A New Approach to Strongly Polynomial Linear Programming

Mihály Bárász[1] Santosh Vempala[2]
[1]Google, Zurich
[2]School of Computer Science, Georgia Tech, Atlanta
klao@cs.elte.hu vempala@gatech.edu

Abstract: We present an affine-invariant approach for solving linear programs. Unlike previous approaches, the potential strong polynomiality of the new approach does not require that graphs of polytopes have polynomial diameter (the Hirsch conjecture or weaker versions). We prove that two natural realizations of the approach work efficiently for *deformed products* [AZ99], a class of polytopes that generalizes all known difficult examples for variants of the simplex method, e.g., the Klee-Minty [KM72] and Goldfarb-Sit [GS79] cubes.

Keywords: Linear Programming, Affine-invariant Algorithms, Strongly Polynomial, Deformed Products

1 Introduction

Strongly polynomial linear programming has been a holy grail for the theory of algorithms for several decades. Notable milestones include strongly polynomial algorithms for maximum weight matchings in general graphs [Edm65], linear programming in fixed dimension [Meg84], and minimum cost flow [Tar86] and its extension to combinatorial linear programs. In addition to these breakthroughs, for several other problems and even special cases of these problems, there has been a drive to find combinatorial algorithms that reveal more structure and are possibly faster than their generic counterparts. Strong polynomiality is today both mathematically and algorithmically a central concept in complexity theory.

Linear programming is perhaps the most general setting that holds open the possibility of a strongly polynomial algorithm. Can we solve a standard instance,

$$\begin{aligned} \max \quad & c \cdot x \\ \text{s.t.} \quad & Ax \leq b \end{aligned}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ using at most $f(m,n)$ arithmetic operations with f being a bounded-degree polynomial with no dependence on the description of A, b, c? Two reasons why this possibility appears so tantalizing are that (a) if the program is feasible and bounded, there is a basic solution, i.e., it can be expressed succinctly as the solution to n of the inequalities as equalities, and (b) the complexity of the original LP algorithm, Simplex, and its many variants can be bounded as such a function; however, for all known deterministic variants (pivot rules), there are examples demonstrating that f has to be exponential in m or n.

All variants of the Simplex method maintain a basic feasible solution (n inequalities that define a vertex of the polyhedron $Ax \leq b$) and use a pivot rule to iteratively modify the current basic solution. There is an extensive body of work constructing difficult instances for pivot rules. These instances show that even though many pivot rules are guaranteed not to cycle, they end up exploring an exponential number of possibilities. On the other hand, the simplest randomized pivot rule — of all pivots (swapping one inequality from the current basis for another not in it) that improve the objective value, choose one at random — has been widely studies but not yet successfully analyzed. A lower bound of $\Omega(n^2)$ is known [GHZ94, BP07].

The major conceptual hurdle in proving a strongly polynomial bound for any variant of simplex (randomized or deterministic) is that this

would imply a polynomial bound on the diameter of any *polytope graph*, the graph induced by the vertices and edges of the polytope, sometimes called its skeleton. A long-standing open problem in combinatorics is the Hirsch conjecture: the diameter of any polytope graph for a polytope with m facets in $\mathbb{R}^n$ is at most $m - n$. The best known upper bound is superpolynomial [GK92]. The current best upper bounds on variants of the simplex method are subexponential, roughly $n \cdot 2^{\tilde{O}(\sqrt{d})}$ given by [Kal92] and Matousek et al [MSW96].

Is the possibility of strongly polynomial linear programming inextricably intertwined with resolving the Hirsch conjecture (or a polynomial version)? A very interesting lower bound [MS04] might suggest this: an abstract cube is the polytope graph of a cube with its edges oriented according to some simple rules, in particular that there is unique sink. Simplex naturally applies to optimization over abstract cubes. It has been shown, via specific orientations, that the random edge pivot rule above and several other powerful extensions are doomed to be exponential for abstract cubes.

What hope remains? The difficult orientations of abstract cubes are not geometrically realizable by explicit objective functions. In other words, objective functions map to only a subset of all possible orientations of edges of the polytope graph. Thus, it seems necessary to utilize the geometry of linear programs in any efficient algorithm. Indeed, all the known polynomial algorithms heavily use the geometry. A common high-level ingredient is some scaling of space (affine transformation) for efficiency. However, finding this transformation and making progress towards an optimal solution both depend on the description lengths of the polytope; although polynomial in the input, the number of arithmetic operations depends on the number of bits used to define the input instance.

The main contribution of this paper is an affine-invariant approach and affine-invariant geometric algorithm for solving linear programs. The algorithm is iterative and maintains a set of n inequalities, modifying this set in each iteration. However, unlike Simplex, it does not restrict itself to vertices and edges of the polytope; it typically follows rays in its facets or its interior, thus taking advantage of many geometric shortcuts. Moreover, unlike known geometric algorithms, it is affine-invariant and thus its complexity does not depend on the bit sizes of the input. Before we describe the algorithm precisely, we state what we prove about it so far.

We are able to analyze the algorithm for any polytope in the class of deformed products as defined by Amenta and Ziegler [AZ99]. This class includes all known difficult examples for variants of the simplex algorithm, e.g., the Klee-Minty cubes for Dantzig's largest coefficient rule [KM72], Jeroslow's construction [Jer73] for the greatest increase rule, the Goldfarb-Sit cubes [GS79] for the steepest increase rule, Avis-Chvatal's cubes [AC78] for Bland's rule and the construction by Murty [Mur80] and Goldfarb [Gol83] for the shadow vertex pivot rule. Our main theorem shows that our new algorithm takes $O(n^2)$ iterations on any polytope in a class generalizing all these difficult examples (each iteration takes $O(mn)$ arithmetic operations). We return to a discussion of the significance of this result and its implications in the concluding section.

2 Algorithm AFFINE

We propose the following algorithm for optimizing a linear objective over a simple polyhedron given by a system of linear inequalities. The algorithm maintains a set of n linear inequalities, whose normals are rows of the input constraint matrix A, but the right hand sides are not constrained to the input RHS vector b. Starting at a vertex, the algorithm computes the set of improving rays at the vertex, and a line given by a nonnegative combination of them. It moves along this line till it hits a facet. It then repeats the same step within the facet, to reach a lower-dimensional face and ultimately another vertex. Since it always moves along a combination of improving rays, the new vertex reached has objective value higher than the original vertex. This whole process is repeated till we reach a vertex with no improving rays.

In the above description, the part that is not clear is how to compute improving rays when we are at some point on a facet. To do this in an effective and affine-invariant manner, in each step, the algorithm updates the set of inequalities so that, when taken as equalities, their solution is the current point. We give more intuition following a precise description.

Algorithm: AFFINE
INPUT: Polyhedron P given by linear inequalities $\{a_j \cdot x \leq b_j : j = 1 \dots m\}$, objective vector c and a vertex z.
OUTPUT: A vertex maximizing the objective value, or "unbounded" if the LP is unbounded.

- While the current vertex z is not optimal, repeat:
  1. (Initialize)
     (a) Let H be the set of indices of active inequalities at z.
     (b) (Compute edges) For every $t \in H$ compute a vector $v_t : a_h \cdot v_t = 0$ for $h \in H \setminus t$ and $a_t \cdot v_t < 0$.
     (c) Let $T = \{t \in H : c \cdot v_t \geq 0\}$ and $S = H \setminus T$.
  2. (Iteration) While T is nonempty, repeat:
     (a) (Compute improving rays) For every $t \in T$ compute a vector $v_t \neq 0 : a_h \cdot v_t = 0$ for $h \in H \setminus \{t\}$, $c \cdot v_t \geq 0$ and the length of v_t is the largest value for which $z + v_t$ remains feasible.
     (b) (Pick direction) Invoke a subroutine which computes a nonnegative combination v of $\{v_t : t \in T\}$.
     (c) (Move) Let λ be maximal for which $z + \lambda v \in P$, if there is no such maximum, return "unbounded". Move the current point: $z := z + \lambda v$.
     (d) (Update inequalities) Let s be the index of an inequality which becomes active. Let $t \in T$ be any index such that $\{a_h : h \in \{s\} \cup S \cup T \setminus \{t\}\}$ is linearly independent. Set

$$S := S \cup \{s\}, \quad T := T \setminus \{t\} \text{ and } H := S \cup T.$$

- Return the current vertex.

For the subroutine in step (2b) we propose the following two possibilities:

Subroutine CENTROID
1. Let $\lambda_t = \max\{\lambda : z + \lambda_t v_t \in P\}, t \in I$.
2. Return $v = \sum_{t \in I} \lambda_t v_t$.

Subroutine RANDOM
1. Let $\lambda_t = \max\{\lambda : z + \lambda_t v_t \in P\}, t \in I$.
2. Chose $\{\mu_t : t \in I\}$ st. $\sum \mu_t = 1, \mu_t \geq 0 : t \in I$ uniformly at random.
3. Return $v = \sum_{t \in I} \mu_t \lambda_t v_t$.

Lemma 1. *Both proposed variants of Algorithm AFFINE are affine-invariant.*

We note that the inner loop which takes the algorithm from one vertex to another takes at most n iterations, since in each iteration, the cardinality of the set T is reduced by 1.

The main idea of the algorithm is to take geometric shortcuts through the interior of the polytope and not be restricted to its edges. At first sight, our algorithm might appear to be a modest generalization of the random edge pivot rule for Simplex — at a vertex of the feasible polyhedron, instead of picking an improving edge at random, we go along the average or a random combination of all the improving edges. This could indeed be the case at the first iteration starting at vertex, but then onwards our algorithm is typically not at a vertex; it moves from a point on a facet to another along a chord of the polyhedron. It does so by maintaining a set of hyperplanes whose intersection defines the current point. When the next direction is chosen (in an affine-invariant manner), it moves the point along with all the associated hyperplanes to the other endpoint of the chord, then replaces one of the hyperplanes with the facet just hit, so that the new set of hyperplanes defines the new point reached. This process is repeated.

It is natural to consider the following variant: after moving along a chord (random or centroid) from a vertex, we then jump to any vertex of objective value at least as high and repeat this process. What is the complexity of this (simpler) variant? If the algorithm could go to any improving vertex after following a chord, then for both the random rule and the centroid rule, we can construct instances where the total number of iterations is exponential. Thus, it is important to move from the endpoint of a chord to the next vertex (or next point reached) in a more systematic (in particular, affine-invariant) manner.

3 Preliminaries

We observe that the maximum number of iterations of the algorithm is at most n times the number of distinct vertices visited and the overall complexity is a fixed polynomial (time to compute improving rays) times the number of iterations. In our analysis, we will focus on bounding the number of vertices visited.

The following measures of complexity will be used to analyze the algorithm. Given a polytope P, an objective direction c, and a starting vertex z, let $f(P, c, z)$ is the (expected) maximum number of vertices visited by Algorithm AFFINE applied to P, c, z; let $f(P, c) = \sup_z f(P, c, z)$ and $f(P) = \sup_c f(P, c)$. We also define $h(P, c)$ is the maximum length of a directed path in the graph whose vertices and edges are vertices and edges of P with edges oriented in the direction of higher objective value; $h(P) = \sup_c h(P, c)$.

Let P be a k-dimensional polytope defined by the following inequalities:

$$P = \{x \in \mathbb{R}^k : Ax \leq a\}$$

Let V and W be l-dimensional combinatorially equivalent polytopes with corresponding facets parallel (*normally equivalent*). Let them be defined as follows:

$$V = \{x \in \mathbb{R}^l : Bx \leq b\}$$
$$W = \{x \in \mathbb{R}^l : Bx \leq b'\}$$

We assume that every inequality in the above definitions is essential.

Let φ be a k-dimensional linear functional, such that $\varphi(P) \subseteq [0, 1]$.

The deformed product [AZ99],

$$Q = P \rtimes_\varphi (V, W),$$

is defined as follows:

$$\{x \in \mathbb{R}^{k+l} : \exists y \in P, v \in V, w \in W$$
$$\text{s.t. } x = (y, \varphi(y)v + (1 - \varphi(y))w)\}$$

When $V = W$, we get the usual direct product, $Q = P \times V$. The Klee-Minty cube is obtained recursively, with P being a K-M cube in R^{n-1} and V, W are line segments (of different lengths. In Jeroslow's construction for the greatest increase rule, V, W are polygons in $\mathbb{R}^2$. All known bad examples for simplex pivot rules are recursively defined deformed products with $\dim V \leq 2$.

The next lemma collects useful properties of deformed products and is from [AZ99].

Lemma 2. *Deformed products have the following properties.*

1. *Q is combinatorially equivalent to $P \times V$.*

2. *Let q be a vertex of Q, where $q = (y, \varphi(y)v + (1-\varphi(y))w)$ with $y \in P$ and $v \in V$, $w \in W$. Then y, v and w are uniquely determined and are vertices of the corresponding polytopes. Moreover v and w are corresponding vertices of V, W.*
   We use the following notation for this decomposition: $\pi_P(q) = y$, $\pi_V(q) = v$, $\pi_W(q) = w$.
3. *Let*

$$C = \begin{pmatrix} A & 0 \\ F & B \end{pmatrix}, \quad c = \begin{pmatrix} a \\ b \end{pmatrix}$$

   where $F = \operatorname{diag}(b - b')\varphi$. Then

$$Q = \{x \in \mathbb{R}^n : Cx \leq c\}$$

   Every inequality in this formulation is essential. We refer to the facets defined by the rows of A as P-facets, and the rest as (V, W)-facets.

4 Analysis

4.1 Deformed products

The main theorem of this section is the following.

Theorem 1. *Let P, V, W, φ be as in the definition of the deformed product. Let $Q := P \rtimes_\varphi (V, W)$. Then*

$$f(Q) \leq h(V) + f(P).$$

The next corollary shows that Algorithm AFFINE is efficient on all known bad examples for simplex pivot rules.

Corollary 1. *Let Q be a recursively defined deformed product polytope, where at every step of the recursion, $\dim(V) \leq 2$. Then*

$$f(Q) \leq \dim(Q) + \#\{\textit{facets of } Q\}.$$

To prove the theorem, we need the following definitions.

Definition 1. *For a polyhedron Q, a* defining hyperplane *is any hyperplane whose normal vector is the same as the normal vector of one of the inequalities defining Q.*

Definition 2. *Let $Q = P \rtimes_\varphi (V, W)$, $n = \dim Q$. We call an intersection of $n - 1$ defining hyperplanes a P-ray if there are at most $\dim P - 1$ hyperplanes corresponding to inequalities of P among them. Otherwise we call a it (V, W)-ray. That is, there are at most $\dim V - 1$ hyperplanes corresponding to (V, W) inequalities defining the intersection.*

Lemma 3. *The first $\dim P$ coordinates of a (V, W)-ray are 0.*

Proof. Let $x = (\pi_P(x), \varphi(\pi_P(x))\pi_V(x) + (1 - \varphi(\pi_P(x))\pi_W(x))$ be a point on a (V, W) ray. Then $y = \pi_P(x)$ satisfies $\dim P$ linearly independent inequalities as equalities from the set $Ay \leq a$ defining P. Thus y is uniquely defined for all points x along the ray, and the difference between two points (a vector along the ray) is zero along the first $\dim P$ coordinates. □

Lemma 4. *Let x and y be two consecutive vertices visited by the algorithm applied to $Q := P \rtimes_\varphi (V, W)$. Then $\pi_V(x) < \pi_V(y)$ in the partial order induced on the vertices of V by c_V, unless $\pi_V(x)$ is already maximal in the partial order.*

Proof. The partial order is the same on V and W and all their "copies" in the deformed product. We can write the objective vector $c = (c_P, c_{V,W})$ and assume that $c_{V,W}$ is not entirely zero. ¿From a vertex x, the algorithm only chooses a nonnegative combination of improving rays to move. If these rays do not include any (V, W)-ray, then $\pi_V(x)$ is already maximal. If some (V, W) rays are included, then with probability 1, their coefficient is positive and so the next point reached will have higher objective value overall and higher value w.r.t. to $c_{V,W}$ as well. Since the algorithm only uses improving directions, the next vertex reached will be higher in the partial order. □

Proof. (of Theorem 1.) By Lemma 4, the number of vertices of Q visited before $\pi_V(z)$ is maximal wrt. c_V in V, is at most $h(V)$. After reaching a vertex z, for which $\pi_V(z)$ is maximal, by Lemma 3 the V, W-rays are never improving. So, the algorithm proceeds as if in P, and finishes in at most $f(P)$ additional visits to vertices. □

4.2 Direct Products

To analyze a direct product of two arbitrary simple polytopes, we define an extended complexity measure.

For the purpose of analysis, consider the following version of the algorithm.

Algorithm: PROJECTED AFFINE

...

After step (2b) insert the following steps arbitrary many times.

i. Chose an arbitrary λ st. $z + \lambda v \in P$. Move the current point there.

ii. Recompute λ_ts with the subroutine and let $v = \sum_{t \in I} \lambda_t v_t$.

In words, as the algorithm moves along a chosen line, it can stop at any point and recompute a new line to move along using the same subroutine. It can do this arbitrarily many times.

Let $g(P, c, z)$ be the (expected) maximum number of vertices visited by Algorithm PROJECTED AFFINE on input polytope P with objective vector c and starting point z. Thus, the inserted steps can affect which vertices are visited but are *not* counted separately in the complexity of the algorithm.

Theorem 2. *Let P and Q be two polyhedra in $\mathbb{R}^{\dim P}$ and $\mathbb{R}^{\dim Q}$ respectively with $z_P \in P, z_Q \in Q$, vertices in P, Q respectively and $c_P \in \mathbb{R}^{\dim P}$, $c_Q \in \mathbb{R}^{\dim Q}$ vectors in the corresponding spaces. Let $z = (z_P, z_Q)$ and $c = (c_P, c_Q)$. Then,*

$$\begin{aligned} f(P \times Q, c, z) &\leq g(P \times Q, c, z) \\ &\leq g(P, c_P, z_P) + g(Q, c_Q, z_Q). \end{aligned}$$

Proof. Any vertex x of $P \times Q$ can be written as $x = (x_P, x_Q)$ where x_P, x_Q are vertices of P and Q respectively. Let x and y be two consecutive vertices visited by Algorithm AFFINE . Let $n = \dim P + \dim Q$. When the algorithm moves from x along a line, it hits a facet which is either a P-facet (corresponding to an inequality defining P) or a Q-facet. Let the point reached be x^1 and the facet hit be a P-facet. Then $x^1 = (x_P^1, x_Q^1)$. If the algorithm were applied directly on P from x_P, then we would reach the same (distribution for) x_P^1. The centroid subroutine would generate the same subcombination of rays, and the random subroutine would have the same distribution on lines generated. The same is true for Q, except that the step is not completed, i.e., the algorithm stops in Q before reaching a facet. Taking the next step in $P \times Q$ from x^1 can be viewed as attempting a step of Algorithm AFFINE in P from x_P^1 and Algorithm PROJECTED AFFINE in Q from x_Q^1. Within n iterations we reach a vertex $y = (y_P, y_Q)$ of $P \times Q$. Thus, in the steps from x to y, we also move from a vertex to a vertex in both P and Q, however, for one step in $P \times Q$, we are guaranteed to take a "complete" step in only one of P, Q. In the other one, we perform one of the "inserted" steps. Therefore we can view the algorithm in $P \times Q$ as running Algorithm PROJECTED AFFINE in both P and Q, coordinated in a specific way. The complexity bound follows. □

We believe that the analysis of deformed products in Theorem 1 can be improved using Algorithm PROJECTED AFFINE as done in this section.

4.3 Perturbed products

Here we argue that the analysis of the previous section is not delicately aligned with the structure of products and deformed products. We do this by showing that we can perturb the facets defining a product polytope, and for small enough perturbations, the edges of the polytope can change their orientations with respect to the objective function, but our analysis still holds.

For simplicity consider a direct product polytope $Q = P \times V$. We know that every vertex q of Q can be written as $q = (p, v)$ where p and v are vertices of P and V respectively. Now suppose the facets of P and V by small but arbitrary perturbations of each coefficient of each facet normal. Further assume that the perturbation is at most ϵ in magnitude and ϵ is small enough that in the perturbed polytope Q', each vertex q' is the solution of exactly $\dim P$ facets that are perturbations of facets of P and $\dim V$ facets that are perturbations of facets of V. Then many edges can change their orientation with respect to the objective function. However, our analysis using Algorithm PROJECTED AFFINE is still valid and we still get a bound of $g(Q) \leq g(P) + g(V)$. Essentially the same reasoning also applies to deformed products.

5 Discussion

We have presented a new approach to solving linear programs and two algorithmic realizations of it. The highlights of the method are (a) it takes geometric shortcuts through the input polyhedron and (b) it is affine-invariant. As a result, its complexity is not related to the Hirsch conjecture and is not dependent on the bit sizes of the input. As an illustration, suppose the input is a rotated cube stretched along one of its axes. Then the complexity of known polynomial-time algorithms for linear programming would depend on the stretch factor, i.e., the number of bits of the long axis. To analyze our algorithm, we first note that we can equivalently analyze it on any affine transformation of the input, in particular the one that brings it back to a cube and thus it is independent of bit sizes.

In this paper, we have focussed our analysis on bad instances for the simplex method. These instances have been constructed over decades and show that known deterministic pivoting rules for the simplex method are exponential. Fortunately, none of these instances poses a problem for our new algorithm. It would be very interesting to extend the analysis to combinatorial cubes, i.e., polytopes whose face structure is identical to that of the cube.

One can construct classes of polytopes which have the property that the centroid rule (or random rule) makes significant progress towards the optimum in each step even though the lengths of edges are small, e.g., triangulations of the sphere where the edges are all roughly the same length. One direction of future research to make this more precise and more general would be investigate the behavior of our algorithm on random polytopes. For example, what is the complexity of Algorithm AFFINE on polytopes of the form $Ax \leq 1$ where the rows of A are random unit vectors (and so the polytope contains the unit ball).

We conclude the paper with a loose, heuristic argument for analyzing the algorithm in the general case. Assume the input is a polytope P and we are at a vertex v. Using the affine-invariance of the algorithm, we can assume for the sake of analysis that the set $P_v = P \cap \{x : c \cdot x \geq c \cdot v\}$ is in isotropic position, i.e., its covariance matrix is the identity and it is centered at the origin. Then the set P_v contains a unit ball and it appears plausible that a random combination of the rays of v generates a chord through v that is likely to intersect this ball. If it does, then moving along the chord, we make significant progress towards the optimum, roughly reducing the distance to optimal by a $(1-1/n)$ factor. This would yield a polynomial bound, which again by affine-invariance should give a strongly polynomial bound.

Acknowledgements. We thank Luis Rademacher for many helpful discussions on this topic. The first author was supported in part by the Algorithms and Randomness Center (ARC) at Georgia Tech and the second author acknowledges NSF Award CCF-0721503.

References

[AC78] D. Avis and V. Chvtal, *Notes on bland's pivoting rule*, Polyhedral Combinatorics. Math. Programming Study **8** (1978), 24–34.

[AZ99] Nina Amenta and Gnter M. Ziegler, *Deformed products and maximal shadows of polytopes*, Advances in Discrete and Computational Geometry, Contemporary Mathematics **223** (1999), 57–90.

[BP07] József Balogh and Robin Pemantle, *The klee–minty random edge chain moves with linear speed*, Random Struct. Algorithms **30** (2007), no. 4, 464–483.

[Edm65] J. Edmonds, *Paths, trees, and flowers*, Canadian Journal on Mathematics **17** (1965), 449–467.

[GHZ94] B. Gärtner, M. Henk, and G. M. Ziegler, *Randomized simplex algorithms on Klee-Minty cubes*, Combinatorica **18** (1998 (preliminary version at FOCS'94)), no. 3, 349–372.

[GK92] D.J. Kleitman G. Kalai, *A quasi-polynomial bound for the diameter of graphs of polyhedra*, Bull. Amer. Math. Soc. (1992), 315–316.

[Gol83] D. Goldfarb, *Worst case complexity of the shadow vertex simplex algorithm*, Tech Rep. Columbia Univ. (1983).

[GS79] D. Goldfarb and W. T. Sit, *Worst case behaviour of the steepest edge simplex method*, Disc. Appl. Math **1** (1979), 277–285.

[Jer73] R. G. Jeroslow, *The simplex algorithm with the pivot rule of maximizing improvement criterion*, Disc. Math. **4** (1973), 367–377.

[Kal92] Gil Kalai, *A subexponential randomized simplex algorithm (extended abstract)*, STOC, 1992, pp. 475–482.

[KM72] V. Klee and G. J. Minty, *How good is the simplex algorithm?*, p. 159175, Academic Press, New York, 1972.

[Meg84] Nimrod Megiddo, *Linear programming in linear time when the dimension is fixed*, J. ACM **31** (1984), no. 1, 114–127.

[MS04] Jirí Matousek and Tibor Szabó, *Random edge can be exponential on abstract cubes*, FOCS, 2004, pp. 92–100.

[MSW96] Jirí Matousek, Micha Sharir, and Emo Welzl, *A subexponential bound for linear programming*, Algorithmica **16** (1996), no. 4/5, 498–516.

[Mur80] K. G. Murty, *Computational complexity of parametric linear programming*, Math. Programming **19** (1980), 213–219.

[Tar86] Eva Tardos, *A strongly polynomial algorithm to solve combinatorial linear programs*, Oper. Res. **34** (1986), no. 2, 250–256.