathcal{P}(k_i) = 1$ and $\mathcal{S}(k_i) \equiv_2 (x^i)_{k_i}$. We prove the statement by induction on $n \in \mathbb{N}$. It obviously holds for $n = 1$, so assume it holds for some fixed $n \in \mathbb{N}$.

Consider the sequence $(\boldsymbol{x^i})_{i \in [2^{n+1}]} \subseteq \{0,1\}^{n+1}$ as defined in the statement. By induction, the subsequence $(\boldsymbol{x^i})_{i \in [2^n]}$ forms a Hamiltonian path in the polyhedral graph of $[0,1]^n$ ending in $\boldsymbol{x^{2^n}} = \boldsymbol{e^n}$, since the additional entry $(x^i)_{n+1} = 0$ does not change the value of $\mathcal{P}(k)$ or $\mathcal{S}(k)$ for any $k \in [n]$. Further, since we have $\mathcal{P}_{n+1}(\boldsymbol{e^n}, n+1) = 1$ and $\mathcal{S}_{n+1}(\boldsymbol{e^n}, n+1) = 0 = (\boldsymbol{e^n})_{n+1}$, it holds that $k_{2^n} = n + 1$ and thus $\boldsymbol{x^{2^n+1}} = \boldsymbol{e^n} + \boldsymbol{e^{n+1}}$.

For the remaining part of the sequence, note that having $(x^i)_{n+1} = 1$ instead of $(x^i)_{n+1} = 0$ in the additional dimension does not change $\mathcal{P}_{n+1}(\boldsymbol{x^i}, k)$ and flips the value of $(\mathcal{S}_{n+1}(\boldsymbol{x^i}, k) \mod 2) \in \{0,1\}$ for all $k \in [n]$.

By Eq. (4), for all $i \in [2^n - 1]$, we have $\mathcal{P}_{n+1}(\boldsymbol{x^i}, k_i) = \mathcal{P}_{n+1}(\boldsymbol{x^{i+1}}, k_i)$ and $\mathcal{S}_{n+1}(\boldsymbol{x^i}, k_i) = \mathcal{S}_{n+1}(\boldsymbol{x^{i+1}}, k_i)$. Therefore, the second half of the sequence is the inverse of the first half; more precisely, we have

$$(\boldsymbol{x^i})_{i \in \{2^n+1, 2^n+2, \dots, 2^{n+1}\}} = (\boldsymbol{y^k})_{k \in [2^n]},$$

where $\boldsymbol{y^k} := \boldsymbol{x^{2^n+1-k}} + \boldsymbol{e^{n+1}}$ for all $k \in [2^n]$. In particular, $(\boldsymbol{x^i})_{i \in [2^{n+1}]}$ forms a Hamiltonian path in the polyhedral graph of $[0,1]^{n+1}$ ending in $\boldsymbol{e^{n+1}}$. $\square$

3.4   Proofs of the Super-Polynomial Bounds

We define a matrix $C \in \{-1, 0, 1\}^{2n \times n}$ by $\boldsymbol{C_{i \cdot}} = (\boldsymbol{e^i})^\top$ and $\boldsymbol{C_{(i+n) \cdot}} = -(\boldsymbol{e^i})^\top$ for all $i \in [n]$, and a vector $\boldsymbol{c} \in \{0,1\}^{2n}$ by $c_i = 1$ and $c_{i+n} = 0$ for all $i \in [n]$. So we have

$$[0,1]^n = \{\boldsymbol{x} : C\boldsymbol{x} \le \boldsymbol{c}\}.$$

Fixing this implementation of the hypercube, we can now state our main result formally. For the proof, we combine the insights of the previous subsections.

Theorem 1. *For all $n \in \mathbb{N}$, ACTIVESET($\mathcal{F}_n, C, \boldsymbol{c}, \boldsymbol{0}$) solves*

$$\begin{aligned} \max \quad & \mathcal{F}_n(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in [0,1]^n \end{aligned}$$

in $2^n - 1$ iterations, irrespective of the pivot rule.

Proof. We show that ACTIVESET visits all vertices of the hypercube $[0,1]^n$. Let $(x^i)_{i\in[2^n]}$ denote the sequence of Proposition 4. The initial solution is given by $\mathbf{0} = x^1$. Now fix some $i \in [2^n - 1]$ and assume that x^i is the intermediate solution at the start of iteration i of ACTIVESET$(\mathcal{F}_n, C, c, \mathbf{0})$ and that the current active set is $\mathcal{A} = \mathrm{eq}(x^i) := \{j: C_{j.}x^i = c_j\}$. Note that, according to Proposition 4, we have $x^i \neq e^n$ so we can fix $k \in [n]$ as the unique dimension of Proposition 2 in the point x^i.

All edge directions in x^i are of the form

$$d^{j,\lambda} := \lambda(1 - 2(x^i)_j)e^j$$

for some $j \in [n]$ and some $\lambda > 0$. Note that $\partial_j \mathcal{F}_n = \nabla\mathcal{F}_n^\top e^j = -\nabla\mathcal{F}_n^\top(-e^j)$ for all $j \in [n]$, so we have $\nabla\mathcal{F}_n(x^i)^\top d^{j,\lambda} > 0$ if and only if $\partial_j \mathcal{F}_n(x^i) > 0$ and $(x^i)_j = 0$, or $\partial_j \mathcal{F}_n(x^i) < 0$ and $(x^i)_j = 1$. Hence, by Proposition 2, the improving edge directions in x^i are given by $d^{k,\lambda} = \lambda(1 - 2(x^i)_k)e^k$ for $\lambda > 0$.

Since the system $Cx \leq c$ is non-degenerate and $\mathcal{A} = \mathrm{eq}(x^i)$, every feasible direction d in the vertex x^i satisfies

$$|\{i \in \mathcal{A}: A_{i.}d = 0\}| = |\mathcal{A}| - 1$$

if and only if it is an edge direction in x^i. In particular, for every edge direction d in x^i, there is a unique index $\ell \in A$ with $A_\ell.d < 0$ and $A_{\mathcal{A}\setminus\{\ell\}.}d = \mathbf{0}$. Further, the system $A_{\mathcal{A}.}d = \mathbf{0}$ only has the trivial solution. Hence, taking an arbitrary feasible improving direction maximizing $|\{i \in \mathcal{A}: A_{i.}d = 0\}|$ is equivalent to taking an arbitrary improving edge direction.

This yields that ACTIVESET chooses the improving edge direction $d := d^{k,\lambda}$ for some fixed $\lambda > 0$ in iteration i and deletes the unique $\ell \in \mathcal{A}$ with $A_\ell.d < 0$. We have $A_{\mathcal{A}\setminus\{\ell\}.}d = \mathbf{0}$ and, by Proposition 3,

$$\nabla\mathcal{F}_n(x^i + \mu d)^\top d = \nabla\mathcal{F}_n(x^i)^\top d > 0$$

for all $\mu \in \mathbb{R}$. Hence, ACTIVESET moves from x^i along d in iteration i until reaching the next vertex

$$x^i + d = x^i + (1 - 2(x^i)_k)e^k = x^{i+1}.$$

Finally, since $\nabla\mathcal{F}_n(x^{i+1})^\top d > 0$, an index $j \in \mathrm{eq}(x^{i+1}) \setminus \mathcal{A}$ is added to $\mathcal{A}$. In fact, since the system $Cx \leq c$ is non-degenerate, this choice of j is unique, i.e., ACTIVESET starts iteration $i + 1$ in the point x^{i+1} with $\mathcal{A} = \mathrm{eq}(x^{i+1})$.

We have seen that ACTIVESET transitions from x^i to x^{i+1} in iteration i, where $i \in [2^n - 1]$ was chosen arbitrarily. This proves the statement since the optimum solution of the given program is $x^{2^n} = e^n$ by Lemma 1.   □

If we are willing to relax the exponential bound from Theorem 1, we can lower the degrees of the polynomials. In Corollary 1, the degrees are chosen just large enough to yield a super-polynomial bound.

Proof of Corollary 1. Let $d(n) = \omega(\log n)$ with $d(n) \leq n$ for all $n \in \mathbb{N}$. Since the additional dimensions do not affect its behavior, by Theorem 1, ActiveSet started in **0** solves

$$\max \quad \mathcal{F}_{\lfloor d(n) \rfloor}(x_1, \ldots, x_{\lfloor d(n) \rfloor})$$
$$\text{s.t.} \quad \boldsymbol{x} \in [0,1]^n$$

in $2^{\lfloor d(n) \rfloor} - 1$ iterations for all $n \in \mathbb{N}_{>2}$. Since $2^{\lfloor d(n) \rfloor} = 2^{\omega(\log n)}$ and the polynomials $\left(\mathcal{F}_{\lfloor d(n) \rfloor}\right)_{n \in \mathbb{N}}$ are of degrees $\lfloor d(n) \rfloor = \mathcal{O}(d(n))$, this concludes the proof. $\square$

3.5   The Complexity of Maximizing Over the Hypercube

The lower bounds in Theorem 1 and Corollary 1 are unconditional, meaning they show that ActiveSet is inefficient on the hypercube even if $\mathsf{P} = \mathsf{NP}$. It is folklore that the underlying decision problem is already NP-hard for polynomials of degree 3. We include a proof to make our presentation self-contained.(see footnote 2)

Proposition 5. *Given a multivariate polynomial* $f \colon \mathbb{R}^n \to \mathbb{R}$ *of degree* 3, *it is* NP-*complete to decide whether there is* $\boldsymbol{x} \in \{0,1\}^n$ *with* $f(\boldsymbol{x}) \geq 0$.

While the underlying problem is hard, the next statements show that our polynomial $\mathcal{F}_n$ belongs to a class of functions for which linear time algorithms exist that find the optimum vertex of the hypercube. We need some definitions before we can state this formally.

Every function $f \colon \mathbb{R}^n \to \mathbb{R}$ attaining unique values on the vertices of $[0,1]^n$ induces an orientation σ_f of the edges of the hypercube if we direct every edge $\{\boldsymbol{x}, \boldsymbol{y}\}$ from $\boldsymbol{x}$ to $\boldsymbol{y}$ if $f(\boldsymbol{x}) < f(\boldsymbol{y})$, and from $\boldsymbol{y}$ to $\boldsymbol{x}$ if $f(\boldsymbol{y}) < f(\boldsymbol{x})$. We write $\boldsymbol{x} \to_\sigma \boldsymbol{y}$ if edge $\{\boldsymbol{x}, \boldsymbol{y}\}$ is directed from $\boldsymbol{x}$ to $\boldsymbol{y}$ in σ.

An orientation σ of the edges of the hypercube is called *unique sink orientation* (USO) if there is exactly one *sink*, i.e., a vertex without outgoing edges, in every face of the hypercube. For more details on USOs, see e.g. [23,37].

We say that an orientation σ is *combed* if there is a dimension $i \in [n]$ such that all edges along dimension i point in the same direction, i.e., if we have

$$\boldsymbol{x} \to_\sigma \boldsymbol{x} + (1 - 2x_i)\boldsymbol{e^i}$$

for all $\boldsymbol{x} \in \{0,1\}^n$ with $x_i = \delta$, where $\delta \in \{0,1\}$ is fixed. An orientation is called *decomposable* if it is combed on every nonzero-dimensional subcube of $[0,1]^n$.(see footnote 2)

Proposition 6. *The function* $\mathcal{F}_n$ *induces a decomposable orientation of* $\{0,1\}^n$.

Every decomposable orientation is also a USO, and can be solved in linear time [34]. Hence, the polynomials of Theorem 1 and Corollary 1 do not induce NP-hard optimization problems (unless $\mathsf{P} = \mathsf{NP}$).

4   Future Research

In this paper, we considered the active-set method as a natural generalization of the simplex method in order to derive the first unconditional running time bounds for all pivot rules. A canonical next step towards a general bound for the simplex method would be to lower the degree of our polynomial.

Question 1. Is there a polynomial of degree $\mathcal{O}(\log n)$ over the hypercube for which active-set takes super-polynomially many iterations for all pivot rules?

It might be possible to reduce the complexity of the objective function even more drastically by increasing the complexity of the feasible region.

Question 2. Is there a polynomial of constant degree over some polytope for which active-set takes super-polynomially many iterations for all pivot rules?

While a general bound for linear objectives, i.e., for the simplex method, would be a massive breakthrough, reaching quadratic and concave objectives would already be very interesting: Much like in the linear setting, only weakly polynomial algorithms are known, namely ellipsoid and interior-point methods [38,39], and the active-set method is a promising candidate for a strongly polynomial algorithm, due to its combinatorial nature.

Question 3. Is there a concave, quadratic polynomial over some polytope for which active-set takes super-polynomially many iterations for all pivot rules?

References

1. Adler, I., Papadimitriou, C., Rubinstein, A.: On simplex pivoting rules and complexity theory. In: Proceedings of the 17th Conference on Integer Programming and Combinatorial Optimization (IPCO), pp. 13–24 (2014)
2. Allamigeon, X., Dadush, D., Loho, G., Natura, B., Végh, L.A.: Interior point methods are not worse than simplex. In: Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 267–277 (2022)
3. Allamigeon, X., Gaubert, S., Vandame, N.: No self-concordant barrier interior point method is strongly polynomial. In: Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC), pp. 515–528 (2022)
4. Amenta, N., Ziegler, G.M.: Deformed products and maximal shadows of polytopes. Contemp. Math. **223**, 57–90 (1999)
5. Avis, D., Chvátal, V.: Notes on Bland's pivoting rule. Polyhedral Combinatorics: Dedicated to the memory of D.R. Fulkerson, pp. 24–34 (1978)
6. Avis, D., Friedmann, O.: An exponential lower bound for Cunningham's rule. Math. Program. **161**, 271–305 (2017)
7. Black, A.E.: Exponential lower bounds for many pivot rules for the simplex method (2024). https://arxiv.org/abs/2403.04886, to appear in IPCO 2025
8. Black, A.E., De Loera, J.A., Kafer, S., Sanità, L.: On the simplex method for 0/1-polytopes. Math. Oper. Res. (2024)

9. Dadush, D., Huiberts, S.: A friendly smoothed analysis of the simplex method. In: Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC), pp. 390–403 (2018)

10. Dadush, D., Koh, Z.K., Natura, B., Olver, N., Végh, L.A.: A strongly polynomial algorithm for linear programs with at most two nonzero entries per row or column. In: Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC), pp. 1561–1572 (2024)

11. Dantzig, G.: Linear Programming and Extensions. Princeton University Press, Princeton (1963)

12. Dantzig, G., Orden, A., Wolfe, P.: The generalized simplex method for minimizing a linear form under linear inequality restraints. Pac. J. Math. **5**(2), 183–195 (1955)

13. Deshpande, A., Spielman, D.A.: Improved smoothed analysis of the shadow vertex simplex method. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 349–356 (2005)

14. Disser, Y., Friedmann, O., Hopp, A.V.: An exponential lower bound for Zadeh's pivot rule. Math. Program. **199**(1–2), 865–936 (2023)

15. Disser, Y., Mosis, N.: A unified worst case for classical simplex and policy iteration pivot rules. In: Proceedings of the 34th International Symposium on Algorithms and Computation (ISAAC), pp. 27:1–27:17 (2023)

16. Disser, Y., Mosis, N.: An unconditional lower bound for the active-set method on the hypercube (2025). https://arxiv.org/abs/2502.18019

17. Disser, Y., Skutella, M.: The simplex algorithm is NP-mighty. ACM Trans. Algorithms **15**(1), 1–19 (2018)

18. Fearnley, J., Savani, R.: The complexity of the simplex method. In: Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC), pp. 201–208 (2015)

19. Fletcher, R.: Practical Methods of Optimization. John Wiley & Sons, Hoboken (2000)

20. Friedmann, O.: A subexponential lower bound for Zadeh's pivoting rule for solving linear programs and games. In: Proceedings of the 15th Conference on Integer Programming and Combinatorial Optimization (IPCO), pp. 192–206 (2011)

21. Friedmann, O., Hansen, T.D., Zwick, U.: Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC), pp. 283–292 (2011)

22. Friedmann, O., Hansen, T.D., Zwick, U.: A subexponential lower bound for the random facet algorithm for parity games. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 202–216 (2011)

23. Gärtner, B., Schurr, I.: Linear programming and unique sink orientations. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 749–757. No. 26 (2006)

24. Goldfarb, D., Sit, W.Y.: Worst case behavior of the steepest edge simplex method. Discret. Appl. Math. **1**(4), 277–285 (1979)

25. Huiberts, S., Lee, Y.T., Zhang, X.: Upper and lower bounds on the smoothed complexity of the simplex method. In: Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC), pp. 1904–1917 (2023)

26. Jeroslow, R.G.: The simplex algorithm with the pivot rule of maximizing criterion improvement. Discret. Math. **4**(4), 367–377 (1973)

27. Karmarkar, N.: A new polynomial-time algorithm for linear programming. In: Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC), pp. 302–311 (1984)

28. Karush, W.: Minima of functions of several variables with inequalities as side conditions. In: Traces and emergence of nonlinear programming, pp. 217–245. Springer (2013)
29. Klee, V., Minty, G.J.: How good is the simplex algorithm? Inequalities **3**(3), 159–175 (1972)
30. Kuhn, H., Tucker, A.: Nonlinear programming. In: Proceedings of the 2nd Berkeley Symposium, pp. 481–492 (1951)
31. Murty, K.G.: Computational complexity of parametric linear programming. Math. Program. **19**(1), 213–219 (1980)
32. Nocedal, J., Wright, S.J.: Numerical optimization. Springer (1999)
33. Santos, F.: A counterexample to the Hirsch conjecture. Ann. Math. 383–412 (2012)
34. Schurr, I., Szabó, T.: Finding the sink takes some time: an almost quadratic lower bound for finding the sink of unique sink oriented cubes. Discret. Comput. Geom. **31**(4), 627–642 (2004)
35. Smale, S.: Mathematical problems for the next century. Math. Front. Perspect. 271–294 (2000)
36. Spielman, D.A., Teng, S.H.: Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. J. ACM **51**(3), 385–463 (2004)
37. Szabó, T., Welzl, E.: Unique sink orientations of cubes. In: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 547–555 (2001)
38. Tarasov, S.P., Khachiyan, L.G.: Polynomial solvability of convex quadratic programming. In: Doklady Akademii Nauk, vol. 248, pp. 1049–1051. Russian Academy of Sciences (1979)
39. Ye, Y., Tse, E.: An extension of Karmarkar's projective algorithm for convex quadratic programming. Math. Program. **44**, 157–179 (1989)
40. Ziegler, G.M.: Lectures on polytopes, vol. 152. Springer (2012)