CS 435, 2018
Lecture 9, Date: 3 May 2018
Instructor: Nisheeth Vishnoi

Cutting Plane and Ellipsoid Methods for Linear Programming

In this lecture we introduce the class of cutting plane methods for convex optimization and present an analysis of a special case of it: the ellipsoid method. We then show how to use this ellipsoid method to solve linear programs with exponentially many constraints.

Contents

1 Optimizing over 0-1 polytopes with exponentially many constraints

1.1 Combinatorial Optimization and 0-1 polytopes

In combinatorial optimization one often deals with discrete optimization problems of the following type: given a set family $\mathcal{F} \subseteq 2^{[n]}$ (i.e., a collection of subsets of $[n] := \{1, 2, \ldots, n\}$) and a cost vector $c \in \mathbb{Z}^n$ find the minimum cost set $S \in \mathcal{F}$. Formally this can be described as

Discrete linear optimization problem

Input: A family of subsets $\mathcal{F} \subseteq 2^{[n]}$, a cost vector $c \in \mathbb{Z}^n$

Goal: Find a set $S^\star$ such that

$$S^\star = \operatorname*{argmin}_{S \in \mathcal{F}} \sum_{i \in S} c_i.$$

In fact, many familiar problems on graphs can be formulated in this setting. For instance, we obtain the maximum cardinality matching problem by taking $\mathcal{F} \subseteq 2^E$ to be the set of all matchings in a graph $G = (V, E)$ and letting $c = -1$, the negative of the all-ones vector of length $|E|$. Similarly, one can obtain the minimum spanning tree problem, or the problem of maximum flow (for unit capacity graphs), or minimum cut.

When trying to tackle these problems using continuous techniques, such as the ones introduced in this course, a useful idea is to first make the domain into a "continuous" (and ideally convex) space. More specifically, in this particular setting, given a family $\mathcal{F} \subseteq 2^{[n]}$ one can define a polytope $P_\mathcal{F}$ being the "convex hull" of $\mathcal{F}$

$$P_\mathcal{F} := \operatorname{conv}\{1_S : S \in \mathcal{F}\},$$

where $1_S \in \{0,1\}^n$ is the indicator vector of the set $S \subseteq [n]$. Such polytopes are called **0-1 polytopes** as all their vertices belong to the set $\{0,1\}^n$. An analogue of a discrete optimization setting for such polytopes is as follows.

Linear optimization problem over 0-1 polytopes

Input: A 0-1 polytope $P \subseteq [0,1]^n$, a cost vector $c \in \mathbb{Z}^n$

Goal: Find a vector $x^\star \in P$ such that

$$x^\star = \operatorname*{argmin}_{x \in P} c^\top x.$$

Note now that by solving a linear optimization problem over $P_\mathcal{F}$ for the cost vector c we

can solve the corresponding disrecte optimization problem over $\mathcal{F}$. Indeed, since[1] the optimal solution is always attained at a vertex, say $x^\star = 1_S$, then S is the optimal solution to the discrete problem and vice versa.

For concreteness, let us now introduce several important examples of 0-1 polytopes involving graphs.

Definition 1. *For an undirected graph $G = (V, E)$ with n vertices and m edges we define*

1. *The matching polytope $P_M(G) \subseteq [0,1]^m$ to be*

$$P_M(G) := \text{conv}\left\{1_S : S \subseteq [m] \text{ is a matching in } G\right\}.$$

2. *The perfect matching polytope $P_{PM}(G) \subseteq [0,1]^m$ to be*

$$P_{PM}(G) := \text{conv}\left\{1_S : S \subseteq [m] \text{ is a perfect matching in } G\right\}.$$

3. *The spanning tree polytope $P_{PM}(G) \subseteq [0,1]^m$ to be*

$$P_{ST}(G) := \text{conv}\left\{1_S : S \subseteq [m] \text{ is a spanning tree in } G\right\}.$$

We also refer to [11] for more examples of polytopes related to various combinatorial objects. Now, for instance, solving the maximum matching problem over G reduces to the problem of minimizing a linear function over $P_M(G)$. Since this is a certain form of a linear program, one could ask: can we solve these linear programs using the interior point method?

1.2 The problem with applying IPM to the matching polytope

Unfortunately, as it turns out, the polytope $P_M(G)$ does not have a particularly simple structure and requires exponentially many inequalities in its description.

Theorem 2 (Facet Description of the Matching Polytope [3])**.** *Let $G = (V, E)$ be an undirected graph with n vertices and m edges, then*

$$P_M(G) := \left\{ x \in [0,1]^m : \sum_{i \in S} x_i \leq \frac{|S|-1}{2} \text{ for all subsets } S \subseteq [m] \text{ of odd cardinality} \right\}.$$

Thus in other words, every odd subset $S \subseteq [m]$ induces a linear constraint $\sum_{i \in S} x_i \leq \frac{|S|-1}{2}$ and the intersection of all these constraints defines the polytope $P_M(G)$. This provides us with a description of $P_M(G)$, yet using an exponential number of linear constraints.

Similar descriptions are available for $P_{ST}(G)$ and $P_{PM}(G)$, and while there are results showing that there are efficient ways to represent $P_{ST}(G)$ using polynomially many variables and inequalities, for the matching polytope it has been proved [9] that essentially any representation of $P_M(G)$ requires exponentially many constraints.

[1] Here we assume for simplicity that the optimal solution is unique.

For this reason, applying an algorithm based on the path following method to minimize over $P_M(G)$ is rather problematic. Indeed, to do that, one is required to provide access to an "efficient" self-concordant barrier function for $P_M(G)$. Clearly, since the logarithmic barrier would have an exponential number of terms here, we cannot apply it directly. Similarly, computing either the volumetric barrier or the Lee-Sidford barrier, requires time at least polynomial in the number of constraints and thus is not acceptable, when looking for a polynomial-time (in m) algorithm. It is still an open question to derive a self-concordant barrier function for $P_M(G)$ that would be computationally tractable.

More precisely, one can ask: is there a self-concordant barrier function $F : P_M(G) \to \mathbb{R}$ with complexity parameter polynomial in m such that the gradient $\nabla F(x)$ and the Hessian $\nabla^2 F(x)$ are efficiently computable? By efficiently computable we mean: polynomial time in m, but apart from that also the computation should be "simpler" than solving the linear program itself. In fact, it has been observed in [1] that computing gradients and Hessians of the entropic barrier (whose complexity parameter is $O(m)$ [2]) reduces to sampling points from a certain log-concave distribution over $P_M(G)$ and thus by a result of [7] can be approximated up to a desired precision in polynomial time. Note however that since sampling from polytopes is generally a harder task than optimizing over them, this result is not of practical relevance for interior point methods. Finding an efficient barrier (whose evaluation does not reduce to harder problems) is an open problem.

1.3 Efficient separation oracles

Given the discussion in the previous section it seems that directly optimizing linear function over $P_M(G)$ might be a hard task (at least using the interior point method), and thus one would perhaps try to start with something simpler. For instance, while the problem of checking if a discrete set of edges forms a matching or not is rather trivial, its continuous counterpart: checking if $x \in P_M(G)$ is already challenging. More generally, one can state the following separation problem (already introduced in Lecture 2)

Separation problem for polytopes

Input: A polytope $P \subseteq \mathbb{R}^n$, a vector $x \in \mathbb{Q}^n$

Goal: Output one of the following answers

- if $x \in P$ output YES
- if $x \notin P$ output NO and provide a vector $h \in \mathbb{Q}^n \setminus \{0\}$ such that

$$\forall y \in P \quad \langle h, y \rangle < \langle h, x \rangle .$$

Recall that in the "NO" case, the vector h defines a hyperplane separating x from the polytope P. An important set of results in combinatorial optimization is that for many

"combinatorial" 0-1 polytopes, the separation problem can be solved in polynomial time[2].

Theorem 3 (Separation for combinatorial polytopes [4])**.** *There are polynomial time separation oracles for the polytopes $P_M(G)$, $P_{PM}(G)$ and $P_{ST}(G)$.*

Instead of providing here a proof of Theorem 3 we sketch it in one of the exercises for this lecture. The question which now arises naturally is: given a separation oracle for $P_M(G)$ does it allow us to optimize linear functions over this polytope? The main result of this lecture is a theorem that says that this is the case, not only for the matching polytope, but for all 0-1 polytopes.

Theorem 4 (Efficient optimization using a separation oracle)**.** *There is an algorithm that given a separation oracle for a 0-1 polytope $P \subseteq [0,1]^n$ and a vector $c \in \mathbb{Z}^n$ outputs*

$$x^\star \in \operatorname*{argmin}_{x \in P} c^\top x.$$

The algorithm runs in polynomial time in n, $L(c)$ and makes a polynomial number of queries to the separation oracle of P.

Combining the above with the existence of an efficient separation oracle for $P_M(G)$ (Theorem 3) we conclude in particular that the problem of computing the maximum cardinality matching in a graph is polynomial-time solvable. Similarly, using the above, one can compute a matching (or a perfect matching, or a spanning tree) of maximum or minimum weight in polynomial time. The remaining part of the lecture is devoted to the proof of Theorem 4. We start by sketching a general algorithm scheme in the subsequent section and recover the *Ellipsoid method* as a special case of it. We then prove that the Ellipsoid method indeed takes just a polynomial number of "simple" iterations to converge.

2 The Cutting Plane Method

In this section we develop a general method for optimizing linear functions over polytopes, called the cutting plane method. We are interested in solving the problem

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & x \in P, \end{aligned} \tag{1}$$

where $P \subseteq \mathbb{R}^n$ is a polytope and we have access to a separation oracle form P. Throughout this section we assume that P is full-dimensional. Later we comment on how to get around this assumption.

[2]We measure the running time of a separation oracle as a function of the dimension n and the bit complexity of the input vector x.

2.1 First step: reducing optimization to feasibility using binary search

Before we proceed with the description of the algorithm we first introduce the first idea that allows us to reduce the optimization problem (1) to a series of simpler problems of the following form.

Feasibility problem for polytopes

Input: A polytope $P \subseteq \mathbb{R}^n$

Goal: Output one of the following answers

- if $P \neq \emptyset$ output YES and provide a point $x \in P$
- if $P = \emptyset$ output NO

We explain the main idea of this reduction without delving too deep into technical details behind it. Later, in the proof of Theorem 4 we show how one can implement this idea in the setting of 0-1 polytopes.

The idea is quite simple here: suppose we have a rough estimate that the optimal value of (1) belongs to the interval $[l_0, u_0]$. We can then use binary search, along with an algorithm for solving the feasibility problem to find an estimate on the optimal value c_{opt} of (1) up to an arbitrary precision.

Reducing Optimization to Feasibility

1. Set $l := l_0$ and $u := u_0$
2. while $u - l > \varepsilon$
   (a) set $g := \frac{l+u}{2}$
   (b) Check whether $P' := P \cap \{x \in \mathbb{R}^n : \langle c, x \rangle \leq g\}$ is nonempty
   - If $P' \neq \emptyset$, then $u := g$
   - Else, $l := g$
3. Output u

The above algorithm maintains an interval $[l, u]$ that contains the optimal value c_{opt} to (1) and uses an algorithm to check non-emptiness of a polytope in step (2b) to halve it in every step until its length is at most $\varepsilon > 0$. Thus essentially, given an algorithm to check feasibility we are able to perform optimization. Several remarks are in order

1. Note that given a separation oracle for P, the separation oracle for P' is easy to construct.

2. The above allows to learn an additive ε-approximate optimal value in roughly $\log \frac{u_0 - l_0}{\varepsilon}$ calls to a feasibility oracle. However it is not clear how to recover the exact optimum c_{opt} out of it. We show that for certain classes of polytopes, such as 0-1 polytopes, one can compute c_{opt} exactly, given a good enough estimate of c_{opt}.

3. The above scheme computes only (approximate) optimal value of (1) and not the optimal point $x^\star$. One possible way to deal with this issue is to simply use the feasibility oracle to find a point $\hat{x}$ in $P' := P \cap \{x \in \mathbb{R}^n : \langle c, x \rangle \leq u\}$ for the final value of u. The polytope P' is guaranteed to be nonempty and the point $\hat{x}$ has value at most $c_{opt} + \varepsilon$. Again, in the proof of Theorem 4 we show how to round $\hat{x}$ to an exact optimal solution.

4. To make this into a polynomial time reduction one needs to make sure that the polytopes P' constructed in step (2b) do not become difficult for the feasibility oracle. In fact, the running time of the algorithm we provide for this task depends on the volume of P' and it becomes slower as the volume of P' gets very small. Thus in general, one has to carefully bound how complex the polytopes P' become as we progress with the algorithm.

Given this idea, it is enough (up to the technical details highlighted above) to construct an algorithm for checking non-emptiness of polytopes.

2.2 Second step: checking feasibility using cutting planes

Suppose we are given a large convex set E_0 containing the polytope P. To check whether P is empty or not, we construct a descending sequence of convex subsets

$$E_0 \supseteq E_1 \supseteq E_2 \supseteq \cdots$$

such that all of them contain P and their volume[3] is significantly reducing in every step. Eventually, the set E_k "approximates" P so well that a point $x \in E_k$ is likely to be in P as well. Formally, consider the following scheme.

[3]Formally, we use the Lebesgue measure to talk about volumes of sets. We denote by $\mathrm{vol}(K)$ the $n-$dimensional Lebesgue measure of a set $K \subseteq \mathbb{R}^n$.

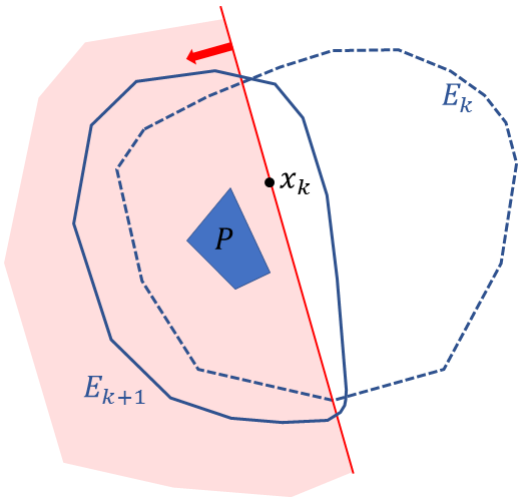

Figure 1: An illustration of one iteration of the cutting plane method. One starts with a set E_k guaranteed to contain the polytope P and queries a point $x_k \in E_k$. If x_k is not in P, then the separation oracle for P provides a separating hyperplane (in red). Hence one can reduce the region to which P belongs by picking a set E_{k+1} containing the intersection of E_k with the "red" hyperplane.

Cutting Plane Method

1. Let E_0 be a convex set containing P
2. For $k := 0, 1, 2, \ldots$ until $\text{vol}(E_k) < \text{vol}(P)$
   (a) Select a point $x_k \in E_k$
   (b) Query a separation oracle for P on x_k
   - if $x_k \in P$ return x_k and terminate the algorithm,
   - else, let $h_k \in \mathbb{Q}^n$ be the separating hyperplane output by the oracle
   - construct a new set E_{k+1} so that

$$E_k \cap \{x : \langle x, h_k \rangle \leq \langle x_k, h_k \rangle\} \subseteq E_{k+1}.$$

3. return EMPTY ($P = \emptyset$).

We refer to figure 1 for an illustration of the above method.

The following theorem characterizes the performance of the above scheme

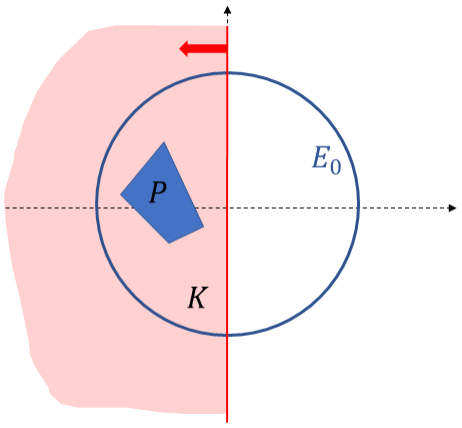

Figure 2: An illustration of the use of Euclidean balls in the cutting plane method. Note that the smallest ball containing the intersection of E with the "red" halfplane is E itself.

Theorem 5 (Number of iterations of the cutting plane method)**.** *Suppose that the following conditions on P and the nested sequence of sets $E_0, E_1, E_2, \ldots$ are satisfied*

1. **(Bounding Ball)** *The initial set E_0 is contained in an Euclidean ball of radius $R > 0$,*
2. **(Inner Ball)** *The polytope P contains an Euclidean ball of radius $r > 0$,*
3. **(Volume Drop)** *For every step $k = 0, 1, \ldots,$ we have*

$$\mathrm{vol}(E_{k+1}) \leq \alpha \mathrm{vol}(E_k),$$

where $\alpha < 1$ is a quantity possibly depending on n.

Then, the Cutting Plane Method outputs a point $\hat{x} \in P$ after $O\left(n \log \alpha^{-1} \log \frac{R}{r}\right)$ iterations.

Proof. The algorithm maintains the invariant that $P \subseteq E_k$ at every step k, hence it never terminates with the verdict EMPTY, as $\mathrm{vol}(P) \leq \mathrm{vol}(E_k)$ for every k.

It remains to upper bound the time until the volume of E_k cannot be reduced anymore. Note that from the Volume Drop condition, we have

$$\mathrm{vol}(E_k) \leq \mathrm{vol}(E_0)\alpha^k,$$

moreover, since P contains some ball $B(x', r)$ and E_0 is contained in some ball $B(x'', r)$, we have

$$\mathrm{vol}(B(x', r)) \leq \mathrm{vol}(E_k) \leq \mathrm{vol}(E_0)\alpha^k \leq \mathrm{vol}(B(x'', R))\alpha^k.$$

Thus

$$\frac{\mathrm{vol}(B(x',r))}{\mathrm{vol}(B(x'',R))} \leq \alpha^k. \tag{2}$$

However, both $B(x',r)$ and $B(x'',R)$ are scalings (and translations) of the unit Euclidean ball $B(0,1)$ and hence

$$\mathrm{vol}(B(x',r)) = r^n \cdot \mathrm{vol}(B(0,1)) \qquad \text{and} \qquad \mathrm{vol}(B(x'',R)) = R^n \cdot \mathrm{vol}(B(0,1)).$$

By plugging the above in (2) we obtain

$$k \leq n \log \alpha^{-1} \log \frac{R}{r}.$$

□

Note that if we could find an instantiation of the scheme (a way of choosing E_k and x_k at every step k) such that the corresponding parameter α is a constant less than 1, then the above yields an algorithm for linear programming that performs a polynomial number of iterations – indeed using the results of Exercise 1. in Problem Set 7 it follows that we can always enclose P in a ball of radius roughly $R = 2^{O(Ln)}$ and find a ball inside of P of radius $r = 2^{-O(Ln)}$. Thus $\log \frac{R}{r} = O(Ln)$.

The method developed in this lecture will not quite achieve a constant drop in volume but rather $\alpha \approx 1 - \frac{1}{n}$, however this still implies a method for linear programming which performs a polynomial number of iterations, and (as we will see later) each iteration is just a simple matrix operation that can be implemented in polynomial time.

2.3 Possible choices for E_k and x_k

Before we give more details, let us discuss two possible, simple strategies of picking the sets E_k (and the points x_k)

2.3.1 Approximation using Euclidean balls

One of the simplest ideas to implement the *cutting plane method* is to use Euclidean balls as E_k's. There is then a very natural choice of x_k one can make at every step, being the center of the current ball E_k.

The problem with this method is that we cannot force the volume drop when Euclidean balls are used. Indeed, consider a simple an example in $\mathbb{R}^2$. Let $E_0 = B(0,1)$ be the starting ball and suppose that the separation oracle provides the separating hyperplane $h = (1,0)^\top$, meaning that

$$K := P \subseteq E_0 \cap \{x \in \mathbb{R}^2 : x_1 \leq 0\}.$$

What is the "smallest" ball containing K as a subset? (See Figure 2 for an illustration.) It turns out that this is again $B(0,1)$ and we cannot find a ball with smaller radius containing

K. This simply follows from the fact that the diameter of K is 2 (the distance between $(0,-1)^\top$ and $(0,1)^\top$ is 2).

For this reason we need to look for a larger family of sets, which will allow us to achieve a suitable volume drop in every step.

2.3.2 Approximation using polytopes

The second idea one can try is to simply use polytopes as E_k's. We start with $E_0 = [-R, R]^n$ – a polytope containing P and whenever a new hyperplane is obtained from the separation oracle, we update

$$E_{k+1} = E_k \cap \{x : \langle x, h_k \rangle \leq \langle x_k, h_k \rangle \}.$$

This is certainly the most "aggressive" strategy one can imagine, as we always cut out as much a possible from E_k to obtain E_{k+1}.

To complete the description of the scheme we also need to give a strategy of picking a point $x_k \in E_k$ at every iteration. For this, it is crucial that the point x_k is "well in the interior" of E_k, as otherwise we might end up cutting only a small piece out of E_k and thus not reduce the volume enough. Therefore, to make it work, at every step we need to find an approximate "center" of the polytope E_k and use it as the query point x_k. As the polytope gets more and more complex, finding a center also becomes more and more time consuming. In fact, we are reducing a problem of finding a point in P to a problem of finding a point in another polytope, and thus this becomes a "cyclic process".

We remark that, this scheme can be made work by maintaining a center of the polytope and using a recentering step whenever a new constraint is added [12], this is based on the idea that the new center will be close to the old one, when the polytope does not change too drastically and hence can be found rather efficiently using for instance the Newton's method.

3 The Ellipsoid method

What we have concluded from the above discussions is that on the one hand using Euclidean balls is too restrictive, since we cannot make eny progress in subsequent steps. On the other hand, maintaining a polytope is problematic, since it is not easy to find a suitable point $x_k \in E_k$ to query, at every step k.

One can then use an idea somewhere in between these two. We maintain an ellipsoid: a set which is more general then just a ball, it can approximate polytopes quite well and its center can be found very easily. A formal definition of ellipsoids along with a piece of notation follows.

Definition 6 (Ellipsoid)**.** *An ellipsoid in $\mathbb{R}^n$ is any set of the form*

$$E(x_0, M) = \left\{ x \in \mathbb{R}^n : (x - x_0)^\top M^{-1} (x - x_0) \leq 1 \right\},$$

such that $x_0 \in \mathbb{R}^n$ and M is an $n \times n$, symmetric positive definite matrix.

It is clear that $E(x_0, M)$ is symmetric around its center x_0. Suppose that $E_k = E(x_k, M_k)$ and the separation oracle provides us with the information that the polytope P is contained in $\{x \in E_k : \langle x, h_k \rangle \leq \langle x_k, h_k \rangle\}$. Then a natural strategy is to take E_{k+1} so as to contain this set and have the smallest volume among all such. More formally, we define

Definition 7 (Minimum Enclosing Ellipsoid)**.** *Given an ellipsoid $E(x_0, M) \subseteq \mathbb{R}^n$ and a half-space $H := \{x \in \mathbb{R}^n : \langle x, h \rangle \leq \langle x_0, h \rangle\}$ through its center, for some $h \in \mathbb{R}^n \setminus \{0\}$ we define the minimum enclosing ellipsoid of $E(x_0, M) \cap H$ to be the ellipsoid $E(x^\star, M^\star)$ being the optimal solution to*

$$\begin{aligned} \min_{x \in \mathbb{R}^n, M' \succ 0} \quad & \mathrm{vol}(E(x, M')) \\ \text{s.t.} \quad & E(x_0, M) \cap H \subseteq E(x, M'). \end{aligned}$$

One can show that such a minimum enclosing ellipsoid always exists and is unique, as the above can be formulated as a convex program.

3.1 The Algorithm

The ellipsoid algorithm is a simple adaptation of the cutting plane method where we just use ellipsoids as the sets E_k. Moreover, for every step k we take the minimum enclosing ellipsoid as E_{k+1}. Let us state it formally now. Again, we assume that $P \subseteq \mathbb{R}^n$ is a full-dimensional polytope and a separation oracle is available for P

Ellipsoid Method

1. Let $E_0 = E(0, M_0)$ be a ball containing containing P
2. For $k := 0, 1, 2, \ldots$ until $\mathrm{vol}(E_k) < \mathrm{vol}(P)$
   (a) Let x_k be the center of E_k
   (b) Query a separation oracle for P on x_k
   - if $x_k \in P$ return x_k and terminate the algorithm,
   - else, let $h_k \in \mathbb{Q}^n$ be the separating hyperplane output by the oracle
   - let $E_{k+1} = E(x_{k+1}, M_{k+1})$ be the minimum volume ellipsoid such that
   $$E_k \cap \{x : \langle x, h_k \rangle \leq \langle x_k, h_k \rangle\} \subseteq E_{k+1}.$$
3. return EMPTY ($P = \emptyset$).

The above algorithm was first developed for linear programs in explicit form by [6] and was further generalized to the case of implicitly defined polytopes (via separation oracles) by [5], [8] and [4].

Note that in this form the algorithm is not yet complete, as we have not provided a way to compute minimum enclosing ellipsoids (as is required in step (2b)). We discuss this question in Section 4.

3.2 Analysis of the algorithm

We are now ready to state a theorem on the computation efficiency of the ellipsoid method. The main part of the proof is provided in Section 4.

Theorem 8 (Efficiency of the ellipsoid method)**.** *Suppose that $P \subseteq \mathbb{R}^n$ is a full-dimensional polytope that is contained in an n-dimensional Euclidean ball of radius $R > 0$ and contains an n-dimensional Euclidean ball of radius $r > 0$. Then, the Ellipsoid Method outputs a point $\hat{x} \in P$ after $O\left(n^2 \log \frac{R}{r}\right)$ iterations.*

Moreover, every iteration can be implemented in $O(n^2 + T_{SO})$ time, where T_{SO} is the time required to answer one query by the separation oracle.

Given what has been already proved in Theorem 5, we are missing the following two components (stated in a form of a lemma) to deduce Theorem 8

Lemma 9 (Informal; see Lemma 10)**.** *Consider the ellipsoid algorithm defined above*

1. *The volume of the ellipsoids $E_0, E_1, E_2, \ldots$ constructed in the algorithm drops at a rate $\alpha \approx \left(1 - \frac{1}{2n}\right)$,*
2. *Every iteration of the ellipsoid algorithm can be implemented efficiently (in polynomial time).*

Given the above lemma (established formally in Lemma 10) we are ready to deduce Theorem 8.

Proof of Theorem 8. By combining Theorem 5 with Lemma 9 we obtain the bound on the number of iterations:

$$O\left(n \log \alpha^{-1} \cdot \log \frac{R}{r}\right) = O\left(n^2 \log \frac{R}{r}\right).$$

Moreover, according to Lemma 9 every iteration can be performed efficiently, hence the second part of Theorem 8 also follows. □

4 Analysis of Volume Drop and Efficiency for Ellipsoids

In this section we show that given an ellipsoid $E(x_0, M)$ and a vector $h \in \mathbb{R}^n$ we can efficiently construct a new ellipsoid $E(x_0', M')$ that has significantly smaller volume and

$$E(x_0, M) \cap \{x \in \mathbb{R}^n : \langle x, h\rangle \leq \langle x_k, h\rangle\} \subseteq E(x_0', M').$$

While we do not show it here, the construction demonstrated in this section yields the minimum volume ellipsoid containing the intersection of $E(x_0, M)$ and the halfspace above. Nevertheless the lemma below is enough to complete the description and analysis of the ellipsoid method.

Lemma 10 (Volume Drop). *Let $E(x_0, M) \subseteq \mathbb{R}^n$ be any ellipsoid in $\mathbb{R}^n$ and let $h \in \mathbb{R}^n$ be a non-zero vector. There exists an ellipsoid $E(x_0', M') \subseteq \mathbb{R}^n$ such that*

$$E(x_0, M) \cap \{x \in \mathbb{R}^n : \langle x, h\rangle \leq \langle x_k, h\rangle\} \subseteq E(x_0', M')$$

and $\mathrm{vol}(E(x_0', M')) \leq e^{-\frac{1}{2(n+1)}} \cdot \mathrm{vol}(E(x_0, M))$. *Moreover x_0' and M' can be given explicitly as follows*

$$x_0' := x_0 - \frac{1}{n+1} Mg, \tag{3}$$

$$M' := \frac{n^2}{n^2-1}\left(M - \frac{2}{n+1}(Mg)^\top Mg\right), \tag{4}$$

where $g := \left(h^\top M h\right)^{-1/2} h$.

The above lemma states in particular that using Ellipsoids we can implement the cutting plane method with volume drop $\alpha = e^{-\frac{1}{2(n+1)}} \approx (1 - \frac{1}{2(n+1)}) < 1$.

The proof of Lemma 10 is divided into two phases. In the first phase we prove Lemma 10 for a very simple and symmetric case. In the second phase, we reduce the general case to the symmetric instance studied in the first phase using an affine transformation of $\mathbb{R}^n$.

4.1 Volumes and ellipsoids under affine transformations

In this section we review basic facts about affine transformation and ellipsoids. We start by proving how the volume of a set changes when an affine transformation is applied to it.

Fact 1 (Change of volume under affine map). *Consider an affine map $\phi(x) = Tx + b$, where $T : \mathbb{R}^n \to \mathbb{R}^n$ is an invertible linear map and $b \in \mathbb{R}^n$ is any vector. Let $K \subseteq \mathbb{R}^n$ be a Lebesgue measurable set. Then*

$$\mathrm{vol}(\phi(K)) = |\det(T)| \cdot \mathrm{vol}(K).$$

Proof. This is a simple consequence of integrating by substitution. We have

$$\mathrm{vol}(\phi(K)) = \int_{\phi(K)} d\lambda_n(x),$$

where λ_n is the n-dimensional Lebesgue measure. We apply the change of variables $x := \phi(y)$ and obtain

$$\int_{\phi(K)} d\lambda_n(x) = \int_K |\det(T)|\, d\lambda_n(y) = |\det(T)| \cdot \mathrm{vol}(K),$$

because T is the Jacobian of the mapping ϕ. □

The next fact talks about the effect of applying an affine transformation on an ellipsoid.

Fact 2 (Affine transformations of ellipsoids)**.** *Consider an affine map $\phi(x) = Tx + b$, where $T : \mathbb{R}^n \to \mathbb{R}^n$ is an invertible linear map and $b \in \mathbb{R}^n$ is any vector. Let $E(x_0, M)$ be any ellipsoid in $\mathbb{R}^n$, then*

$$\phi(E(x_0, M)) = E(Tx_0 + b, TMT^\top).$$

Proof.

$$\begin{aligned}
\phi(E(x_0, M)) &= \{\phi(x) : (x_0 - x)^\top M^{-1}(x_0 - x) \leq 1\} \\
&= \{y : (x_0 - \phi^{-1}(y))^\top M^{-1}(x_0 - \phi^{-1}(y)) \leq 1\} \\
&= \{y : (x_0 - T^{-1}(y - b))^\top M^{-1}(x_0 - T^{-1}(y - b)) \leq 1\} \\
&= \{y : T^{-1}(Tx_0 - (y - b))^\top M^{-1} T^{-1}(Tx_0 - (y - b)) \leq 1\} \\
&= \{y : (Tx_0 - (y - b))^\top (T^{-1})^\top M^{-1} T^{-1}(Tx_0 - (y - b)) \leq 1\} \\
&= E(Tx_0 + b, TMT^\top).
\end{aligned}$$

□

From the two previous facts one can easily derive the following corollary regarding the volume of an ellipsoid.

Corollary 11 (Volume of an ellipsoid)**.** *Let $x_0 \in \mathbb{R}^n$ and $M \in \mathbb{R}^{n \times n}$ be a PD matrix. Then*

$$\mathrm{vol}(E(x_0, M)) = \det(M)^{1/2} \cdot V_n,$$

where V_n denotes the volume of the unit Euclidean ball in $\mathbb{R}^n$.

Proof. We first observe that

$$E(x_0, M) = \phi(E(0, I)), \tag{5}$$

where $\phi(x) = M^{1/2}x + x_0$. This follows from Fact 2. Fact 1 gives now that

$$\mathrm{vol}(E(x_0, M)) = \det(M)^{1/2}\mathrm{vol}(E(0, I)) = \det(M)^{1/2} \cdot V_n.$$

□

4.2 The Symmetric case

In the symmetric case we assume that the ellipsoid is the unit ball $E(0, I)$ and we intersect it with the halfspace $H = \{x \in \mathbb{R}^n : x_1 \geq 0\}$. Then one can obtain an ellipsoid of a relatively small volume that contains $E(0, I) \cap H$ via an explicit formula, as in the lemma below.

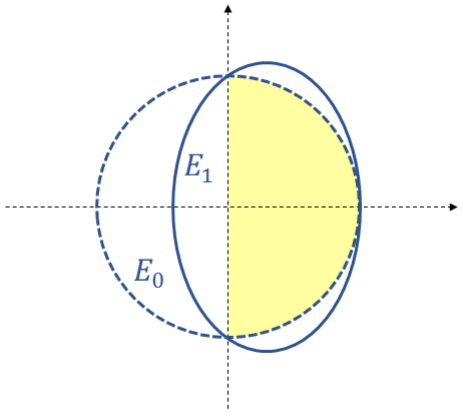

Figure 3: Illustration of the symmetric case of the argument. E_0 is the unit Euclidean ball in $\mathbb{R}^2$ and E_1 is the ellipse of smallest area that contains the "right half" (depicted in yellow) of E_0. The area of E_1 is $\frac{4\sqrt{3}}{9} < 1$.

Lemma 12 (Volume drop for the symmetric case)**.** *Consider the Euclidean ball $E(0, I)$. Then the ellipsoid $E' \subseteq \mathbb{R}^n$ given by*

$$E' := \left\{ x \in \mathbb{R}^n : \left(\frac{n+1}{n}\right)^2 \left(x_1 - \frac{1}{n+1}\right)^2 + \frac{n^2-1}{n^2} \sum_{j=2}^{n} x_j^2 \leq 1 \right\}$$

has volume $\mathrm{vol}(E') \leq e^{-\frac{1}{2(n+1)}} \cdot \mathrm{vol}(E(0, I))$ *and satisfies*

$$\{x \in E(0, I) : x_1 \geq 0\} \subseteq E'.$$

Proof. We start by showing that $\{x \in E(0, I) : x_1 \geq 0\} \subseteq E'$. To this end, take any point $x \in E(0, I)$ with $x_1 \geq 0$, the goal is to show that $x \in E'$. We first reduce the problem to a univariate question using the following observation

$$\sum_{j=2}^{n} x_j^2 \leq 1 - x_1^2.$$

It follows from the fact that x belongs to the unit ball. Thus, to conclude that $x \in E'$ we just need to show that

$$\left(\frac{n+1}{n}\right)^2 \left(x_1 - \frac{1}{n+1}\right)^2 + \frac{n^2-1}{n^2} \sum_{j=2}^{n} (1 - x_1^2) \leq 1. \tag{6}$$

Note that the left hand side of (6) is a convex quadratic function in one variable. We would like to show that it is bounded by 1 for every $x_1 \in [0, 1]$. To this end, it suffices to verify

it for both end-points $x_1 = 0, 1$. By directly plugging in $x_1 = 0$ and $x_1 = 1$ we obtain that the left-hand side of (6) equals 1 and thus the inequality is satisfied.

We now proceed to bounding the volume of E'. To this end, note that $E' = E(z, S)$ with

$$z := \left(\frac{1}{n+1}, 0, 0, \ldots, 0\right)^\top,$$
$$S := \mathsf{Diag}\left(\left(\frac{n}{n+1}\right)^2, \frac{n^2}{n^2-1}, \ldots, \frac{n^2}{n^2-1}\right).$$

Where $\mathsf{Diag}\,(x)$ is a diagonal matrix with diagonal entries as in the vector x. To compute the volume of E' we can simply apply Corollary 11

$$\mathrm{vol}(E') = \det(S)^{1/2} \cdot \mathrm{vol}(E(0, I)).$$

We have

$$\det(S) = \left(\frac{n}{n+1}\right)^2 \cdot \left(\frac{n^2}{n^2-1}\right)^{n-1} = \left(1 - \frac{1}{n+1}\right)^2 \left(1 + \frac{1}{n^2-1}\right)^{n-1}.$$

Using the inequality $1 + x \leq e^x$, valid for every $x \in \mathbb{R}$ we arrive at the upper bound

$$\det(S) \leq \left(e^{-\frac{1}{n+1}}\right)^2 \cdot \left(e^{\frac{1}{n^2-1}}\right)^{n-1} = e^{-\frac{1}{n+1}}.$$

Finally, we obtain

$$\mathrm{vol}(E') = \det(S)^{1/2} \cdot \mathrm{vol}(E(0, I)) \leq e^{-\frac{1}{2(n+1)}} \cdot \mathrm{vol}(E(0, I)).$$

□

4.3 General case

We are now ready to prove Lemma 10 in full generality, by reducing it to the symmetric case, see Figure 4.

Proof of Lemma 10. We note first that the Ellipsoid $E' = E(z, S)$ defined for the symmetric case (Lemma 12) is given by

$$z = \frac{1}{n+1} e_1,$$
$$S = \frac{n^2}{n^2-1}\left(I - \frac{2}{n+1} e_1 e_1^\top\right).$$

thus indeed, this gives a proof of Lemma 10 in the case when $E(x_0, M)$ is the unit ball and $h = -e_1$.

The idea is now to find an affine transformation ϕ so that

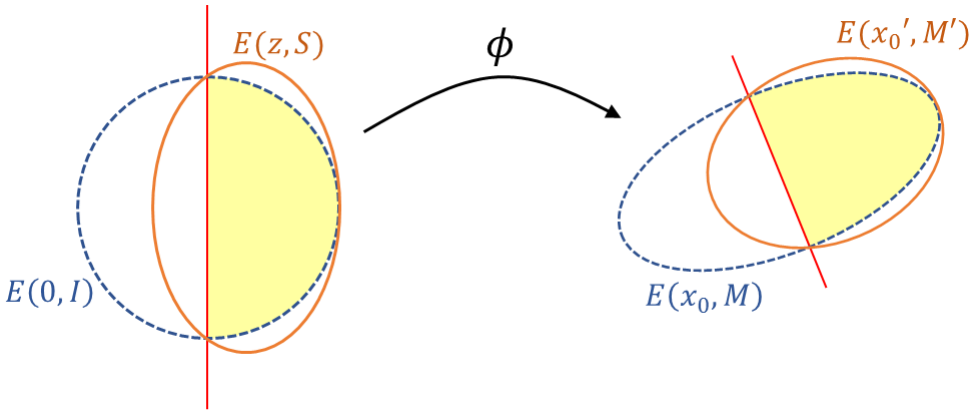

Figure 4: The general case of the argument follows by using an affine transformation ϕ to map the symmetric case to the general one. The unit ball $E(0,I)$ is mapped by ϕ to the ellipsoid $E(x_0,M)$ so that the yellow region $\{x \in E(0,I) : x_1 \geq 0\}$ on the left hand side is also mapped to the yellow region $\{x \in E(x_0,M) : \langle x,h\rangle \leq \langle x_0,h\rangle\}$ on the right hand side.

1. $E(x_0,M) = \phi(E(0,I))$,
2. $\{x : \langle x,h\rangle \leq \langle x_0,h\rangle\} = \phi\left(\{x : x_1 \geq 0\}\right)$.

We refer to Figure 4 for an illustration of this idea. We claim that when these conditions hold, the ellipsoid $E(x_0',M') := \phi(E')$ satisfies the conclusion of Lemma 10. Indeed, using Fact 1 and Lemma 12 we have

$$\frac{\mathrm{vol}(E(x_0',M'))}{\mathrm{vol}(E(x_0,M))} = \frac{\mathrm{vol}(\phi(E(x_0',M')))}{\mathrm{vol}(\phi(E(x_0,M)))} = \frac{\mathrm{vol}(E')}{\mathrm{vol}(E(0,I))} \leq e^{-\frac{1}{2(n+1)}}.$$

Moreover, by applying ϕ to the inclusion

$$\{x \in E : x_1 \geq 0\} \subseteq E'$$

we obtain

$$\{x \in E(x_0,M) : \langle x,h\rangle \leq \langle x_0,h\rangle\} \subseteq E(x_0',M').$$

It remains now to derive a formula for ϕ so that we can obtain an explicit expression for $E(x_0',M')$. We claim that the following affine transformation ϕ satisfies the above stated properties

$$\phi(x) = x_0 + M^{1/2}Ux,$$

where U is any orthonormal matrix (i.e., U satisfies $UU^\top = U^\top U = I$) such that $Ue_1 = v$ where $v = -\frac{M^{1/2}h}{\|M^{1/2}h\|}$. We have

$$\phi(E(0,I)) = E(x_0, M^{1/2}U^\top UM^{1/2}) = E(x_0,M).$$

which proves the first condition, further,

$$\begin{aligned}
\phi(\{x : x_1 \geq 0\}) &= \{\phi(x) : x_1 \geq 0\} \\
&= \{\phi(x) : \langle -e_1, x \rangle \leq 0\} \\
&= \{y : \left\langle -e_1, \phi^{-1}(y) \right\rangle \leq 0\} \\
&= \{y : \left\langle -e_1, U^\top M^{-1/2}(y - x_0) \right\rangle \leq 0\} \\
&= \{y : \left\langle -M^{-1/2}Ue_1, y - x_0 \right\rangle \leq 0\} \\
&= \{y : \langle h, y - x_0 \rangle \leq 0\}
\end{aligned}$$

and hence the second condition follows.

It remains to derive a formula for $E(x_0', M')$. By Fact 2 we obtain

$$E(x_0', M') = \phi(E(z, S)) = E(x_0 + M^{1/2}Uz, M^{1/2}USU^\top M^{1/2}).$$

Thus

$$x_0' = x_0 + \frac{1}{n+1} M^{1/2}Ue_1 = x_0 - \frac{1}{n+1} \frac{Mh}{\|M^{1/2}h\|}$$

and

$$\begin{aligned}
M' &= M^{1/2}U^\top \frac{n^2}{n^2-1} \left(I - \frac{2}{n+1} e_1 e_1^\top \right) UM^{1/2} \\
&= \frac{n^2}{n^2-1} \left(I - \frac{2}{n+1} \left(M^{1/2}Ue_1 \right) \left(M^{1/2}Ue_1 \right)^\top \right) \\
&= \frac{n^2}{n^2-1} \left(I - \frac{2}{n+1} \frac{(Mh)\,(Mh)^\top}{h^\top Mh} \right)
\end{aligned}$$

By plugging in $g := \frac{h}{\|M^{1/2}h\|}$ the lemma follows. □

4.4 Discussion on bit precision issues

In all the above discussion we have assumed that the ellipsoids $E_0, E_1, \ldots$ can be computed exactly and no error is introduced at any step. In reality, this is not the case, as computing E_{k+1} from E_k involves taking a square root, an operation that cannot be performed exactly. Moreover, one needs to be careful about the bit complexities of the intermediate computations, as there is no simple reason for why would they stay polynomially bounded over the course of the algorithm.

To counter this issue one can first establish an appropriate bound on the precision required for of all the numbers which show up in the computation and then perform a

suitable rounding after every step. This has the effect of choosing a slightly larger ellipsoid in every step, yet still guarantees a significant drop in the volume, sufficient to converge in a polynomial number of iterations. For details, we refer to [10].

5 Linear Optimization over 0-1 Polytopes Using Separation Oracles

In this section we give a proof of Theorem 4 in the case when the polytope P is full-dimensional (at the end of this section we discuss how to get rid of this assumption). For this we use the binary search technique to reduce the optimization problem to feasibility and use the ellipsoid algorithm to solve the feasibility question in every step. A more precise sketch of the algorithm follows

Optimization for $0-1$ polytopes.

Input: A separation oracle to a full-dimensional $0-1$ polytope $P \subseteq [0,1]^n$ and a vector $c \in \mathbb{Z}^n$
Output: An optimal solution $x^\star$ to $\min_{x \in P} c^\top x$.

1. Perturb the cost function c to guarantee uniqueness of the optimal solution.
2. Let $l = -\|c\|_1$ and $u = \|c\|_1 + 1$
3. while $u - l > 1$
   (a) Let $g := \lfloor \frac{l+u}{2} \rfloor$
   (b) Apply ellipsoid alg. to check if $P' := P \cap \{x \in \mathbb{R}^n : \langle c, x \rangle \leq g + \frac{1}{4}\}$ is nonempty
   - If $P' \neq \emptyset$, then $u := g$
   - Else, $l := g$
4. Apply the ellipsoid algorithm to find a point
$$\hat{x} \in P \cap \left\{ x \in \mathbb{R}^n : \langle c, x \rangle \leq l + \frac{1}{4} \right\}$$
5. Round every coordinate of $\hat{x}$ to the nearest integer and output it.

Note that in contrast to the scheme presented in Section 2.1, the binary search runs over integers. We can do that, since the cost function is integral and the optimum is guaranteed to be a vertex and thus the optimum value is also integral. Note that if $c_{opt} \in \mathbb{Z}$ is the optimal value then the polytope $P' = P \cap \{x : \langle x, c \rangle \leq c_{opt}\}$ is lower-dimensional and hence has volume zero. For this reason, in the above we introduce a small slack and

always ask for objective at most $g + \frac{1}{4}$ instead of at most g, for $g \in Z$. This guarantees that P' is always full-dimensional and we can provide a lower bound on its volume (necessary for the ellipsoid algorithm to run in polynomial time).

5.1 Guaranteeing uniqueness of the optimal solution

In step 1 of the algorithm we want to make sure that there is only one optimal solution. This is done to be able to round a close-to-optimal solution to a vertex easily. To guarantee that one can make a slight perturbation to the weights, which guarantees that if there are many optimal solutions, then our algorithm will find the least lexicographically optimal vertex.

To this end one can simply define a new cost function $c' \in \mathbb{Z}^n$ such that

$$c'_i := 2^n c_i + 2^{i-1}.$$

The total contribution of the perturbation for any vertex solution is less than 2^n and hence it does not create any new optimal vertex. Moreover, it is now easy to see that every vertex has a different cost, and thus the optimal cost is unique.

Finally note that by doing this perturbation, the bit complexity of c grows only by roughly n.

5.2 Lower-bounding the volume of polytopes

Here we prove formally that the volume of the polytopes P' queried in the algorithm is never too small, for the ellipsoid method to work in polynomial time.

Lemma 13. *Suppose that $P \subseteq [0,1]^n$ is a full-dimensional 0-1 polytope and $c \in \mathbb{Z}^n$ is any cost vector and $C \in \mathbb{Z}$ is any integer. Consider the following polytope*

$$P' = \left\{ x \in P : \langle c, x \rangle \leq C + \frac{1}{4} \right\},$$

then either P' is empty or P' contains a Euclidean ball of radius at least $2^{-\mathrm{poly}(n,L)}$, where L is the bit complexity of c.

Proof. Assume without loss of generality that $0 \in P$ and that 0 is the unique optimal solution. It is then enough to consider $C = 0$, as for $C < 0$ the polytope P' is empty and for $C > 0$ it is larger than for $C = 0$.

We start by noting that P contains a ball B of radius $2^{-\mathrm{poly}(n)}$. To see this, we first pick $n+1$ affinely independent vertices of P and show that the simplex spanned by them contains a ball of such a radius (this follows from the results of Exercise 1 in Problem Set 7).

Next, we show that by scaling this ball down, roughly 2^L times, with respect to the origin, it will be contained in $P' := \left\{ x \in P : \langle c, x \rangle \leq \frac{1}{4} \right\}$. To see this, note that for every point $x \in P$ we have

$$\langle x, c \rangle \leq \max_i |c_i| \leq 2^L.$$

Thus we have, for every $x \in P$:

$$\left\langle \frac{x}{2^{L+3}}, c \right\rangle \leq 2^{-3} = \frac{1}{8},$$

in particular $2^{-L-3}B \subseteq P'$, but

$$\mathrm{vol}\left(2^{-L-3}B\right) = 2^{-n(L+3)}\mathrm{vol}(B) = 2^{-n(L+3)-\mathrm{poly}(n)} = 2^{-\mathrm{poly}(n,L)}.$$

□

5.3 Rounding

Lemma 14. *Suppose that $P \subseteq [0,1]^n$ is a full-dimensional 0-1 polytope and $c \in \mathbb{Z}^n$ is any cost vector such that there is a unique point $x^\star$ in P which minimizes $x \mapsto \langle c, x \rangle$ over $x \in P$. Then, if $x \in P$ satisfies*

$$c^\top x \leq c^\top x^\star + \varepsilon$$

for $\varepsilon < \frac{1}{2}$ then by rounding every coordinate of x to the nearest integer we obtain $x^\star$.

Proof. Let us write x as $x := \alpha x^\star + (1-\alpha)z$ for some $\alpha \in [0,1]$ and $z \in P$ being a convex combination of non-optimal vertices. We have

$$\begin{aligned} c^\top x^\star + \varepsilon &\geq c^\top x \\ &= c^\top(\alpha x^\star + (1-\alpha)z) \\ &= \alpha c^\top x^\star + (1-\alpha)c^\top z \\ &\geq \alpha c^\top x^\star + (1-\alpha)(c^\top x^\star + 1) \\ &\geq c^\top x^\star + (1-\alpha) \end{aligned}$$

Thus $1 - \alpha \leq \varepsilon$. For this reason, for every coordinate $i = 1, 2, \ldots, n$ we have

$$|x_i - x_i^\star| = (1-\alpha)|x_i^\star - z_i| \leq 1 - \alpha \leq \varepsilon < \frac{1}{2}.$$

It follows that rounding x yields $x^\star$. □

5.4 The proof

We are now ready to summarize the discussion from above and prove Theorem 4.

Proof of Theorem 4. We show that the algorithm presented in the beginning of this section is correct and runs in polynomial time.

The correctness is a straightforward consequence of the definition of the algorithm. The only step which requires justification is why does the rounding produce the optimal solution $x^\star$. This follows from Lemma 14.

The bound the running time of the algorithm observe that it performs $O(\log \|c\|_1) = O(L)$ iterations and in every iteration it applies the ellipsoid algorithm to check the non-emptiness of P'. Using Theorem 8 and the lower-bound on the size of a ball that can be fit in P' provided in Lemma 13 we conclude that the running time of one such application is $\text{poly}(n, L)$. □

5.5 The full-dimensionality assumption

For the proof of Theorem 4 in this section we have crucially used the fact that the polytope P is full dimensional. Indeed, we have used it to give lower bound on the volume of P and its restrictions. When P is lower-dimensional, no such bounds hold and the analysis needs to be adjusted.

In principle there are two ways to deal with this issue. The first idea assumes that one is given an affine subspace $F = \{x \in \mathbb{R}^n : Ax = b\}$ such that the polytope P is full-dimensional relative to F. In such a case one can simply apply the ellipsoid method restricted to F and obtain the same running time bound as in the full-dimensional case.

In the case when the description of a subspace on which P is full-dimensional is not available, the situation becomes more complicated, but still possible to deal with. The idea is to apply ellipsoid method to the low-dimensional polytope P in order to first find the subspace F. To this end, after enough iterations of the ellipsoid method, the ellipsoid becomes "flat" and one can read off the directions along which this happens (the ones corresponding to tiny eigenvalues). Then one can use the idea of simultaneous Diophantine approximations to round these vectors to a low bit complexity representation of the subspace F. The details of how is it exactly done are somewhat non-trivial but also rather tedious – we refer the reader to [4].

References

[1] Jacob D. Abernethy and Elad Hazan. Faster convex optimization: Simulated annealing with an efficient universal barrier. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2520–2528, 2016.

[2] Sébastien Bubeck and Ronen Eldan. The entropic barrier: a simple and optimal universal self-concordant barrier. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, page 279, 2015.

[3] Jack Edmonds. Maximum matching and a polyhedron with 0, 1 vertices. *J. of Res. the Nat. Bureau of Standards*, 69:125–130.

[4] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

[5] Richard M. Karp and Christos H. Papadimitriou. On linear characterizations of combinatorial optimization problems. *SIAM J. Comput.*, 11(4):620–632, 1982.

[6] Leonid Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53 – 72, 1980.

[7] László Lovász and Santosh Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 57–68, 2006.

[8] Manfred W Padberg and M Rammohan Rao. *The Russian method for linear inequalities III: Bounded integer programming*. PhD thesis, INRIA, 1981.

[9] Thomas Rothvoss. The matching polytope has exponential extension complexity. *J. ACM*, 64(6):41:1–41:19, 2017.

[10] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[11] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2002.

[12] P. M. Vaidya. A new algorithm for minimizing convex functions over convex sets. In *30th Annual Symposium on Foundations of Computer Science*, pages 338–343, Oct 1989.