ISyE/Math/CS/Stat 525
Linear Optimization

8. Complexity of linear programming and the ellipsoid method

Prof. Alberto Del Pia
University of Wisconsin-Madison

Based on the book *Introduction to Linear Optimization* by D. Bertsimas and J.N. Tsitsiklis

Outline

Sec. 8.1 Efficient algorithms and computational complexity.

Sec. 8.2 The key geometric result behind the ellipsoid method.

Sec. 8.3 The ellipsoid method for the feasibility problem.

Sec. 8.4 The ellipsoid method for optimization.

Sec. 8.5 Problems with exponentially many constraints.

Outline

- The ellipsoid method did not lead to a practical algorithm for solving LP problems.
- Rather, it demonstrated that LP is efficiently solvable from a theoretical point of view.
- This is important, because once a problem is shown to be efficiently solvable, usually more efficient and practical algorithms follow.
- Indeed a new class of algorithms, known as interior point methods, that are both practical and theoretically efficient have been proposed.

8.1 Efficient algorithms and computational complexity

Efficient algorithms and computational complexity

- In Section 1.6, we have discussed the notion of an efficient algorithm.
- In this section we refine that discussion and give a formal definition of polynomial time algorithms under the bit model.

Efficient algorithms and computational complexity

- First, we draw a distinction between a problem and an instance of a problem.
- For example, "LP" is a problem.
- Whereas

$$\begin{aligned} \text{minimize} \quad & 2x + 3y \\ \text{subject to} \quad & x + y \leq 1 \\ & x, y \geq 0, \end{aligned}$$

  is an instance of the LP problem.
- We define these terms more generally, as follows.

Efficient algorithms and computational complexity

Definition 8.1
An instance of an optimization problem consists of a feasible set F and a cost function $c : F \to \mathbb{R}$ [the objective is to minimize $c(x)$ over all $x \in F$].
An optimization problem is defined as a collection of instances.

- Instances of a problem need to be described according to a common format.
- For example, instances of LP in standard form can be described by listing the entries of A, b, and c.
- Note that this is a chicken-and-egg definition.

Size of an instance

- Some instances are “larger” than others, and it is convenient to define the notion of the “size” of an instance.
- The definition given below is geared towards computers in which information is represented by binary numbers.

Definition 8.2
The size of an instance is defined as the number of bits used to describe the instance, according to a prespecified format.

- Given that arbitrary real numbers cannot be represented in binary, this definition is geared towards instances involving integer (or rational) numbers.

Size of an instance

- Any integer r with $0 \leq r \leq U$ can be be written in binary as

$$r = a_k 2^k + a_{k-1} 2^{k-1} + \cdots + a_1 2^1 + a_0,$$

  where $k \leq \lfloor \log_2 U \rfloor$ and the scalars a_i are 0 or 1.
- Thus we can represent r using at most

$$\lfloor \log_2 U \rfloor + 1 \qquad \text{bits.}$$

- With an extra bit for the sign, we can also represent negative numbers.
- In other words, we can represent any integer r with $|r| \leq U$ using at most

$$\lfloor \log_2 U \rfloor + 2 = O(\log_2 U) \qquad \text{bits.}$$

Size of an instance

Example: Matrix inversion.

- An instance consists of an $n \times n$ matrix A.
- Assume that the absolute value of the largest element of A is equal to U.
- Since there are n^2 entries in A, the size of such an instance is at most
$$n^2 \cdot O(\log_2 U) = O(n^2 \log_2 U).$$

Size of an instance

Example: LP problem in standard form.

- An instance consists of:
  - an $m \times n$ matrix A,
  - an m-vector b, and
  - an n-vector c.
- Assume that the absolute value of the largest element of A , b, c is equal to U.
- Since there are $(mn + m + n)$ entries in A, b, and c, the size of such an instance is at most

$$(mn + m + n) \cdot O(\log_2 U) = O(mn \log_2 U).$$

Efficient algorithms and computational complexity

- It is to be expected that when an algorithm is applied to instances of larger size, it may take longer to terminate.
- For this reason, the running time of an algorithm is usually expressed as a function of the size of the instance to which it is applied.
- Recall that, in Section 1.6, we defined the running time of an algorithm as the total number of steps involved in carrying out the instructions of the algorithm until termination.
- In particular, we assume that each instruction (including arithmetic operations) takes unit time.
- This is the simplest model, which we call the arithmetic model.

The bit model

- We now discuss the bit model as opposed to the arithmetic model.
- In the bit model, each instruction is decomposed into a set of elementary instructions that operate on single-bit numbers.
- Each such elementary instruction is assumed to take unit time.
- The bit model is more natural than the arithmetic model, and can capture the fact that the time to add two numbers increases with the size of these numbers.
- On the other hand, it complicates the calculation of running time.

The bit model

- Let $T(s)$ be the worst-case running time of some algorithm over all instances of size s, under the bit model.

Definition 8.3
An algorithm runs in polynomial time (under the bit model) if there exists an integer k such that $T(s) = O(s^k)$.

The bit model

- Given that it is easier to count time using the arithmetic model, the following fact is often useful.

> Fact:
> Suppose that an algorithm takes polynomial time under the arithmetic model. Furthermore, suppose that on instances of size s, any integer produced in the course of the execution of the algorithm has size bounded by a polynomial in s. Then, the algorithm runs in polynomial time under the bit model as well.

- The essential reason that makes this fact true is that arithmetic operations can be carried out by subroutines that take polynomial time under the bit model.

The bit model

Example: Gaussian elimination.

- As mentioned in Section 1.6, an $n \times n$ matrix can be inverted using $O(n^3)$ arithmetic operations, by means of the Gaussian elimination algorithm.
- Moreover, it can be shown that Gaussian elimination only produces numbers whose size is bounded by a polynomial in s.
- Hence matrix inversion can be performed in polynomial time also under the bit model.

8.2 The key geometric result behind the ellipsoid method

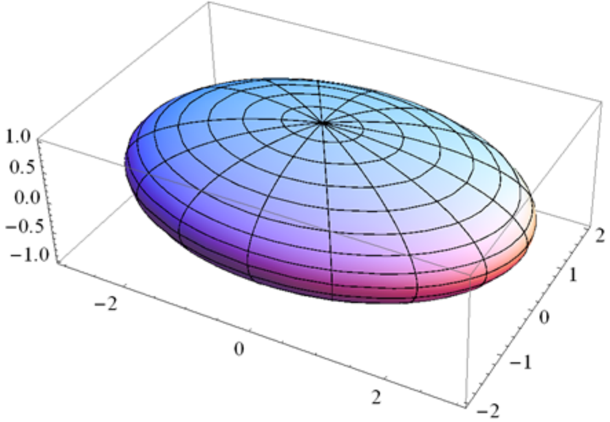
The key geometric result behind the ellipsoid method

- In this section, we prepare the ground for the ellipsoid method, by developing its elements.
- The ellipsoid method can be used to decide whether a polyhedron
$$P = \{x \in \mathbb{R}^n \mid Ax \geq b\}$$
is empty or not.
- Later in this chapter, we provide an extension that solves the optimization problem

$$\begin{aligned} \text{minimize} \quad & c'x \\ \text{subject to} \quad & Ax \geq b. \end{aligned}$$

The key geometric result behind the ellipsoid method

- We start by defining ellipsoids, which are higher dimensional generalizations of the two-dimensional ellipses.



The key geometric result behind the ellipsoid method

Definition 8.4
An $n \times n$ symmetric matrix D is called positive definite if $x'Dx > 0$ for all nonzero vectors $x \in \mathbb{R}^n$.

- In particular, a positive definite matrix D is nonsingular.
- Moreover, D^{-1} is also positive definite.

The key geometric result behind the ellipsoid method

Definition 8.5

A set E of vectors in $\mathbb{R}^n$ of the form

$$E = E(z, D) = \{x \in \mathbb{R}^n \mid (x - z)' D^{-1} (x - z) \leq 1\},$$

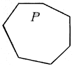
where D is an $n \times n$ positive definite symmetric matrix, is called an ellipsoid with center $z \in \mathbb{R}^n$.

- For any $r > 0$, the ellipsoid

$$\begin{aligned} E(z, r^2 I) &= \{x \in \mathbb{R}^n \mid (x - z)'(x - z) \leq r^2\} \\ &= \{x \in \mathbb{R}^n \mid \|x - z\| \leq r\} \end{aligned}$$

is called a ball centered at z, of radius r.

The key geometric result behind the ellipsoid method

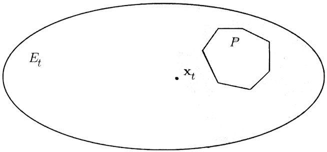


- We first explain intuitively how the ellipsoid method can be used to decide whether a given polyhedron

$$P = \{x \in \mathbb{R}^n \mid Ax \geq b\}$$

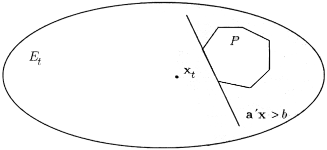
is nonempty.

The key geometric result behind the ellipsoid method



- The algorithm generates a sequence of ellipsoids E_t with centers x_t, such that P is contained in E_t.
- If $x_t \in P$, then P is nonempty and the algorithm terminates.

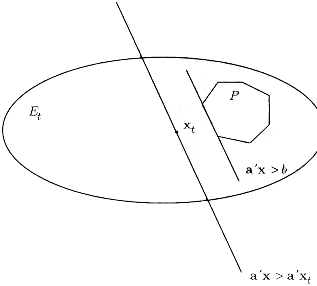
The key geometric result behind the ellipsoid method



- If $x_t \notin P$, then there exists a constraint $a_i' x \geq b_i$ of P that is violated by x_t, i.e.,

$$a_i' x_t < b_i.$$

The key geometric result behind the ellipsoid method

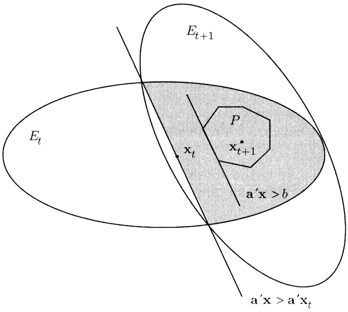


- Any element x of P then satisfies $a_i'x \geq b_i > a_i'x_t$.
- Thus, P is contained in

$$E_t \cap \{x \in \mathbb{R}^n \mid a_i'x \geq a_i'x_t\}.$$

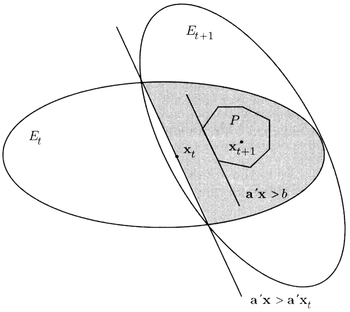
- Since the halfspace passes through the center of the ellipsoid, we call this intersection a half-ellipsoid.

The key geometric result behind the ellipsoid method



- A key geometric property of ellipsoids is that we can find a new ellipsoid E_{t+1} that covers the half-ellipsoid and whose volume is only a fraction of the volume of the previous ellipsoid E_t.

The key geometric result behind the ellipsoid method



- Repeating this process, we either find a point in P, or we conclude that the volume of P is very small and, therefore, P is empty.
- The last step is based on the fact that the volume of a nonempty "full-dimensional" polyhedron cannot be smaller than a certain threshold.

The key geometric result behind the ellipsoid method

- The next theorem gives the analytical formula of the ellipsoid E_{t+1} and the explicit volume reduction.

Theorem 8.1
Let $E = E(z, D)$ be an ellipsoid in $\mathbb{R}^n$, and let a be a nonzero n-vector. Consider the halfspace $H = \{x \in \mathbb{R}^n \mid a'x \geq a'z\}$ and let

$$\bar{z} = z + \frac{1}{n+1}\frac{Da}{\sqrt{a'Da}},$$

$$\bar{D} = \frac{n^2}{n^2-1}\left(D - \frac{2}{n+1}\frac{Daa'D}{a'Da}\right).$$

The matrix $\bar{D}$ is symmetric and positive definite and thus $E' = E(\bar{z}, \bar{D})$ is an ellipsoid. Moreover,

(a) $E \cap H \subset E'$,

(b) $\mathrm{Vol}(E') < e^{-1/(2(n+1))}\,\mathrm{Vol}(E)$.

We will not prove this.

8.3 The ellipsoid method for the feasibility problem

The ellipsoid method for the feasibility problem

- In this section, we describe formally the ellipsoid method for the feasibility problem: Given a polyhedron

$$P = \{x \in \mathbb{R}^n \mid Ax \geq b\},$$

  is it empty or not?
- We will first present the algorithm for full-dimensional polyhedra defined as follows.

Definition 8.7
A polyhedron P is full-dimensional if it has positive volume.

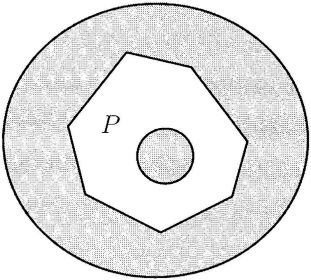
- In order to simplify the presentation and highlight the fundamental geometric ideas we make the following two assumptions, which we relax later.

The ellipsoid method for the feasibility problem

Assumptions

1. The polyhedron P is bounded.
2. The polyhedron P is either empty or full-dimensional.

- Boundedness implies that there exists a ball $E_0 = E(x_0, r^2 I)$, with volume V, that contains P.
- Assumption 2 requires that either P is empty, or P has positive volume, i.e., $\mathrm{Vol}(P) > v$ for some $v > 0$.
- We will assume initially that the ellipsoid E_0, as well as the numbers v, V, are a priori known.



The ellipsoid method for the feasibility problem

The ellipsoid method

Input:

(a) A matrix A and a vector b that define the polyhedron $P = \{x \in \mathbb{R}^n \mid a_i' x \geq b_i, \ i = 1, \ldots, m\}$.

(b) A number v, such that either P is empty or $\mathrm{Vol}(P) > v$.

(c) A ball $E_0 = E(x_0, r^2 I)$ with volume at most V, such that $P \subset E_0$.

Output: A feasible point $x^* \in P$ if P is nonempty, or a statement that P is empty.

The ellipsoid method for the feasibility problem

The ellipsoid method
Algorithm:

1. (Initialization) Let $t^* = \lceil 2(n+1)\log(V/v) \rceil$;
   $E_0 = E(x_0, r^2 I)$; $t = 0$.
2. (Main iteration)
   (a) If $t = t^*$ stop; P is empty.
   (b) If $x_t \in P$ stop; P is nonempty.
   (c) If $x_t \notin P$ find a violated constraint, that is, find an i such that $a_i' x_t < b_i$.
   (d) Let $H_t = \{x \in \mathbb{R}^n \mid a_i' x \geq a_i' x_t\}$. Find an ellipsoid E_{t+1} containing $E_t \cap H_t$ by applying Theorem 8.1.
   (e) $t := t + 1$.

The ellipsoid method for the feasibility problem

Theorem 8.2
Let P be a bounded polyhedron that is either empty or full-dimensional and for which the prior information x_0, r, v, V is available. Then, the ellipsoid method decides correctly whether P is nonempty or not.

Proof idea (1/2):

- If $x_t \in P$ for $t < t^*$, then the algorithm correctly decides that P is nonempty.
- Now assume $x_0, \ldots, x_{t^*-1} \notin P$, thus $P \subset E_k$ for $k = 0, 1, \ldots, t^*$ by Theorem 8.1(a).
- From Theorem 8.1(b), we obtain

$$\frac{\mathrm{Vol}(E_{t^*})}{\cancel{\mathrm{Vol}(E_{t^*-1})}} \cdot \frac{\cancel{\mathrm{Vol}(E_{t^*-1})}}{\cancel{\mathrm{Vol}(E_{t^*-2})}} \cdots \frac{\cancel{\mathrm{Vol}(E_1)}}{\mathrm{Vol}(E_0)} < e^{-t^*/(2(n+1))}.$$

The ellipsoid method for the feasibility problem

Theorem 8.2
Let P be a bounded polyhedron that is either empty or full-dimensional and for which the prior information x_0, r, v, V is available. Then, the ellipsoid method decides correctly whether P is nonempty or not.

Proof idea (2/2):

- $$\mathrm{Vol}(E_{t^*}) < \mathrm{Vol}(E_0)e^{-t^*/(2(n+1))} \le Ve^{-\left\lceil 2(n+1)\log\frac{V}{v}\right\rceil/(2(n+1))}$$
  $$\le Ve^{-\log\frac{V}{v}} = Ve^{\log\frac{v}{V}} = \cancel{V}\frac{v}{\cancel{V}} = v.$$
- Hence $\mathrm{Vol}(P) \le \mathrm{Vol}(E_{t^*}) \le v$.
- By assumption, this implies that P is empty. $\square$

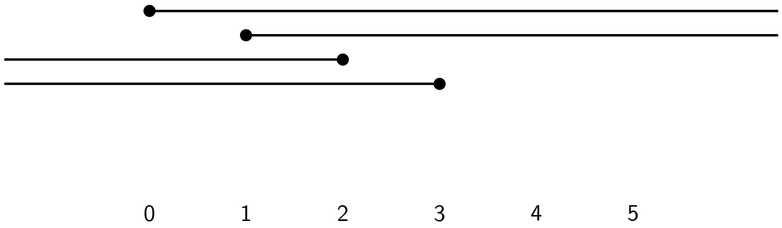
Example 8.1 (The ellipsoid method and binary search)

- We show that in dimension $n = 1$, the ellipsoid method closely resembles binary search, a technique to decide if intervals in the real line have a nonempty intersection.

Example 8.1 (The ellipsoid method and binary search)

- We show that in dimension $n = 1$, the ellipsoid method closely resembles binary search, a technique to decide if intervals in the real line have a nonempty intersection.
- Consider the polyhedron

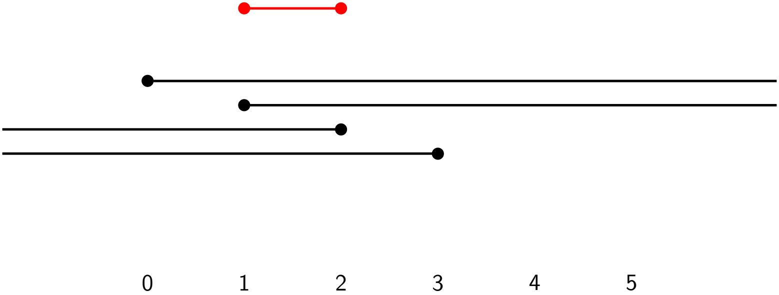
$$P = \{x \in \mathbb{R}^1 \mid x \geq 0,\ x \geq 1,\ x \leq 2,\ x \leq 3\}.$$



Example 8.1 (The ellipsoid method and binary search)

- We show that in dimension $n = 1$, the ellipsoid method closely resembles binary search, a technique to decide if intervals in the real line have a nonempty intersection.
- Consider the polyhedron

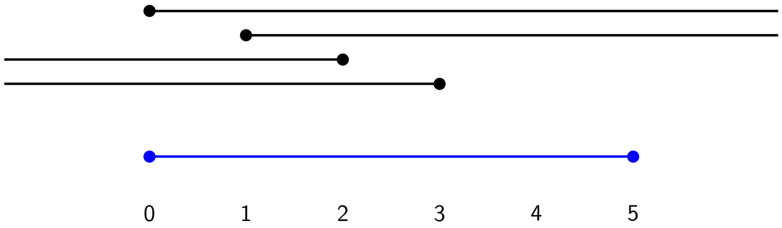
$$P = \{x \in \mathbb{R}^1 \mid x \geq 0,\ x \geq 1,\ x \leq 2,\ x \leq 3\}.$$



Example 8.1 (The ellipsoid method and binary search)

- We show that in dimension $n = 1$, the ellipsoid method closely resembles binary search, a technique to decide if intervals in the real line have a nonempty intersection.
- Consider the polyhedron

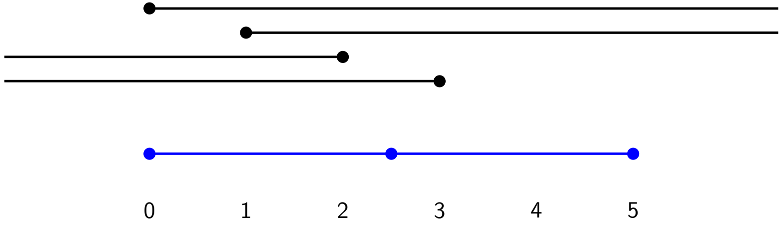
$$P = \{x \in \mathbb{R}^1 \mid x \geq 0,\ x \geq 1,\ x \leq 2,\ x \leq 3\}.$$



Example 8.1 (The ellipsoid method and binary search)

- We show that in dimension $n = 1$, the ellipsoid method closely resembles binary search, a technique to decide if intervals in the real line have a nonempty intersection.
- Consider the polyhedron

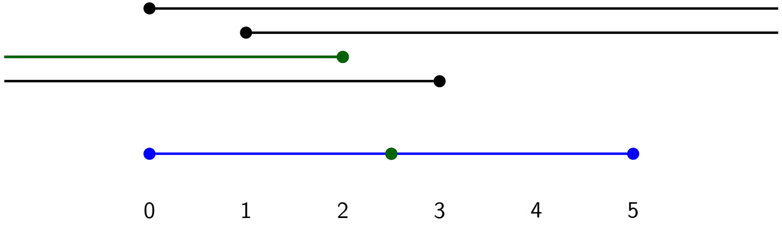
$$P = \{x \in \mathbb{R}^1 \mid x \geq 0,\ x \geq 1,\ x \leq 2,\ x \leq 3\}.$$



Example 8.1 (The ellipsoid method and binary search)

- We show that in dimension $n = 1$, the ellipsoid method closely resembles binary search, a technique to decide if intervals in the real line have a nonempty intersection.
- Consider the polyhedron

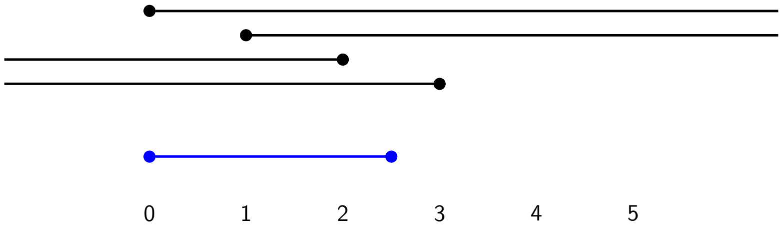
$$P = \{x \in \mathbb{R}^1 \mid x \geq 0,\ x \geq 1,\ x \leq 2,\ x \leq 3\}.$$



Example 8.1 (The ellipsoid method and binary search)

- We show that in dimension $n = 1$, the ellipsoid method closely resembles binary search, a technique to decide if intervals in the real line have a nonempty intersection.
- Consider the polyhedron

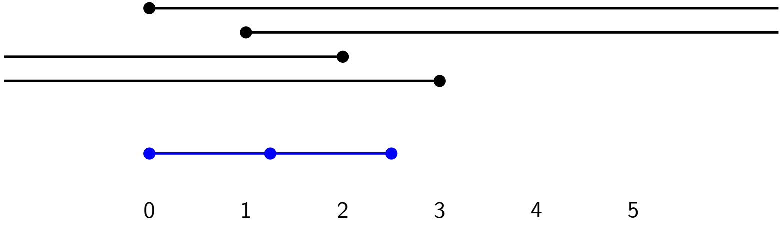
$$P = \{x \in \mathbb{R}^1 \mid x \geq 0,\ x \geq 1,\ x \leq 2,\ x \leq 3\}.$$



Example 8.1 (The ellipsoid method and binary search)

- We show that in dimension $n = 1$, the ellipsoid method closely resembles binary search, a technique to decide if intervals in the real line have a nonempty intersection.
- Consider the polyhedron

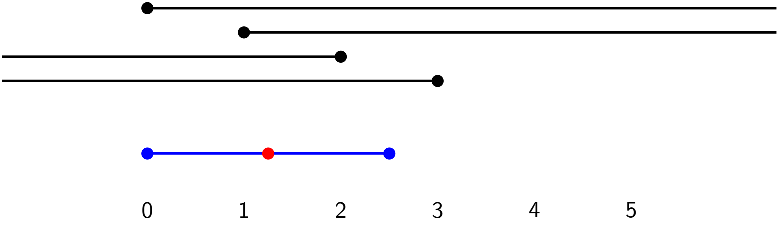
$$P = \{x \in \mathbb{R}^1 \mid x \geq 0,\ x \geq 1,\ x \leq 2,\ x \leq 3\}.$$



Example 8.1 (The ellipsoid method and binary search)

- We show that in dimension $n = 1$, the ellipsoid method closely resembles binary search, a technique to decide if intervals in the real line have a nonempty intersection.
- Consider the polyhedron

$$P = \{x \in \mathbb{R}^1 \mid x \geq 0,\ x \geq 1,\ x \leq 2,\ x \leq 3\}.$$



The assumptions of boundedness and full-dimensionality revisited

The assumptions of boundedness and full-dimensionality

- In our development of the ellipsoid method, we have made two assumptions:

Assumptions

1. The polyhedron P is bounded.
2. The polyhedron P is either empty or full-dimensional.

- Moreover, we assumed that the ball E_0 and the numbers v and V were available.

The assumptions of boundedness and full-dimensionality

- In our development of the ellipsoid method, we have made two assumptions:

Assumptions

1. The polyhedron P is bounded.
2. The polyhedron P is either empty or full-dimensional.

- Moreover, we assumed that the ball E_0 and the numbers v and V were available.
- We next show how to modify the input to the ellipsoid method if P does not necessarily satisfy these two assumptions.
- Moreover, we compute E_0, and bounds on v and V that depend only on the number of variables n and the largest number in A and b.

The assumptions of boundedness

We first relax Assumption 1 and consider possibly unbounded polyhedra.

Lemma 8.2
Let A be an $m \times n$ integer matrix and let b a vector in $\mathbb{R}^m$.
Let U be the largest absolute value of the entries in A and b.
Then, every extreme point of the polyhedron
$P = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ satisfies

$$-(nU)^n \leq x_j \leq (nU)^n \qquad j = 1, \ldots, n.$$

Proof idea:
Definition of basic feasible solution and Cramer's rule. $\square$

The assumptions of boundedness

- Lemma 8.2 establishes that all extreme points of P are contained in the bounded polyhedron P_B defined by

$$P_B = \{x \in P \mid -(nU)^n \leq x_j \leq (nU)^n,\ j = 1, \ldots, n\}.$$

- Therefore,

$$P \text{ nonempty} \quad \Leftrightarrow \quad P_B \text{ nonempty.}$$

  Question: Why? What if P has no extreme point?

- Therefore, we only need to solve the feasibility problem for P_B, which satisfies Assumption 1 since it is bounded.

The assumptions of boundedness

- Notice that P_B is a bounded polyhedron contained in the ball

$$E_0 = E(0, n(nU)^{2n} I).$$

- The volume of E_0 is less than

$$V = (2n(nU)^n)^n = (2n)^n (nU)^{n^2}.$$

- The ball E_0 and the number V will be part of the input of the ellipsoid method.

The assumptions of full-dimensionality

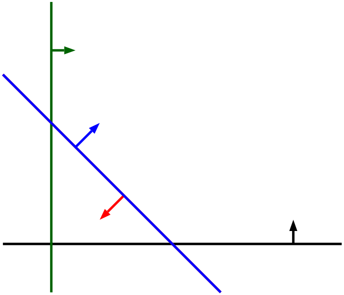
- We next discuss Assumption 2: The polyhedron P, if non empty, has full dimension, i.e., it has positive volume.
- Why do we need this assumption?
- If P is nonempty, but has dimension lower than n, then $\text{Vol}(P) = 0$.
- Because the polyhedron has zero volume, the algorithm could terminate after t^* steps and decide incorrectly that P is empty.

The assumptions of full-dimensionality

- Example:

$$P = \{(x_1, x_2) \mid x_1 + x_2 = 1,\ x_1, x_2 \geq 0\}.$$

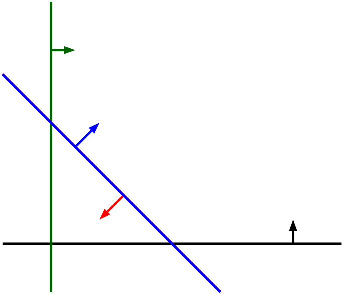
- Clearly, $\mathrm{Vol}(P) = 0$, even though P is nonempty.



The assumptions of full-dimensionality

- We next see that a small perturbation of a nonempty polyhedron produces a polyhedron that has full dimension.
- Example:

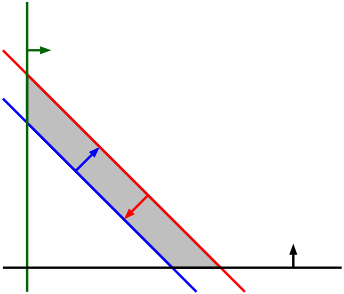
$$P = \{(x_1, x_2) \mid x_1 + x_2 \geq 1,\ x_1 + x_2 \leq 1,\ x_1, x_2 \geq 0\}$$



The assumptions of full-dimensionality

- We next see that a small perturbation of a nonempty polyhedron produces a polyhedron that has full dimension.
- Example:

$$P = \{(x_1, x_2) \mid x_1 + x_2 \geq 1,\ x_1 + x_2 \leq 1,\ x_1, x_2 \geq 0\}$$
$$P_\epsilon = \{(x_1, x_2) \mid x_1 + x_2 \geq 1 - \epsilon,\ x_1 + x_2 \leq 1 + \epsilon,\ x_1, x_2 \geq -\epsilon\}$$



The assumptions of full-dimensionality

Lemma 8.3

Let $P = \{x \in \mathbb{R}^n \mid Ax \geq b\}$. We assume that A and b have integer entries, which are bounded in absolute value by U. Let

$$\epsilon = \frac{1}{2(n+1)}((n+1)U)^{-(n+1)},$$

$$P_\epsilon = \{x \in \mathbb{R}^n \mid Ax \geq b - \epsilon e\},$$

where $e = (1, 1, \ldots, 1)$.

(a) If P is empty, then P_ϵ is empty.

(b) If P is non empty, then P_ϵ is full-dimensional.

We will not prove this.

The assumptions of full-dimensionality

Lemma 8.3

(a) If P is empty, then P_ϵ is empty.

(b) If P is non empty, then P_ϵ is full-dimensional.

- Lemma 8.3 implies that

$$P \text{ nonempty} \quad \Leftrightarrow \quad P_\epsilon \text{ nonempty}.$$

- Therefore, we only need to solve the feasibility problem for P_ϵ.
- Moreover, P_ϵ satisfies Assumption 2: it is either empty or full-dimensional.

The assumptions of full-dimensionality

- Part of the input of the ellipsoid method is a number v such that if the polyhedron P is nonempty, then $\text{Vol}(P) > v$.
- We next establish a bound on v in terms of the problem data.

Lemma 8.4

Let $P = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ be a full-dimensional bounded polyhedron, where the entries of A and b are integer and have absolute value bounded by U. Then,

$$\text{Vol}(P) > n^{-n}(nU)^{-n^2(n+1)}.$$

Proof idea:

- There exist $n+1$ affinely independent extreme points of P.
- Compute the volume of their convex hull (a simplex). $\square$

The complexity of the ellipsoid method

The complexity of the ellipsoid method

- Consider now a polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \geq b\}$, where A, b have integer entries with absolute value bounded by some U.
- If the polyhedron P satisfies Assumptions 1 and 2, we have shown in Theorem 8.2 that the ellipsoid method correctly decides whether P is empty or not in

$$O(n \log(V/v)) \quad \text{iterations.}$$

- We have seen that we can choose E_0, v, and V as follows:

$$E_0 = E(0, n(nU)^{2n} I), \quad v = n^{-n}(nU)^{-n^2(n+1)}, \quad V = (2n)^n (nU)^{n^2}.$$

- This leads to an upper bound on the number of iterations of the ellipsoid method, which is

$$n \log(2^n (nU)^{O(n^3)}) = O(n^4 \log(nU)).$$

The complexity of the ellipsoid method

- Considern now an arbitrary polyhedron P.
- We first form the bounded polyhedron P_B, and then perturb P_B as in Lemma 8.3, to form a new polyhedron $P_{B,\epsilon}$ that satisfies both our assumptions.
- As already noted,

$$P \text{ nonempty} \quad \Leftrightarrow \quad P_B \text{ nonempty} \quad \Leftrightarrow \quad P_{B,\epsilon} \text{ nonempty.}$$

- We can therefore apply the ellipsoid algorithm to $P_{B,\epsilon}$, and decide whether P is empty or not.
- It can be checked that the number of iterations is

$$O(n^6 \log(nU)).$$

The complexity of the ellipsoid method

- We wish to show that the ellipsoid method runs in polynomial time under the bit model.
- We also need to ensure that the number of arithmetic operations per iteration is polynomially bounded in n and $\log U$.
- There are two difficulties:

The complexity of the ellipsoid method

Difficulty 1.

- The computation of the new ellipsoid involves taking a square root.
- Although this might seem easy, taking square roots in a computer cannot be done exactly (the square root of an integer can be an irrational number).
- Therefore, we need to show that if we only perform calculations in finite precision, the error we make at each step of the computation will not lead to large inaccuracies in later stages of the computation.

The complexity of the ellipsoid method

Difficulty 2.

- We need to show that the numbers we generate at each step of the computation have polynomial size.
- A potential difficulty is that as numbers get multiplied, we might create numbers as large as 2^U.
- The number of bits needed to represent such a number would be $O(U)$, which is exponential in $\log U$.

The complexity of the ellipsoid method

We can overcome both these difficulties:

- It has been shown that if we only use

$$O(n^3 \log U) \text{ binary digits of precision,}$$

  the numbers computed during the algorithm have polynomially bounded size and the algorithm still correctly decides whether P is empty in $O(n^6 \log(nU))$ iterations.
- We do not cover these results as they are very technical and do not offer much insight.
- This discussion leads to the following theorem.

Theorem 8.3
The LP feasibility problem with integer data can be solved in polynomial time under the bit model.

8.4 The ellipsoid method for optimization

The ellipsoid method for optimization

- So far we have described the ellipsoid method for deciding whether a polyhedron P is empty or not.
- We next consider the following optimization problem and its dual:

$$\begin{aligned} \text{minimize} \quad & c'x \\ \text{subject to} \quad & Ax \geq b \end{aligned} \qquad\qquad \begin{aligned} \text{maximize} \quad & b'p \\ \text{subject to} \quad & A'p = c \\ & p \geq 0. \end{aligned}$$

- By strong duality, both the primal and dual optimization problems have optimal solutions if and only if the following system of linear inequalities is feasible:

$$b'p = c'x, \qquad Ax \geq b, \qquad A'p = c, \qquad p \geq 0.$$

The complexity of the ellipsoid method

- Let Q be the feasible set of this system of inequalities.
- We can apply the ellipsoid method to decide whether Q is nonempty.
- If it is indeed nonempty and a feasible solution (x, p) is obtained, then x is an optimal solution to the original optimization problem and p is an optimal solution to its dual.

The complexity of the ellipsoid method

- If the polyhedron Q is not full-dimensional and is first perturbed to Q_ϵ, as in Lemma 8.3, the ellipsoid method may terminate with some $(x_\epsilon, p_\epsilon) \in Q_\epsilon$, which does not necessarily belong to Q.
- However, provided that ϵ is sufficiently small, an element of Q can be obtained using a suitable rounding procedure.
- This fact together with Theorem 8.3, leads to the following result.

Theorem 8.4
The LP problem with integer data can be solved in polynomial time (under the bit model).

The complexity of the ellipsoid method

- An alternative but more direct method of solving the LP problem is to use the so-called sliding objective ellipsoid method, which we describe next.

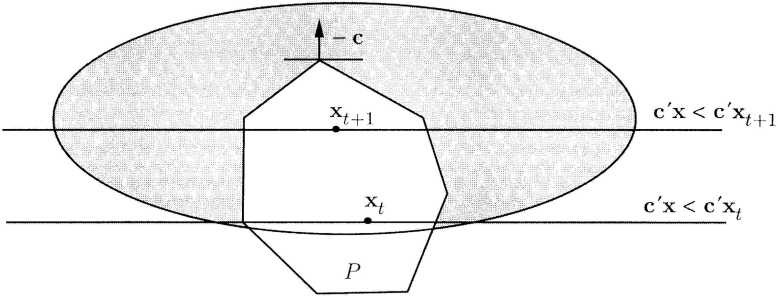
Sliding objective ellipsoid method

Sliding objective ellipsoid method

- We first run the ellipsoid method to find a feasible solution

$$x_0 \in P = \{x \in \mathbb{R}^n \mid Ax \geq b\}.$$

- It can be shown that the same ellipsoid method can be applied to LP problems involving some strict inequality constraints.

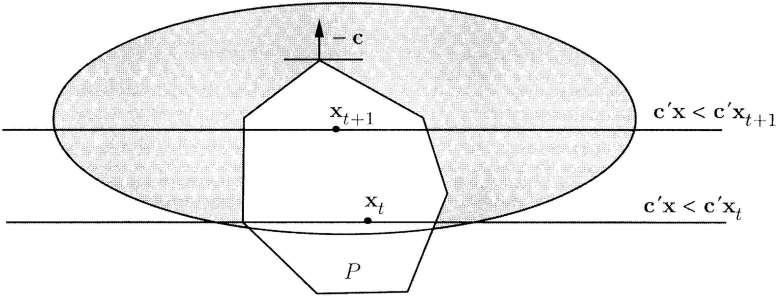


Sliding objective ellipsoid method

- We apply the ellipsoid method to decide whether the set

$$P \cap \{x \in \mathbb{R}^n \mid c'x < c'x_0\}$$

  is empty.
- If it is empty, then x_0 is optimal.
- If it is nonempty, we find a new solution $x_1 \in P$ with $c'x_1 < c'x_0$.

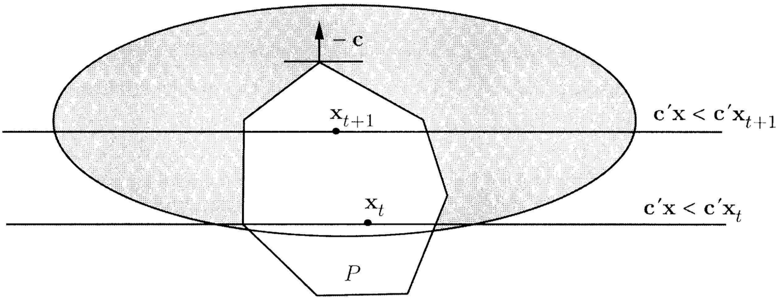


Sliding objective ellipsoid method

- More generally, every time a better feasible solution x_t is found, we take

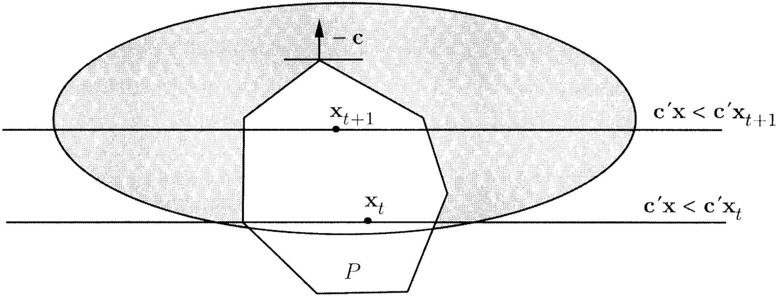
$$P \cap \{x \in \mathbb{R}^n \mid c'x < c'x_t\}$$

as the new set of inequalities and reapply the ellipsoid method.



Sliding objective ellipsoid method

- Note that in every iteration we add a constraint in the direction of vector c.
- All the constraints $c'x < c'x_t$ we add in the course of the algorithm are parallel to each other.
- This explains the name of the algorithm.



Performance of the ellipsoid method in practice

Performance of the ellipsoid method in practice

- The ellipsoid method solves LP problems in

$$O(n^6 \log(nU)) \quad \text{iterations.}$$

- Since the number of arithmetic operations per iteration is a polynomial function of n and $\log U$, this results in a polynomial number of arithmetic operations.
- This running time compares favorably with the worst-case running time of the simplex method, which is exponential.

Performance of the ellipsoid method in practice

- However, the ellipsoid method has not been practically successful, because it needs a very large number of iterations even on moderate size LP problems.
- In contrast, the theoretically inefficient simplex method needs a small number of iterations on most practical LP problems.

Performance of the ellipsoid method in practice

- This behavior of the ellipsoid method emphasizes the pitfalls in identifying

$$\text{polynomial time algorithms} \equiv \text{efficient algorithms.}$$

- The difficulty arises because we insist on a universal polynomial bound for all instances of the problem.
- The ellipsoid method achieves a polynomial bound for all instances.
- However, it typically exhibits slow convergence.

Performance of the ellipsoid method in practice

- This behavior of the ellipsoid method emphasizes the pitfalls in identifying

$$\text{polynomial time algorithms} \equiv \text{efficient algorithms.}$$

- The difficulty arises because we insist on a universal polynomial bound for all instances of the problem.
- The ellipsoid method achieves a polynomial bound for all instances.
- However, it typically exhibits slow convergence.
- There have been several proposed improvements to accelerate the convergence of the ellipsoid method, such as the idea of deep cuts (see Exercise 8.3).
- However, it does not seem that these modifications can fundamentally affect the speed of convergence.

Performance of the ellipsoid method in practice

- The ellipsoid method did not revolutionize LP.
- However, it has shown that LP is efficiently solvable from a theoretical point of view.
- In this sense, the ellipsoid method can be seen as a tool for classifying the complexity of LP problems.
- This is important, because a theoretically efficient algorithm is usually followed by the development of practical methods.
- This has been the case with interior point methods.

8.5 Problems with exponentially many constraints

Problems with exponentially many constraint

- In this section, we explain how the ellipsoid method can be applied to problems with exponentially many constraints, in order to solve them in polynomial time.
- We first describe the main ideas, and then discuss one example.

Problems with exponentially many constraint

- Consider the LP problem

$$\begin{aligned} \text{minimize} \quad & c'x \\ \text{subject to} \quad & Ax \geq b. \end{aligned}$$

- We assume that A is an $m \times n$ matrix and that the entries of A and b are integer.
- Recall that the number of iterations in the ellipsoid method is polynomial in n and $\log U$.
- In particular, the number of iterations is independent of the number m of constraints.

Problems with exponentially many constraint

- Consider the LP problem

$$\begin{aligned} \text{minimize} \quad & c'x \\ \text{subject to} \quad & Ax \geq b. \end{aligned}$$

- We assume that A is an $m \times n$ matrix and that the entries of A and b are integer.
- Recall that the number of iterations in the ellipsoid method is polynomial in n and $\log U$.
- In particular, the number of iterations is independent of the number m of constraints.
- This suggests that we may be able to solve, in time polynomial in n and $\log U$, problems in which the number m of constraints is very large, e.g., exponential in n.

Obstacle 1: Input problem data

- This turns out to be possible in several situations, but there are a couple of obstacles to be overcome.
- If, for example, $m = 2^n$, we need $\Omega(2^n)$ time just to input problem data, such as the matrix A.
- An algorithm which is polynomial in n would then appear to be impossible.

Obstacle 1: Input problem data

- The way around this obstacle is to assume that the input A and b is not given as an explicit listing of all entries.
- Instead, we will assume that A and b are given in some concise form, maybe in terms of formulas involving a relatively small number of parameters.

Obstacle 1: Input problem data

- The way around this obstacle is to assume that the input A and b is not given as an explicit listing of all entries.
- Instead, we will assume that A and b are given in some concise form, maybe in terms of formulas involving a relatively small number of parameters.
- Example: Consider the set of constraints

$$\sum_{i \in S} a_i x_i \geq |S|, \qquad \text{for all subsets } S \text{ of } \{1, \ldots, n\}.$$

- We have a total of 2^n constraints, but they are described concisely in terms of the n scalar parameters $a_1, \ldots, a_n$.

Obstacle 1: Input problem data

- We are interested in the problem

$$\begin{aligned} \text{minimize} \quad & c'x \\ \text{subject to} \quad & x \in P, \end{aligned}$$

  where P is assumed to belong to a particular family of polyhedra.
- The polyhedra in this family can have arbitrary dimension, but they must have a special structure, in the following sense.

Obstacle 1: Input problem data

- A polyhedron in this family is described by specifying:
  - the dimension n and
  - an integer vector h of primary data, of dimension $O(n^k)$, where $k \geq 1$ is some constant.
    (In our example, $h = (a_1, \ldots, a_n)$ and $k = 1$.)
- We then have some mapping which, given n and h, defines an integer matrix A, with n columns, and an integer vector b.
- No restriction is placed on the number of rows of A.
- We only need one more assumption...

Obstacle 1: Input problem data

- Let U_0 be the largest absolute value of the entries of h.
- Let U be the largest absolute value of the entries of A and b.
- Our last assumption is that there exist constants C and ℓ such that

$$\log U \leq C(n \log U_0)^{\ell}.$$

Obstacle 1: Input problem data

- Note that the size of an instance of this problem, as described by the primary problem data n and h, is

$$O(n^k \log U_0).$$

- If we apply the ellipsoid method to a problem with the above structure, the number of iterations is

$$\begin{aligned} O(n^6 \log(nU)) &= O(n^6 \log n + n^6 \log U) \\ &= O(n^6 \log n + n^{6+\ell} \log^\ell U_0), \end{aligned}$$

which is polynomial in the size of the primary problem data.

Obstacle 2: Iteration complexity

- Does this mean that we have an algorithm which is polynomial in n and $\log U_0$?
- Not necessarily, because we also need to account for the computational complexity of a typical iteration.

Obstacle 2: Iteration complexity

In a typical iteration, the only differences from the case where P is explicitly given as $P = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ lies in Steps 2(b)–(c).

> 2. (Main iteration)
>   (b) If $x_t \in P$ stop; P is nonempty.
>   (c) If $x_t \notin P$ find a violated constraint, that is, find an i such that $a_i' x_t < b_i$.

- In these steps, we need to check whether x_t is feasible and, if not, we have to display a violated constraint.
- In general, this is accomplished by examining each one of the m constraints.
- But if m is exponential in n, this would result in an exponential time algorithm.

Obstacle 2: Iteration complexity

In a typical iteration, the only differences from the case where P is explicitly given as $P = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ lies in Steps 2(b)–(c).

2. (Main iteration)
   (b) If $x_t \in P$ stop; P is nonempty.
   (c) If $x_t \notin P$ find a violated constraint, that is, find an i such that $a_i' x_t < b_i$.

- Hence, the key to a polynomial time algorithm, when m is large, hinges on our ability to carry out Steps 2(b)–(c), in polynomial time.
- These steps are important enough to have a name of their own.

Problems with exponentially many constraint

Definition 8.8
Given a polyhedron $P \subset \mathbb{R}^n$ and a vector $x \in \mathbb{R}^n$, the <u>separation problem</u> is to:

(a) Either decide that $x \in P$, or

(b) Find a vector d such that $d'x < d'y$ for all $y \in P$.

- In the terminology of Chapter 4, if $x \notin P$, then the separation problem is the problem of finding a separating hyperplane.

Problems with exponentially many constraint

- There is a small difference between the separation problem and Steps 2(b)–(c) of the ellipsoid method.
  - The ellipsoid method asks for a separating hyperplane which corresponds to one of the constraints in the description of P.
  - The separation problem asks for any separating hyperplane.
- It turns out that this difference is of no significance.
- It can be easily verified that all the properties of the ellipsoid method remain valid if we allow for general separating hyperplanes.

Problems with exponentially many constraint

- Let us now assume that we can somehow solve the separation problem in time polynomial in n and $\log U$.
- As will be seen in Example 8.2, this is sometimes possible.
- Then, the computational requirements of each iteration are polynomial in n and $\log U$.
- Hence, the overall running time of the ellipsoid method is also polynomial in n and $\log U$. (as well as polynomial in n and $\log U_0$).
- We summarize our discussion in the following theorem.

Problems with exponentially many constraint

Theorem 8.5
If we can solve the separation problem in time polynomial in n and $\log U$, then we can also solve linear optimization problems in time polynomial in n and $\log U$.
If our assumption

$$\log U \leq C n^{\ell} \log^{\ell} U_0$$

holds, the running time is also polynomial in n and $\log U_0$.

Problems with exponentially many constraint

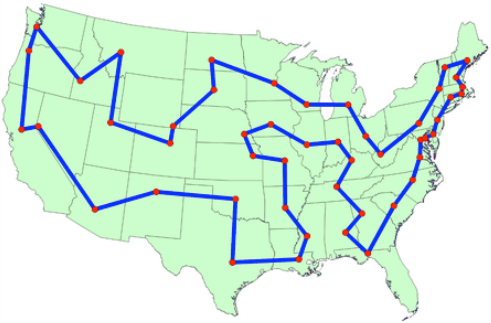
- Under some technical conditions the converse is also true:

  We can solve the separation problem in time polynomial in n and $\log U$, if and only if we can solve the optimization problem in time polynomial in n and $\log U$.

- We illustrate these ideas with the following example.
  (See also Exercise 8.10 for an application of the converse to Theorem 8.5.)

Example 8.2 (A bound for the traveling salesman problem)

- One of the most famous problems in discrete optimization, the traveling salesman problem, is defined as follows.
- Given an undirected graph $G = (\mathcal{N}, \mathcal{E})$ with n nodes, and costs c_e for every edge $e \in \mathcal{E}$, the goal is to find a tour (a cycle that visits all nodes) of minimum cost.



Example 8.2 (A bound for the traveling salesman problem)

- In order to model the problem, we define for every edge $e \in \mathcal{E}$ a variable x_e:

$$x_e = \begin{cases} 1 & \text{if edge } e \text{ is included in the tour,} \\ 0 & \text{otherwise.} \end{cases}$$

- To facilitate the formulation we define for every nonempty $S \subset \mathcal{N}$,

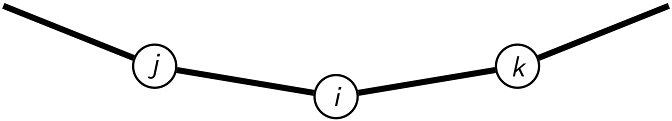
$$\delta(S) = \{e \mid e = \{i, j\},\ i \in S,\ j \notin S\}.$$

- In particular, $\delta(\{i\})$ is the set of edges incident to i.

Example 8.2 (A bound for the traveling salesman problem)

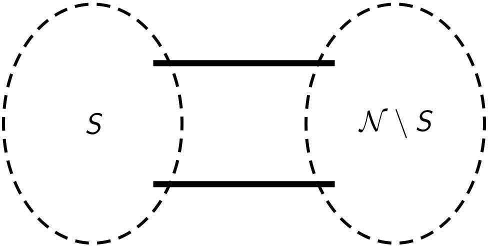
- Since in every tour, each node is incident to two edges, we have

$$\sum_{e \in \delta(\{i\})} x_e = 2 \qquad i \in \mathcal{N}.$$



Example 8.2 (A bound for the traveling salesman problem)

- In addition, if we partition the nodes into two nonempty sets S and $\mathcal{N} \setminus S$, then in every tour there are at least two edges connecting S and $\mathcal{N} \setminus S$.



- Therefore,

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad S \subset \mathcal{N},\ S \neq \emptyset, \mathcal{N}.$$

Example 8.2 (A bound for the traveling salesman problem)

- The following LP problem provides a lower bound to the optimal cost of the traveling salesman problem.

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{e \in \mathcal{E}} c_e x_e & \\
\text{subject to} & \displaystyle\sum_{e \in \delta(\{i\})} x_e = 2 & i \in \mathcal{N} \\
& \displaystyle\sum_{e \in \delta(S)} x_e \geq 2 & S \subset \mathcal{N},\ S \neq \emptyset, \mathcal{N} \\
& 0 \leq x_e \leq 1 & e \in \mathcal{E}.
\end{array}
$$

- This problem has an exponential number of constraints, because $\mathcal{N}$ has $2^n - 2$ nonempty proper subsets S.
- From the previous discussion, in order to solve this problem in polynomial time, it suffices to be able to solve the separation problem in polynomial time.

Example 8.2 (A bound for the traveling salesman problem)

We next show how to solve the separation problem in polynomial time.

- Given a vector x^*, we need to check whether it satisfies the above constraints and if not, to exhibit a violated inequality.
- We first check whether

$$\begin{aligned} \sum_{e \in \delta(\{i\})} x_e^* = 2 \quad & i \in \mathcal{N} \\ 0 \leq x_e^* \leq 1 \qquad & e \in \mathcal{E}. \end{aligned}$$

- There are only $n + 2m$ of these constraints.

Example 8.2 (A bound for the traveling salesman problem)

- We need to check whether one of the remaining constraints is violated:

$$\sum_{e \in \delta(S)} x_e^* \geq 2 \qquad S \subset \mathcal{N},\ S \neq \emptyset, \mathcal{N}$$

- To do so, we search for the subset $S \subset \mathcal{N}$, with $S \neq \emptyset, \mathcal{N}$ that minimizes

$$C(S) = \sum_{e \in \delta(S)} x_e^*.$$

- It is well-known that this set S_0 can be found in time polynomial in n.
- (This problem is known as the minimum cut problem and is equivalent to solving a maximum flow problem.)

Example 8.2 (A bound for the traveling salesman problem)

- We need to check whether one of the remaining constraints is violated:

$$\sum_{e \in \delta(S)} x_e^* \geq 2 \qquad S \subset \mathcal{N},\ S \neq \emptyset, \mathcal{N}$$

- If $C(S_0) \geq 2$, then the point x^* is feasible, since for all S,

$$\sum_{e \in \delta(S)} x_e^* \geq \sum_{e \in \delta(S_0)} x_e^* \geq 2.$$

- Otherwise, the inequality corresponding to the set S_0 is violated, i.e.,

$$\sum_{e \in \delta(S_0)} x_e^* < 2.$$

Example 8.2 (A bound for the traveling salesman problem)

- We need to check whether one of the remaining constraints is violated:

$$\sum_{e \in \delta(S)} x_e^* \geq 2 \qquad S \subset \mathcal{N},\ S \neq \emptyset, \mathcal{N}$$

- Thus, we can solve the separation problem in polynomial time.
- Therefore, by Theorem 8.5, this particular bound to the cost of an optimal traveling salesman tour can be computed in polynomial time.