Chapter 4

Linear Programming

Chapter by J. Matoušek[1]

This is mostly a digest from the book *Understanding and Using Linear Programming* by Matoušek and Gärtner, to which we refer for a more detailed presentation and additional material. The part about the ellipsoid method is extended compared to the book, in a way partially inspired by lecture notes by Petr Kolman. Details of the ellipsoid method missing in our treatment can be found, e.g., in

> M. Grötschel, L. Lovász, L. Schrijver: *Geometric Algorithms and Combinatorial Optimization*, 2nd edition, Springer, Heidelberg 1994.

4.1 Basic setting

Linear programming is one of the most powerful tools in algorithm design, and it is extremely important in practice, especially for solving optimization problems.

The word "programming" here does not refer to computer programming; rather, it comes from military slang of the 1950s, where it was used for planning logistics or deployment of men.

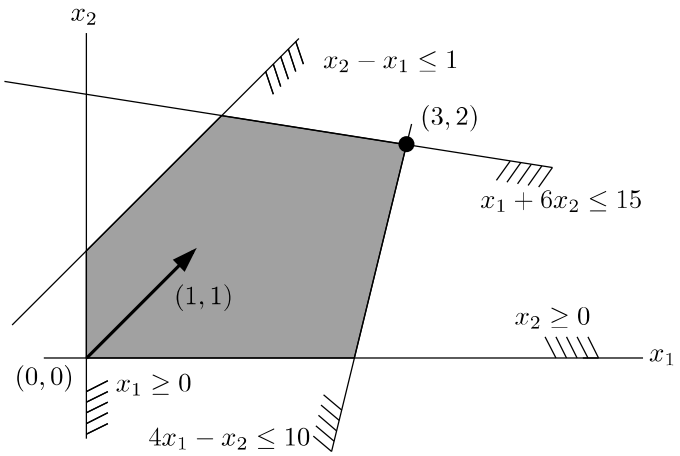
[1]Thanks to Malte Milatz, May Szedlák, and Manuel Wettstein for proofreading and very valuable comments.

Terminology of linear programs. We begin with a very simple linear programming problem (or **linear program** for short):

$$\begin{array}{ll} \text{Maximize} & x_1 + x_2 \\ \text{subject to} & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_2 - x_1 \leq 1 \\ & x_1 + 6x_2 \leq 15 \\ & 4x_1 - x_2 \leq 10. \end{array}$$

Here x_1 and x_2 are variables, which are supposed to attain real values.

For this linear program we can easily draw a picture. The set $\{\mathbf{x} \in \mathbf{R}^2 : x_2 - x_1 \leq 1\}$ is the half-plane lying below the line $x_2 = x_1 + 1$, and similarly, each of the remaining four inequalities defines a half-plane. The set of all vectors satisfying the five constraints simultaneously is a convex polygon:



Which point of this polygon maximizes the value of $x_1 + x_2$? To see this, we consider a line perpendicular to vector $(1, 1)$, drawn by the arrow, and we think of translating that line in the direction of the arrow. Then we are seeking a point where the moving line intersects our polygon for the last time.

In a general linear program we want to find a vector $\mathbf{x}^* \in \mathbf{R}^n$ maximizing (or minimizing) the value of a given linear function among all vectors $\mathbf{x} \in \mathbf{R}^n$ that satisfy a given system of linear equations and non-strict linear inequalities. The linear function to be maximized, or sometimes minimized, is called the **objective function.** It has the form $\mathbf{c}^T\mathbf{x} = c_1x_1 + \cdots + c_nx_n$, where $\mathbf{c} \in \mathbf{R}^n$ is a given vector. The linear equations and inequalities in the linear

program are called the **constraints.** It is customary to denote the number of constraints by m.

We stress that only *non-strict* linear inequalities are considered; strict inequalities, such as $x_1 + x_2 < 1$, are *not* allowed in a linear program. The reason is that, on the one hand, strict inequalities would hardly bring any additional power for modeling problems as linear programs, and on the other hand, they would destroy a nice theory concerning the existence of solutions of linear programs (which will be presented in the next sections).

Every vector $\mathbf{x} \in \mathbf{R}^n$ satisfying all constraints of a given linear program is a **feasible solution.** Each $\mathbf{x}^* \in \mathbf{R}^n$ that gives the maximum possible value of $\mathbf{c}^T\mathbf{x}$ among all feasible $\mathbf{x}$ is called an **optimal solution,** or **optimum** for short. In our linear program above we have $n = 2$, $m = 5$, and $\mathbf{c} = (1, 1)$. The only optimal solution is the vector $(3, 2)$, while, for instance, $(2, \frac{3}{2})$ is a feasible solution that is not optimal.

A linear program may in general have a single optimal solution, or infinitely many optimal solutions, or none at all.

Exercise 4.1. *(a) Modify the objective function of the linear program above so that there are infinitely many optimal solutions.*

(b) Add a constraint to the linear program above so that there are no feasible solutions (such a linear program is called **infeasible***).*

(c) Remove some of the constraints from the linear program above so that the objective function can attain arbitrarily large values, and hence there is no optimal solution (such a linear program is called **unbounded***). What is the smallest possible number of removed constraints?*

Matrix notation. The constraints in a general linear program can be both equations and inequalities. Often it is useful to convert linear programs into certain special forms, for example with no equations. These special forms may have various favorable properties, as we will see later, and they are also easier to write down in a compact notation using matrices and vectors.

For example, let us consider a linear program that has only inequalities with the $\leq$ sign (assuming all variables on the left) and such that the objective function is maximized. In a way similar to the notation $A\mathbf{x} = \mathbf{b}$ for a system of linear equations in linear algebra, such a linear program can be written as follows:

$$\text{maximize } \mathbf{c}^T\mathbf{x} \text{ subject to } A\mathbf{x} \leq \mathbf{b},$$

where A is a given $m \times n$ real matrix and $\mathbf{c} \in \mathbf{R}^n$, $\mathbf{b} \in \mathbf{R}^m$ are given vectors. The relation $\leq$ holds for two vectors of equal length if it holds componentwise.

We claim that an arbitrary linear program can be converted to this form. Indeed, we replace each equation by two opposite inequalities, and where needed, we reverse the direction of the inequalities by changing the signs. If the original linear program asked for minimization of $\mathbf{c}^T\mathbf{x}$, we replace this by maximizing $-\mathbf{c}^T\mathbf{x}$. Note that such transformation may increase the number of constraints (at most twice).

The next important exercise asks for a somewhat more complicated transformation into another special form.

Exercise 4.2. *Show that every linear program can also be converted into the following* **equational form**[2]*:*

$$\textit{maximize } \mathbf{c}^T\mathbf{x} \textit{ subject to } A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}.$$

That is, all variables are required to be nonnegative, and besides this, there are only equality constraints. What is the maximum increase in the number of variables and in the number of constraints in such a transformation? Hint: introduce new nonnegative variables to convert inequalities into equations, and express an unconstrained variable $x \in \mathbf{R}$ *as the difference of two nonnegative variables* x', x''*.*

On solving linear programs. One of the key pieces of knowledge about linear programming is this:

A linear program is efficiently solvable, both in theory and in practice.

- *In practice*, a number of software packages are available. They can handle inputs with thousands, and sometimes even millions, of variables and constraints.

- *In theory*, algorithms have been developed that provably solve each linear program in time bounded by a certain polynomial function of the input size. The input size is measured as the total number of bits

[2] In most of the literature this is called **standard form;** here we prefer the more descriptive name.

needed to write down all coefficients in the objective function and in all the constraints.

These two statements summarize the results of long and strenuous research, and efficient methods for linear programming are not simple. In this course we will not consider practically efficient algorithms, but we will explain a theoretical algorithm achieving polynomial running time.

4.2 Direct applications

Linear programming is a wonderful tool. But in order to use it, one first has to start suspecting that the considered computational problem might be expressible by a linear program, and then one has to really express it that way. Here we demonstrate a few tricks for reformulating problems that do not look like linear programs at first sight.

Once we have a suitable linear programming formulation (a "model" in the mathematical programming parlance), we can employ general algorithms. From a programmer's point of view this is very convenient, since it suffices to input the appropriate objective function and constraints into general-purpose software.

If efficiency is a concern, this need not be the end of the story. Many problems have special features, and sometimes specialized algorithms are known, or can be constructed, that solve such problems substantially faster than a general approach based on linear programming. But even for such problems, starting with a linear programming formulation makes sense: for fast prototyping, case studies, and deciding whether developing problem-specific software is worth the effort.

The diet problem and such. Historically, in the late 1940s, the first large-scale linear program solved was the determination of an adequate diet of least cost: which foods should be combined and in what amounts, so that the required amounts of all essential nutrients are satisfied and the daily ration is the cheapest possible? The linear program had 77 variables and 9 constraints, and its solution using hand-operated desk calculators took approximately 120 man-days. (This was a great achievement; the computation used the newly developed and highly non-obvious *simplex method* of George Dantzig, which remains among the most effective linear programming algorithms up until today.)

Later on, when George Dantzig had already gained access to an electronic computer, he tried to optimize his own diet as well. The optimal solution of the first linear program that he constructed recommended daily consumption of several liters of vinegar. When he removed vinegar from the next input, he obtained approximately 200 bouillon cubes as the basis of the daily diet. This story, whose truth is not entirely out of the question, does not diminish the power of linear programming in any way, but it illustrates how difficult it is to capture mathematically all the important aspects of real-life problems.

Still, from a mathematical point of view, the dietary problem is straightforward. For every possible food ingredient, say carrot or vinegar, we introduce one nonnegative variable. For every nutrient, such as protein or vitamin A, we have one inequality constraint expressing the minimum consumption. And finally, the coefficients of the objective function are the unit prices of the ingredients.

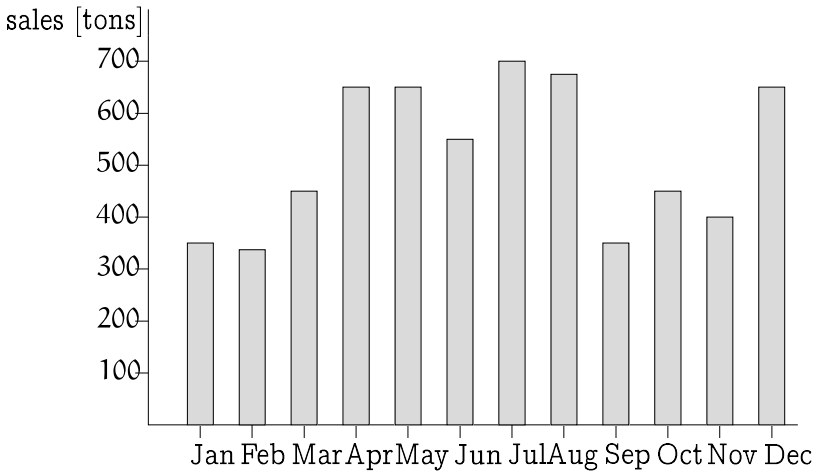
Exercise 4.3. *(a) Pick several nutrients, say proteins, carbohydrates, and sodium. Also choose a small number of specific food ingredients. Look up the recommended daily allowances for these nutrients, prices of the ingredients, and their nutrient contents. Write down the price-minimizing linear program, and see what optimal solution it gives (use some computer package, such as Matlab). Does the daily dose look like something one could live on?*

(b) (For the more adventurous ones) Extend the nutrients and ingredients lists to make them more realistic, solve the linear program, and again see how realistic it looks. Iterate until you run out of energy (or food?).

There are many applications of linear programming in industry, agriculture, services, etc. that from an abstract point of view are variations of the diet problem and do not introduce substantially new mathematical tricks. It may still be challenging to design good models for real-life problems of this kind, but the challenges are not mathematical.

We will proceed with less obvious applications.

Ice cream all year round. An ice cream manufacturer needs to set up a production plan for the next year, based on the following prediction of monthly sales:



A simple solution would be to produce "just in time," meaning that all the ice cream needed in each month is also produced in that month. However, a change in the produced amount has significant costs: temporary workers have to be hired or laid off, machines have to be adjusted, and so on.

Another simple solution might be a completely "flat" production schedule, with the same amount produced every month. Some thought reveals that such a schedule need not be feasible if we want to end up with zero surplus at the end of the year. But even if it is feasible, it need not be ideal either, since storing ice cream incurs a nontrivial cost.

We want a compromise minimizing the total cost resulting both from changes in production and from storage of surpluses.

Let us denote the demand in month i by $d_i \geq 0$ (in tons). Then we introduce a nonnegative variable x_i for the production in month i and another nonnegative variable s_i for the total surplus in store at the end of month i. To meet the demand in month i, we may use the production in month i and the surplus at the end of month $i-1$:

$$x_i + s_{i-1} \geq d_i \quad \text{for } i = 1, 2, \ldots, 12.$$

The quantity $x_i + s_{i-1} - d_i$ is exactly the surplus after month i, and thus we have

$$x_i + s_{i-1} - s_i = d_i \quad \text{for } i = 1, 2, \ldots, 12.$$

Assuming that initially there is no surplus, we set $s_0 = 0$ (if we took the production history into account, s_0 would be the surplus at the end of the

previous year). We also set $s_{12} = 0$, unless we want to plan for another year.

Among all nonnegative solutions of these equations and inequalities, we are looking for one that minimizes the total cost. Let us assume that changing the production by 1 ton from month $i-1$ to month i costs €50, and that storage facilities for 1 ton of ice cream cost €20 per month. Then the total cost is expressed by the function

$$50 \sum_{i=1}^{12} |x_i - x_{i-1}| + 20 \sum_{i=1}^{12} s_i,$$

where we set $x_0 = 0$ (again, history can easily be taken into account).

Unfortunately, this cost function is not linear. Fortunately, there is a simple but important trick that allows us to make it linear, at the price of introducing extra variables.

The change in production is either an increase or a decrease. Let us introduce a nonnegative variable y_i for the increase from month $i-1$ to month i, and a nonnegative variable z_i for the decrease. Then

$$x_i - x_{i-1} = y_i - z_i$$

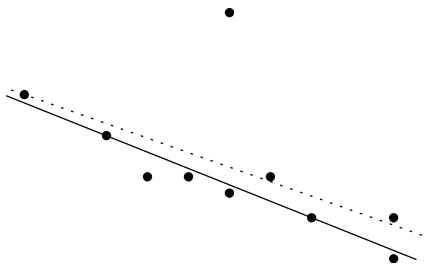
A production schedule of minimum total cost is given by an optimal solution of the following linear program:

$$\begin{array}{ll} \text{Minimize} & 50 \sum_{i=1}^{12} y_i + 50 \sum_{i=1}^{12} z_i + 20 \sum_{i=1}^{12} s_i \\ \text{subject to} & x_i + s_{i-1} - s_i = d_i \text{ for } i = 1, 2, \ldots, 12 \\ & x_i - x_{i-1} = y_i - z_i \text{ for } i = 1, 2, \ldots, 12 \\ & x_0 = 0 \\ & s_0 = 0 \\ & s_{12} = 0 \\ & x_i, s_i, y_i, z_i \geq 0 \text{ for } i = 1, 2, \ldots, 12. \end{array}$$

To see that an optimal solution $(\mathbf{s}^*, \mathbf{y}^*, \mathbf{z}^*)$ of this linear program indeed defines a schedule, we need to note that one of y_i^* and z_i^* has to be zero for all i, for otherwise, we could decrease both and obtain a better solution. This implies that $y_i^* + z_i^* = |x_i - x_{i-1}|$ equals the change in production from month $i-1$ to month i, as required.

The pattern of this example is quite general, and many problems of optimal control can be solved via linear programming in a similar manner.

Fitting a line. Suppose that we have some data points $(x_1, y_1), \ldots, (x_n, y_n)$, and we would like to fit a line approximating the dependence of y on x:



How can one formulate mathematically that a given line "best fits" the points? There is no unique way, and several different criteria are commonly used for line fitting in practice.

The most popular one is the method of *least squares*, which for given points $(x_1, y_1), \ldots, (x_n, y_n)$ seeks a line with equation $y = ax+b$ minimizing the expression

$$\sum_{i=1}^{n} (ax_i + b - y_i)^2. \tag{4.1}$$

In words, for every point we take its vertical distance from the line, square it, and sum these "squares of errors."

This method need not always be the most suitable. For instance, if a small number of exceptional points are measured with very large error, they can influence the resulting line a great deal. An alternative method, less sensitive to a small number of "outliers," is to minimize the sum of absolute values of all errors:

$$\sum_{i=1}^{n} |ax_i + b - y_i|. \tag{4.2}$$

By a trick similar to the one we saw in the ice-cream example, this apparently nonlinear optimization problem can be captured by a linear program:

$$\begin{array}{lll} \text{minimize} & e_1 + e_2 + \cdots + e_n & \\ \text{subject to} & e_i \geq ax_i + b - y_i & \text{for } i = 1, 2, \ldots, n \\ & e_i \geq -(ax_i + b - y_i) & \text{for } i = 1, 2, \ldots, n. \end{array}$$

The variables are a, b, and $e_1, e_2, \ldots, e_n$ (while $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ are given numbers). Each e_i is an auxiliary variable standing for the error

at the ith point. The constraints guarantee that

$$e_i \geq \max\Big(ax_i + b - y_i, -(ax_i + b - y_i)\Big) = |ax_i + b - y_i|.$$

In an optimal solution each of these inequalities has to be satisfied with equality, for otherwise, we could decrease the corresponding e_i. Thus, an optimal solution yields a line minimizing the expression (4.2).

In the picture above, the solid line has been fitted by this method, while the dotted line has been obtained using least squares.

Cutting paper rolls. A paper mill manufactures rolls of paper of a standard width of 3 meters. But customers want to buy paper rolls of shorter width, and the mill has to cut such rolls from the 3 m rolls. One 3 m roll can be cut, for instance, into two rolls 93 cm wide, one roll of width 108 cm, and a rest of 6 cm (which goes to waste).

Let us consider an order of

- 97 rolls of width 135 cm,
- 610 rolls of width 108 cm,
- 395 rolls of width 93 cm, and
- 211 rolls of width 42 cm.

What is the smallest number of 3 m rolls that have to be cut in order to satisfy this order, and how should they be cut?

In order to engage linear programming one has to be generous in introducing variables. We write down all of the requested widths: 135 cm, 108 cm, 93 cm, and 42 cm. Then we list all possibilities of cutting a 3 m paper roll into rolls of some of these widths (we need to consider only possibilities for which the wasted piece is shorter than 42 cm):

P1:	2×135	P7:	$108 + 93 + 2 \times 42$
P2:	$135 + 108 + 42$	P8:	$108 + 4 \times 42$
P3:	$135 + 93 + 42$	P9:	3×93
P4:	$135 + 3 \times 42$	P10:	$2 \times 93 + 2 \times 42$
P5:	$2 \times 108 + 2 \times 42$	P11:	$93 + 4 \times 42$
P6:	$108 + 2 \times 93$	P12:	7×42

For each possibility Pj on the list we introduce a variable $x_j \geq 0$ representing the number of rolls cut according to that possibility. We want to minimize the total number of rolls cut, i.e., $\sum_{j=1}^{12} x_j$, in such a way that the customers are satisfied. For example, to satisfy the demand for 395 rolls of width 93 cm we require

$$x_3 + 2x_6 + x_7 + 3x_9 + 2x_{10} + x_{11} \geq 395.$$

For each of the widths we obtain one constraint.

For a more complicated order, the list of possibilities would most likely be produced by computer. We would be in a quite typical situation in which a linear program is not entered "by hand," but rather is generated by some computer program.

In our specific problem, an optimal solution of the resulting linear program has $x_1 = 48.5$, $x_5 = 206.25$, $x_6 = 197.5$, and all other components 0.

In order to cut 48.5 rolls according to the possibility P1, one has to unwind half of a roll. Here we need more information about the technical possibilities of the paper mill: Is making thinner rolls and cutting them technically and economically feasible? If yes, we have solved the problem optimally. If not, we have to work further and somehow take into account the restriction that only feasible solutions of the linear program with *integral* x_i are of interest. This is not at all easy in general, and we will say a little more about this issue later.

4.3 Geometry of linear programs

Convex sets and polyhedra. We recall several geometric notions.

- A set $C \subseteq \mathbf{R}^n$ is **convex** if C contains the segment connecting every two of its points. That is, for every $\mathbf{x}, \mathbf{y} \in C$ and every $t \in [0,1]$ we have $(1-t)\mathbf{x} + t\mathbf{y} \in C$.

- A **hyperplane** in $\mathbf{R}^n$ is an affine subspace of dimension $n-1$. In other words, it is a set of the form $\{\mathbf{x} \in \mathbf{R}^n : a_1x_1 + a_2x_2 + \cdots + a_nx_n = b\}$, where $a_1, a_2, \ldots, a_n$ are not all 0.

- A (closed) **half-space** in $\mathbf{R}^n$ is a set of the form $\{\mathbf{x} \in \mathbf{R}^n : a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b\}$, again with at least one a_i nonzero.

- A **convex polyhedron** in $\mathbf{R}^n$ is the intersection of finitely many closed half-spaces.[3]

The reader probably knows convex polygons in the plane and convex polytopes in $\mathbf{R}^3$, such as the Platonic solids. A convex polytope in $\mathbf{R}^n$ can be defined as a *bounded* convex polyhedron (i.e., one contained in a sufficiently large ball). Arbitrary convex polyhedra need not be bounded; an example is a single half-space.

The intersection of a collection of convex sets is clearly convex, and a half-space is convex. Hence a convex polyhedron is indeed a convex set.

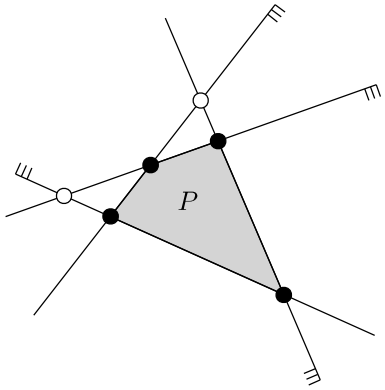
The set of all feasible solutions of a linear program is, more or less by definition, a convex polyhedron in $\mathbf{R}^n$.

Basic feasible solutions. Let us first consider an arbitrary linear program, with both equality and inequality constraints, whose feasible set is a convex polyhedron $P \subseteq \mathbf{R}^n$. A **basic feasible solution** of such a linear program is a feasible solution $\mathbf{x}$ for which n linearly independent constraints hold with equality.

This probably needs some explanation. First, an equality constraint holds with equality for every feasible solution, while an inequality constraint may either hold with equality, or with a strict inequality (and the latter case does not count in the definition of a basic feasible solution). Second, with every constraint $a_1x_1 + \cdots + a_nx_n ? b$, where $?$ is one of $=$, $\leq$, or $\geq$, we associate the vector $\mathbf{a} = (a_1, \ldots, a_n)$ of the coefficients on the left-hand side, and linear independence of constraints means linear independence of the corresponding vectors.

We note that for every n-tuple of linearly independent constraints, there is at most one point satisfying all of them with equality (this is basic linear algebra). Consequently, each linear program has only finitely many basic feasible solutions, namely, at most $\binom{m}{n}$. The following figure shows an example with $m = 4, n = 2$. There are $\binom{4}{2} = 6$ points where two (linearly independent) constraints hold with equality. 4 of them (the black ones) are feasible and are therefore basic feasible solutions.

[3] All of $\mathbf{R}^n$ also counts as a convex polyhedron; it can be regarded as the intersection of an empty set of half-spaces.



We remark that, geometrically, basic feasible solutions correspond to vertices of the polyhedron P (we will not prove that). A point $\mathbf{v} \in P$ is called a *vertex* if there is a linear function whose maximum over P is attained in $\mathbf{v}$ and nowhere else. That is, there is $\mathbf{c} \in \mathbf{R}^n$ with $\mathbf{c}^T\mathbf{v} > \mathbf{c}^T\mathbf{x}$ for all $\mathbf{x} \in P \setminus \{\mathbf{v}\}$.

An arbitrary linear program need not have any basic feasible solutions at all (consider a single inequality constraint in $\mathbf{R}^2$). However, as we will see, linear programs in equational form (see Exercise 4.2) have "enough" basic feasible solutions.

Theorem 4.1. *Let us consider a linear program in equational form*

$$\textit{maximize } \mathbf{c}^T\mathbf{x} \textit{ subject to } A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0},$$

and let $P \subseteq \mathbf{R}^n$ be the convex polyhedron of all feasible solutions. If $P \neq \emptyset$ and the objective function $\mathbf{c}^T\mathbf{x}$ is bounded from above on P, then there exists a basic feasible solution that is optimal, and in particular, the linear program has an optimal solution.

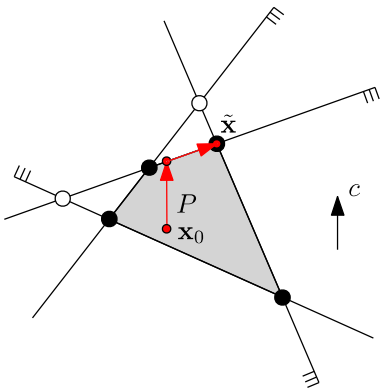
In particular, the theorem tells us that an optimal solution may fail to exist only for obvious reasons: either if there are no feasible solutions at all, or if the objective function is unbounded from above.

Proof. The theorem is a consequence of the following statement:

> *If the objective function is bounded from above, then for every feasible solution $\mathbf{x}$ there exists a basic feasible solution $\tilde{\mathbf{x}}$ with $\mathbf{c}^T\tilde{\mathbf{x}} \geq \mathbf{c}^T\mathbf{x}$.*

How does this statement imply the theorem? If the linear program is feasible and bounded, then according to the statement, for every feasible solution there is a basic feasible solution with the same or larger objective function. Since there are only finitely many basic feasible solutions, at least one of them has to give the maximum value of the objective function, which means that it is optimal.

In order to prove the statement, we maintain a candidate $\tilde{x}$ for the required basic feasible solution (initially $\tilde{x} = x$, the feasible solution we start with). The plan is as follows: if n linearly independent constraints hold with equality at $\tilde{x}$ already, then $\tilde{x}$ is a basic feasible solution, and we are done. Otherwise, the constraints holding with equality at $\tilde{x}$ actually hold with equality in an affine subspace of dimension at least 1, containing $\tilde{x}$. Within this subspace, we will move $\tilde{x}$ into a suitable direction w in such a way that (a) the objective function value does not decrease, and (b) a new inequality constraint eventually holds with equality. From there, we repeat the process until we cannot move anymore, meaning that $\tilde{x}$ has become a basic feasible solution:



Formally, we define an index set

$$K = \{j \in \{1, 2, \ldots, n\} : \tilde{x}_j > 0\}.$$

Let A_K be the submatrix of the $m \times n$ matrix A consisting of the columns indexed by K. We distinguish two cases, depending on whether the columns of A_K are linearly independent.

If they are linearly independent (in particular, $|K| \leq m$ then), we claim that $\tilde{x}$ is a basic feasible solution. Indeed, if we form an $(m + n - |K|) \times n$ matrix B by appending $n - |K|$ new rows $e_j, j \notin K$, to A, then B has

$|K|+n-|K| = n$ linearly independent columns and therefore also n linearly independent rows.[4] Moreover, the corresponding constraints (for the new rows, these are $x_j \geq 0, j \notin K$) hold with equality at $\tilde{\mathbf{x}}$.

So let us suppose that the columns of A_K are linearly dependent, which means that there is a nonzero $|K|$-component vector $\mathbf{v}$ such that $A_K\mathbf{v} = \mathbf{0}$. We extend $\mathbf{v}$ by zeros in positions outside K to an n-component vector $\mathbf{w}$; thus $A\mathbf{w} = A_K\mathbf{v} = \mathbf{0}$.

Let us assume for a moment that $\mathbf{w}$ satisfies the following two conditions (we will show later why we can assume this):

(i) $\mathbf{c}^T\mathbf{w} \geq 0$.

(ii) There exists $j \in K$ with $w_j < 0$.

For a real number $t \geq 0$ let us consider the vector $\mathbf{x}(t) = \tilde{\mathbf{x}} + t\mathbf{w}$. We show that for some suitable $t_1 > 0$ the vector $\mathbf{x}(t_1)$ is a feasible solution with more zero components than $\tilde{\mathbf{x}}$ (more inequalities that hold with equality at this points). At the same time, $\mathbf{c}^T\mathbf{x}(t_1) = \mathbf{c}^T\tilde{\mathbf{x}} + t_1\mathbf{c}^T\mathbf{w} \geq \mathbf{c}^T\mathbf{x}_0 + t_1\mathbf{c}^T\mathbf{w} \geq \mathbf{c}^T\mathbf{x}_0$, and so we can replace our candidate $\tilde{\mathbf{x}}$ by $\mathbf{x}(t_1)$. As the number of components is bounded, this process must eventually stop in which case $\tilde{\mathbf{x}}$ is a basic feasible solution.

Concerning the existence of t_1, we argue as follows: We have $A\mathbf{x}(t) = \mathbf{b}$ for all t since $A\mathbf{x}(t) = A\tilde{\mathbf{x}} + tA\mathbf{w} = A\tilde{\mathbf{x}} = \mathbf{b}$, because $\tilde{\mathbf{x}}$ is feasible. Moreover, for $t = 0$ the vector $\mathbf{x}(0) = \tilde{\mathbf{x}}$ has all components from K strictly positive and all other components zero (and latter stay zero for all t). For the jth component of $\mathbf{x}(t)$ we have $x(t)_j = \tilde{x}_j + tw_j$, and if $w_j < 0$ as in condition (ii), we get $x(t)_j < 0$ for all sufficiently large $t > 0$. If we begin with $t = 0$ and let t grow, then those $x(t)_j$ with $w_j < 0$ are decreasing, and at a certain moment $\tilde{t}$ the first of these decreasing components reaches 0. At this moment, we have found the desired $\tilde{t} = t_1$.

Now what do we do if the vector $\mathbf{w}$ fails to satisfy condition (i) or (ii)? If $\mathbf{c}^T\mathbf{w} = 0$, then (i) holds and (ii) can be recovered by changing the sign of $\mathbf{w}$ (since $\mathbf{w} \neq \mathbf{0}$). So we assume $\mathbf{c}^T\mathbf{w} \neq 0$, and again after a possible sign change we can achieve $\mathbf{c}^T\mathbf{w} > 0$ and thus (i). Now if (ii) fails, we must have $\mathbf{w} \geq 0$. But this means that $\mathbf{x}(t) = \tilde{\mathbf{x}} + t\mathbf{w} \geq 0$ for all $t \geq 0$, and hence all such $\mathbf{x}(t)$ are feasible. The value of the objective function for $\mathbf{x}(t)$ is

[4] $\mathbf{e}_j$ is the j-th unit vector.

$\mathbf{c}^T\mathbf{x}(t) = \mathbf{c}^T\tilde{\mathbf{x}} + t\mathbf{c}^T\mathbf{w}$, and it tends to infinity as $t \to \infty$. Hence the linear program is unbounded. This concludes the proof. □

4.4 Bounds on solutions

Encoding size. For algorithmic purposes, we will consider only linear programs in which all of the coefficients are rational numbers (so that they have a finite encoding, unlike arbitrary real numbers). In order to speak of theoretically efficient, i.e., polynomial, algorithms, we also need to measure the **size** of a given linear program.

In a nutshell, the size of a linear program is the total number of bits needed to write all of the coefficients. More formally, first we define the **encoding size** of an integer z as $\langle z\rangle := \lceil \log_2(|z|+1)\rceil + 1$ (this is the length of the standard binary encoding plus one bit for the sign). Immediate but useful facts are $\langle xy\rangle \le \langle x\rangle + \langle y\rangle$ and $|x| \le 2^{\langle x\rangle}$.

For a rational number r, we write it as $r = \frac{p}{q}$ with p and q relatively prime and set $\langle r\rangle := \langle p\rangle + \langle q\rangle$; for a rational matrix A we have $\langle A\rangle := \sum_{i=1}^{m}\sum_{j=1}^{n}\langle a_{ij}\rangle$, and similarly for a rational vector. Then for a linear program L, the encoding size is the sum of encoding sizes of the vectors and matrices involved. For example, if L is of the form "maximize $\mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} \le \mathbf{b}$," then $\langle L\rangle := \langle A\rangle + \langle \mathbf{b}\rangle + \langle \mathbf{c}\rangle$.

Bounds on optimal solutions. The following bounds are very useful in many algorithmic considerations concerning linear programming.

Theorem 4.2. *If a linear program L with rational coefficients has a feasible solution, then it also has a rational feasible solution $\tilde{\mathbf{x}}$ with $\langle \tilde{x}_j\rangle = O(\langle L\rangle)$ for every j (consequently, $\tilde{\mathbf{x}}$ is contained in the cube $[-K, K]^n$ with $K \le 2^{O(\langle L\rangle)}$). A similar statement holds for optimal solutions.*

The constant in the $O(.)$ notation in the above theorem could be made explicit, but we prefer not to distract the reader's attention with more detailed calculations. We also note that actually $\langle \tilde{x}_j\rangle = O(\langle L\rangle - \langle \mathbf{c}\rangle)$, where $\mathbf{c}$ is the coefficient vector of the objective function; this will be immediate from the proof.

For the proof of the theorem, we need bounds on the encoding size of the determinant of a rational matrix.

Lemma 4.3. *For a rational $n \times n$ matrix A we have $\langle \det A\rangle = O(\langle A\rangle)$.*

Proof. Let us write $a_{ij} = p_{ij}/q_{ij}$ with p_{ij}, q_{ij} relatively prime for all i, j. By definition, $\langle A \rangle = \sum_{i,j}(\langle p_{ij} \rangle + \langle q_{ij} \rangle)$. Let $N = \prod_{i,j} p_{ij}, D = \prod_{i,j} q_{ij}$. Using the above fact $\langle xy \rangle \leq \langle x \rangle + \langle y \rangle$, we get $\langle N \rangle \leq \langle A \rangle$ and $\langle D \rangle \leq \langle A \rangle$.

We have the usual formula $\det A = \sum_{\pi \in S_n} \operatorname{sign}(\pi) \prod_{i=1}^{n} a_{i,\pi(i)}$ with $n!$ terms. As the denominator $\prod_{i=1}^{n} q_{i,\pi(i)}$ of $\prod_{i=1}^{n} a_{i,\pi(i)}$ divides D for all π, we can write $\det A$ in the form

$$\det(A) = \frac{\sum_{\pi \in S_n} \operatorname{sign}(\pi) N_\pi}{D},$$

where $N_\pi = D \prod_{i=1}^{n} p_{i,\pi(i)} / \prod_{i=1}^{n} q_{i,\pi(i)} \leq DN$ is an integer. Hence,

$$\langle \det A \rangle \leq \langle \sum_{\pi \in S_n} \operatorname{sign}(\pi) N_\pi \rangle + \langle D \rangle \leq \langle n! DN \rangle + \langle D \rangle.$$

Using $\langle xy \rangle \leq \langle x \rangle + \langle y \rangle$ again along with $\langle N \rangle, \langle D \rangle \leq \langle A \rangle$, we conclude that

$$\langle \det A \rangle \leq O(n \log n + \langle A \rangle) = O(\langle A \rangle),$$

since $\langle A \rangle \geq n^2$. □

Proof of Theorem 4.2. It suffices to prove the statement concerning optimal solutions, since if we take the zero objective function, every feasible solution is optimal.

First we assume that the given linear program is in equational form. Since it has an optimal solution, it also has a basic optimal solution $\tilde{\mathbf{x}} \in \mathbf{R}^n$ according to Theorem 4.1. By definition, $\tilde{\mathbf{x}}$ attains some n linearly independent constraints with equality.

Hence $\tilde{\mathbf{x}}$ is the solution of a system of n linear equations of the form $\tilde{A}\mathbf{x} = \tilde{\mathbf{b}}$, where $\tilde{A}$ is an $n \times n$ nonsingular matrix; each of the equations is either one of the equations of the system $A\mathbf{x} = \mathbf{b}$, or of the form $x_j = 0$ for some j.

We recall from linear algebra that, by Cramer's rule,

$$\tilde{x}_j = \frac{\det \tilde{A}_j}{\det \tilde{A}}, \qquad j = 1, 2, \ldots, n,$$

where $\tilde{A}_j$ is obtained from $\tilde{A}$ by replacing the jth column with $\tilde{b}$. So using Lemma 4.3, $\langle \tilde{x}_j \rangle = O(\langle \tilde{A}_j \rangle + \langle \tilde{A} \rangle) = O(\langle L \rangle - \langle \mathbf{c} \rangle)$ as needed.

Finally, if the given linear program is not in equational form, we transform it to equational form as in Exercise 4.2. This increases the encoding

size at most by a constant factor. Passing from a feasible solution of the transformed linear program to a feasible solution of the original linear program does not increase the size of the components, as the reader may want to check (returning to the solution of Exercise 4.2), and similarly for optimal solutions. This concludes the proof. □

4.5 Duality and the Farkas lemma

Duality is arguably the most important theoretical result about linear programs. We begin the exposition with a simpler version of the duality of linear programming, called the Farkas lemma.

4.5.1 The Farkas lemma

Let us say that we have completed a linear algebra homework, one of finding a solution of a system $A\mathbf{x} = \mathbf{b}$ of linear equations. How can we convince people that the homework is solved? If we have a solution $\tilde{\mathbf{x}}$, then we just write it down, and everyone equipped with a calculator can check that it actually satisfies the given equations.

But what if the system has no solution? One way is to write down a linear combination of the equations that is obviously inconsistent. Namely, if A is an $m \times n$ matrix and we exhibit a vector $\mathbf{y} \in \mathbf{R}^m$ such that $\mathbf{y}^T A = \mathbf{0}^T$ and $\mathbf{y}^T\mathbf{b} = 1$, then it is clear that $A\mathbf{x} = \mathbf{b}$ has no solution. Indeed, if we multiply the ith equation in the system by y_i and add the resulting equations together, we obtain the obviously unsolvable equation $\mathbf{0}^T\mathbf{x} = 1$.

Easy linear algebra shows that unsolvability of a system of linear equations can *always* be certified in this way.

Exercise 4.4. *Prove that a system $A\mathbf{x} = \mathbf{b}$ of linear equations is unsolvable if and only if there is $\mathbf{y}$ with $A^T\mathbf{y} = \mathbf{0}$ and $\mathbf{b}^T\mathbf{y} = 1$.*

Thus, for the decision problem "does a given system of linear equations have a solution" there are easy certificates both for the YES and NO answers. (The reader probably knows that NP-complete problems are asymmetric in this respect—with polynomial-time certificates for YES but, according to a generally shared belief, with no polynomial-time certificates for NO.)

The Farkas lemma provides easy certificates for unsolvability of systems of linear *inequalities*.

Lemma 4.4 (Farkas lemma I). *A system $Ax \leq b$ of linear inequalities is* unsolvable *if and only if there exists $y \geq 0$ such that $A^T y = 0$ and $b^T y < 0$.*

Be sure to realize why y as in the lemma certifies unsolvability!

Similar results, also commonly called Farkas lemmas, are available for certifying the nonexistence of a nonnegative solution of a system of linear equations, as well of a nonnegative solution of a system of linear inequalities:

Lemma 4.5 (Farkas lemma II and III). *A system $Ax = b$ of linear equations has no nonnegative solution if and only if there exists y such that $A^T y \geq 0$ and $b^T y < 0$.*

A system $Ax \leq b$ of linear inequalities has no nonnegative solution if and only if there exists $y \geq 0$ such that $A^T y \geq 0$ and $b^T y < 0$.

Exercise 4.5. *(a) Explain why the y as in the last lemma indeed certify the nonexistence of a nonnegative solution.*

(b) Prove that all of the three variants of the Farkas lemma above, I–III, are mutually equivalent. (Or do at least one of the implications.) You may want to look to Exercise 4.2 for inspiration.

On proofs of the Farkas lemma. As a fundamental result, the Farkas lemma has a number of proofs. Some of them are quite intuitive, and some of them are rather short, but so far we have not succeeded in finding a proof in the intersection of these two classes.

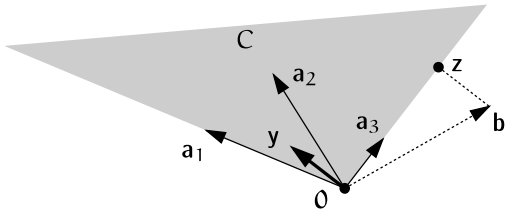
We will indicate a geometric meaning and proof of Farkas lemma II, omitting a proof of an intuitive but nontrivial fact. So we have a $m \times n$ matrix A and a vector $b \in \mathbf{R}^m$, and we seek a nonnegative x with $Ax = b$.

We interpret the columns of A as vectors $a_1, \ldots, a_n \in \mathbf{R}^m$. Let $C = \{x_1 a_1 + \cdots + x_n a_n : x_1, \ldots, x_n \geq 0\}$ be the set of all nonnegative linear combinations of these vectors. Geometrically, C is the **convex cone** generated by $a_1, \ldots, a_n$; an alternative description is that we first replace each a_j by an infinite ray in the direction of a_j, and then we take the convex hull of these rays.

The system $Ax = b$ has a nonnegative solution precisely if $b \in C$. Farkas lemma II then asserts that if $b \notin C$, then there exists y with $y^T a_j \geq 0$ for

all j and $\mathbf{y}^T\mathbf{b} < 0$. Geometrically, this means that all the $\mathbf{a}_j$ lie on one side of the hyperplane $h := \{\mathbf{x} \in \mathbf{R}^m : \mathbf{y}^T\mathbf{x} = 0\}$, and $\mathbf{b}$ lies (strictly) on the other side.

Thus, geometrically, the claim of Farkas lemma II amounts to a result about separating a point from a convex cone by a hyperplane: every $\mathbf{b} \notin C$ can be strictly separated from C by a hyperplane passing through the origin. This has an "easy" geometric proof: let $\mathbf{z}$ be the point of C closest to $\mathbf{b}$; then $\mathbf{y} = \mathbf{z} - \mathbf{b}$ works, as is indicated in the picture.



The problem with this is in proving rigorously that the desired nearest point $\mathbf{z}$ exists (once we know that, it is not hard to complete the proof rigorously). One needs to show that the cone C is a closed set (then a standard compactness argument can be applied), which is not really hard, but we do not know of any short proof.

For the purposes of this course, we will stay with this semi-formal treatment, referring to the Matoušek–Gärtner linear programming book or other sources for rigorous proofs of the Farkas lemma.

4.5.2 The strong duality theorem

Now we go back to linear programs, and this time we consider a linear program of the form

$$\text{maximize } \mathbf{c}^T\mathbf{x} \text{ subject to } A\mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \geq \mathbf{0}. \tag{P}$$

Similar to the Farkas lemma, we can also interpret the duality of linear programming as the existence of easy certificates for NO answers—but what is the question? Well, we ask, "Is the optimal value of (P) greater or equal to some given number γ?"

A YES answer has an obvious certificate, namely, a feasible solution $\tilde{\mathbf{x}}$ with $\mathbf{c}^T\tilde{\mathbf{x}} \geq \gamma$. How can we hope to certify NO?

Suppose that we can make a nonnegative linear combination of the rows of the matrix A in which every coefficient is at least as large as the

corresponding coefficient of $\mathbf{c}$; i.e., we have a vector $\mathbf{y} \geq 0$ with $\mathbf{y}^\top A \geq \mathbf{c}$. Then the corresponding linear combination of the right-hand sides, i.e., $\mathbf{y}^\top \mathbf{b}$, is an upper bound for the maximum of $\mathbf{c}^\top \mathbf{x}$. Indeed, we have $\mathbf{c}^\top \mathbf{x} \leq (\mathbf{y}^\top A)\mathbf{x} = \mathbf{y}^\top (A\mathbf{x}) \leq \mathbf{y}^\top \mathbf{b}$, where the first inequality relies on $\mathbf{x} \geq \mathbf{0}$ and the second on $\mathbf{y} \geq \mathbf{0}$.

Now the task of finding the *best* $\mathbf{y}$ as above, i.e., one that gives the smallest upper bound on $\mathbf{c}^\top \mathbf{x}$, can be written as a linear program:

$$\text{minimize } \mathbf{b}^\top \mathbf{y} \text{ subject to } A^\top \mathbf{y} \geq \mathbf{c} \text{ and } \mathbf{y} \geq \mathbf{0}. \tag{D}$$

This is called the **dual** of the linear program (P).

By the previous considerations, we see that the minimum of (D) is always at least as large as the maximum of (P)—this (easy) result is called the **weak duality theorem** of linear programming.

The *strong duality theorem* tells us that, actually, the minimum of (D) is *equal* to the maximum of (P), assuming (P) feasible and bounded. In this respect, the optimum of (D) is a perfect NO certificate.

Here is a formal statement of the duality theorem, which also discusses what happens if one of (P), (D) is infeasible or unbounded.

Theorem 4.6 (Strong duality theorem). *For the linear programs (P) and (D) as above, exactly one of the following possibilities occurs:*

1. *Neither (P) nor (D) has a feasible solution.*
2. *(P) is unbounded and (D) has no feasible solution.*
3. *(P) has no feasible solution and (D) is unbounded.*
4. *Both (P) and (D) have a feasible solution. Then both have an optimal solution, and if $\mathbf{x}^*$ is an optimal solution of (P) and $\mathbf{y}^*$ is an optimal solution of (D), then*

$$\mathbf{c}^\top \mathbf{x}^* = \mathbf{b}^\top \mathbf{y}^*.$$

Exercise 4.6. *Find an example of a specific linear program (P) for each of the cases in the theorem.*

The duality theorem easily implies several significant min-max theorems in combinatorics and optimization, such as the maxflow–mincut theorem,

König's theorem about matchings in bipartite graphs, Hall's marriage theorem, and others. Often it also yields, with no extra effort, a more general result involving weights, which sometimes does not follow from the usual combinatorial proof.

The duality theorem is valid for each linear program, not only for one of the form (P); we have only to construct the dual linear program properly. To this end, we can transform the given linear program to the form (P), and then the dual linear program has the form (D). The result can often be simplified; for example, the difference of two nonnegative variables can be replaced by a single unbounded variable (one that may attain all real values).

Simpler than doing this again and again is to use a general recipe for dualizing linear programs, as detailed, e.g., in the Matoušek–Gärtner book.

There are several proofs of the duality theorem, some of them using the Farkas lemma and some not.

Proof of strong duality. Weak duality makes sure that if (P) is unbounded, then (D) must be infeasible, and if (D) is unbounded, then (P) must be infeasible. So it suffices to show that if (P) is feasible and bounded, then so is (D), and their optimal values coincide. (We also need the same statement with the role of (P) and (D) interchanged, but this follows immediately, since it can be easily checked that (P) is the dual of (D).)

Let $\mathbf{x}^*$ be an optimum of (P) and let $\gamma = \mathbf{c}^\mathsf{T}\mathbf{x}^*$ be the optimum value. The system of inequalities

$$A\mathbf{x} \le \mathbf{b},\ \mathbf{c}^\mathsf{T}\mathbf{x} \ge \gamma + \varepsilon \tag{4.3}$$

has a nonnegative solution for $\varepsilon = 0$, but it has no nonnegative solution for $\varepsilon > 0$.

Let us fix $\varepsilon > 0$. Farkas lemma III tells us that, because of the unsolvability of (4.3), there is a nonnegative vector $\mathbf{y} = (\mathbf{u}, z)$ such that $A^\mathsf{T}\mathbf{u} \ge z\mathbf{c}$ and $\mathbf{b}^\mathsf{T}\mathbf{u} < z(\gamma + \varepsilon)$. (If you do not believe derive this carefully; start by writing the matrix of (4.3) down explicitly.) We also have $\mathbf{b}^\mathsf{T}\mathbf{u} \ge z\gamma$, for otherwise, $\mathbf{y}$ would witness nonexistence of a nonnegative solution of (4.3) for $\varepsilon = 0$, but we assume that there is a nonnegative solution.

In order that the inequalities $\mathbf{b}^\mathsf{T}\mathbf{u} < z(\gamma + \varepsilon)$ and $\mathbf{b}^\mathsf{T}\mathbf{u} \ge z\gamma$ both hold, we must have $z > 0$. Let us set $\mathbf{v} := \frac{1}{z}\mathbf{u}$. Then we get

$$A^\mathsf{T}\mathbf{v} \ge \mathbf{c},\ \mathbf{b}^\mathsf{T}\mathbf{v} < \gamma + \varepsilon.$$

In other words, $\mathbf{v}$ is a feasible solution of (D), with the value of the objective function smaller than $\gamma + \varepsilon$.

By the weak duality theorem, every feasible solution of (D) has value of the objective function at least γ. Hence (D) is a feasible and bounded linear program, and so we know that it has an optimal solution $\mathbf{y}^*$ (Theorem 4.1). Its value $\mathbf{b}^\top \mathbf{y}^*$ is between γ and $\gamma+\varepsilon$ for every $\varepsilon > 0$, and thus it equals γ. □

4.6 The ellipsoid method

The ellipsoid method was invented in 1970 by Shor, Judin, and Nemirovski as an algorithm for certain nonlinear optimization problems. In 1979 Leonid Khachyian outlined, in a short note, how linear programs can be solved by this method in provably polynomial time.

The world press made a sensation out of this since the journalists contorted the result and presented it as an unprecedented breakthrough in practical computational methods (giving the Soviets a technological edge over the West...). However, the ellipsoid method has never been interesting for the practice of linear programming—Khachyian's discovery was indeed extremely significant, but for the theory of computational complexity. It solved an open problem that many people had attacked in vain for many years. The solution was conceptually utterly different from previous approaches, which were mostly variations of the simplex method.

What is meant by a polynomial algorithm? We say that an algorithm is a **polynomial algorithm for linear programming** if a polynomial $p(x)$ exists such that for every linear program L with rational coefficients the algorithm finds a correct solution in at most $p(\langle L \rangle)$ steps.

The steps are counted in some of the usual models of computation, for example, as steps of a Turing machine (usually the chosen computational model is not crucial; whatever is polynomial in one model is also polynomial in other reasonable models). We stress that a single arithmetic operation is not counted as a single step here! We count as steps operations with single bits, and hence, e.g., addition of two k-bit integers requires at least k steps.

It may be useful to mention that the usual algorithms for Gaussian elimination, as taught in basic linear algebra, are not polynomial. As is well known, Gaussian elimination for an $n \times n$ matrix uses at most $O(n^3)$

arithmetic operations, but unless special provisions are made in the implementation, the intermediate values appearing in the computation may have exponentially many digits, even if the original matrix has only small integer entries. (All of this concerns *exact* computations, while many implementations use floating-point arithmetic and hence the numbers are continually rounded. But then there is no guarantee that the results are correct.) Polynomial-time versions of Gaussian elimination are known, but they are quite sophisticated.

Other linear-programming algorithms. As we have remarked, the ellipsoid method is not practical for actually solving large linear programs. There are basically two classes of methods used in practice, each with many variations.

The *simplex method* is the original approach of Dantzig, but with many sophisticated improvements. Its basic strategy is to go along the edges of the polyhedron of feasible solutions, from one basic feasible solution to another, in such a way that the value of the objective function improves in each step.

Interior point methods start in the interior of the feasible polyhedron and proceed towards an optimal solution, in discrete steps but guided by certain smooth, analytically defined curve inside the polyhedron (but again, there are many versions and some do not quite conform this scheme).

Some of the interior-point methods are claimed to be polynomial in the literature, but it seems hard to find a clean proof of polynomiality for the Turing machine model.

In practice, both simplex and interior-point methods may be competitive, depending on the structure of the linear program, but for very large (and usually sparse) linear programs, interior point seems to be the winner.

4.6.1 The relaxed problem and the algorithm

The ellipsoid method does not directly solve a linear program, but rather it seeks a solution of a system of linear inequalities $A\mathbf{x} \leq \mathbf{b}$, if one exists—this is called the **feasibility problem.**

A polynomial algorithm for the feasibility problem allows us to solve a general linear program in polynomial time. An elegant way of seeing this is via the strong duality theorem: We transform the linear program to the form "maximize $\mathbf{c}^\top\mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$." We set up the following

system of inequalities with vectors $\mathbf{x}$ and $\mathbf{y}$:

$$A\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0},\ \mathbf{c}^T\mathbf{x} \geq \mathbf{b}^T\mathbf{y},\ A^T\mathbf{y} \geq \mathbf{c},\ \mathbf{y} \geq \mathbf{0}. \tag{4.4}$$

By strong duality, this system is feasible exactly if the original linear program has an optimal solution, and if $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is a feasible solution of (4.4), then $\tilde{\mathbf{x}}$ is an optimum of the original linear program.

Exercise 4.7. *Work out a different way of using an algorithm for the feasibility problem for solving linear programs, based on the idea of binary search for the optimum value. Theorem 4.2 is useful for analyzing the number of steps of the binary search.*

The relaxed feasibility problem. Let $P = \{\mathbf{x} \in \mathbf{R}^n : A\mathbf{x} \leq \mathbf{b}\}$ be the polyhedron consisting of all solutions of the considered system of inequalities; we thus want to find a point $\mathbf{y} \in P$, or conclude that $P = \emptyset$.

For a simpler exposition, we will first explain an algorithm for the following relaxed problem, in which we make an additional assumption about P, and we also allow the algorithm to answer incorrectly if P is too flat in a sense specified next.

Problem 4.7 (Relaxed feasibility problem). *Together with the matrix A and vector $\mathbf{b}$ we are given rational numbers $R > \varepsilon > 0$. We assume that the polyhedron P is contained in the ball $B(\mathbf{0}, R)$ centered at $\mathbf{0}$ with radius R. If P contains a ball of radius ε, then the algorithm has to return a point $\mathbf{y} \in P$. However, if P contains no ball of radius ε, then the algorithm may return either some $\mathbf{y} \in P$, or the answer NO SOLUTION.*

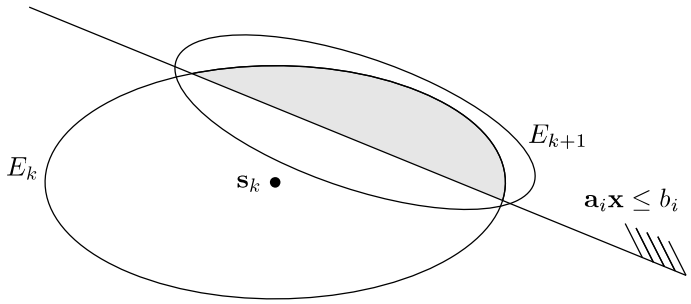
The ellipsoid method algorithm for Problem 4.7 generates a sequence of ellipsoids $E_0, E_1, \ldots, E_t$, each of them guaranteed to contain all of P. A rough outline is as follows:

1. Set $k = 0$ and $E_0 = B(\mathbf{0}, R)$.

2. Let $\mathbf{s}_k$ be the center of the current ellipsoid E_k. If $\mathbf{s}_k$ satisfies all inequalities of the system $A\mathbf{x} \leq \mathbf{b}$, return $\mathbf{s}_k$ as a solution; **stop.**

3. Otherwise, choose an inequality of the system that is violated by $\mathbf{s}_k$. Let it be the ith inequality; so we have $\mathbf{a}_i^T\mathbf{s}_k > b_i$. Compute E_{k+1} as an ellipsoid containing the set $E_k \cap \{\mathbf{x} \in \mathbf{R}^n : \mathbf{a}_i^T\mathbf{x} \leq b\}$ and such that

the $\operatorname{vol} E_{k+1}$, the volume of E_{k+1}, is substantially smaller than $\operatorname{vol} E_k$ (Lemma 4.8 below gives a specific version of "substantially smaller"). Here is a picture:



4. If $\operatorname{vol} E_{k+1}$ is smaller than the volume of a ball of radius ε, return NO SOLUTION; **stop.** Otherwise, increase k by 1 and continue with Step 2.

A crucial geometric fact, captured by the following lemma, is that the ellipsoid E_{k+1} as in Step 3 always exists.

Lemma 4.8. *Let E be an n-dimensional ellipsoid in $\mathbf{R}^n$ with center $\mathbf{s}$, and let H be a closed half-space whose interior does not contain $\mathbf{s}$. Then there exists an ellipsoid E', given by an explicit formula, that contains $E \cap H$ and satisfies*

$$\operatorname{vol} E' \le \rho \cdot \operatorname{vol} E, \qquad \text{where } \rho = \rho(n) := e^{-1/(2n+2)}.$$

The lemma will be proved in Section 4.6.2.

Assuming the lemma, it is easy to bound the number of iterations of the above algorithm. By the lemma we have $\operatorname{vol} E_k \le \rho^k \operatorname{vol} B(0, R)$. Since the volume of an n-dimensional ball is proportional to the nth power of the radius, for k satisfying $R \cdot e^{-k/n(2n+2)} < \varepsilon$ the volume of E_k is smaller than that of a ball of radius ε. Such k provides an upper bound of $\lceil n(2n+2) \ln(R/\varepsilon) \rceil$ on the maximum number of iterations. So the algorithm solves Problem 4.7 in a number of iterations at most polynomial in $n + \log(R/\varepsilon)$.

In order to obtain a polynomial algorithm for linear programming from this, besides proving Lemma 4.8, we still need to overcome the following two issues.

(A) Show that the iterations in the above algorithm can be implemented in polynomial time.

(B) Show that, given a polynomial algorithm for Problem 4.7, one can obtain a polynomial algorithm for deciding the feasibility of an arbitrary system $A\mathbf{x} \leq \mathbf{b}$.

A rigorous treatment of all of (A) would be too lengthy (and boring) for our purposes; we will thus omit a part of it. As a consequence, we will not be able to present a full proof of polynomiality of the ellipsoid method for linear programming, but no significant ideas will be left out—the missing part is laborious but routine.

4.6.2 Geometry of ellipsoids

The goal here is proving Lemma 4.8, but first we recall some basic facts about ellipsoids.

A two-dimensional ellipsoid is an ellipse plus its interior. An ellipse in axial position in the plane has equation $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$, where a, b are the lengths of the semi-axes. An arbitrary ellipse can be obtained from one in axial position by rotation and translation.

An n-dimensional ellipsoid could be introduced analogously, but for us, it seems cleaner to introduce ellipsoids as images of balls under nonsingular affine maps. Let

$$B^n = \{\mathbf{x} \in \mathbf{R}^n : \mathbf{x}^T\mathbf{x} \leq 1\}$$

be the n-dimensional ball of unit radius centered at $\mathbf{0}$. Then an n-dimensional **ellipsoid** is a set of the form

$$E = \{M\mathbf{x} + \mathbf{s} : \mathbf{x} \in B^n\},$$

where M is a nonsingular $n \times n$ matrix and $\mathbf{s} \in \mathbf{R}^n$ is a vector.

The mapping $\mathbf{x} \mapsto M\mathbf{x} + \mathbf{s}$ is a composition of a linear mapping and a translation, and it is called an **affine map.** Since, as is taught in linear algebra, a linear map $\mathbf{x} \mapsto M\mathbf{x}$ maps a set of volume v to a set of volume $v \cdot |\det M|$, and since a translation does not change volume, we have $\operatorname{vol} E = |\det M| \operatorname{vol} B^n$.

By manipulating the definition we can describe the ellipsoid E by an inequality:

$$\begin{aligned} E &= \{\mathbf{y} \in \mathbf{R}^n : M^{-1}(\mathbf{y} - \mathbf{s}) \in B^n\} \\ &= \{\mathbf{y} \in \mathbf{R}^n : (\mathbf{y} - \mathbf{s})^T (M^{-1})^T M^{-1}(\mathbf{y} - \mathbf{s}) \leq 1\} \\ &= \{\mathbf{y} \in \mathbf{R}^n : (\mathbf{y} - \mathbf{s})^T Q^{-1}(\mathbf{y} - \mathbf{s}) \leq 1\}, \end{aligned} \tag{4.5}$$

where we have set $Q = MM^T$. It is well known and easy to check that such a Q is a positive definite matrix, that is, a symmetric square matrix satisfying $\mathbf{x}^T Q \mathbf{x} > 0$ for all nonzero vectors $\mathbf{x}$. Conversely, from matrix theory it is known that each positive definite matrix Q can be factored as $Q = MM^T$ for some nonsingular square matrix M. Therefore, an equivalent definition is that an ellipsoid is a set described by (4.5) for some positive definite Q and some $\mathbf{s}$.

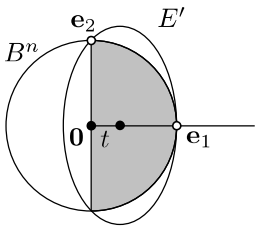
Geometrically, $\mathbf{s}$ is the center of the ellipsoid E. If Q is a diagonal matrix and $\mathbf{s} = 0$, then we have an ellipsoid in *axial position*, of the form

$$\left\{ \mathbf{y} \in \mathbf{R}^n : \frac{y_1^2}{q_{11}} + \frac{y_2^2}{q_{22}} + \cdots + \frac{y_n^2}{q_{nn}} \leq 1 \right\},$$

and in particular, we recover the usual equation of an ellipse mentioned above. The numbers $\sqrt{q_{11}}, \sqrt{q_{22}}, \ldots, \sqrt{q_{nn}}$ are the lengths of the semiaxes of E.

Proof of Lemma 4.8. First we consider a very special case of the lemma, where $E = B^n$ is the unit ball, $\mathbf{s} = 0$, and H is the special half-space $H_0 = \{x_1 \geq 0\}$.

Since the set $B^n \cap H_0$ is rotationally symmetric around the x_1-axis, it is natural to obtain a suitable E' by squeezing the ball B^n in the x_1-direction, and expanding it appropriately in the other directions.



We can thus write the affine map making B^n into E' in the form $\mathbf{x} \mapsto M\mathbf{x} + t\mathbf{e}_1$, where $t > 0$ is a yet unknown parameter determining the translation

and M is a diagonal matrix of the form

$$\begin{pmatrix} a & & & & \\ & b & & & \\ & & b & & \\ & & & \ddots & \\ & & & & b \end{pmatrix},$$

where $a \in (0,1)$ is the squeezing factor along the x_1-axis and $b > 1$ is the expansion factor in the other directions.

The volume of E' is proportional to $\det M = ab^{n-1}$, so we will want to minimize that expression. Intuitively, it should be sufficient to guarantee that E' of the above form contains the points e_1 and e_2, which leads to the inequalities

$$a \geq 1 - t \quad \text{and} \quad \frac{t^2}{a^2} + \frac{1}{b^2} \leq 1,$$

as the reader may want to check carefully. Assuming that both of these inequalities hold with equality, we express a and b in terms of t, obtaining $a = 1 - t$, $b = (1-t)/\sqrt{1-2t}$, and

$$ab^{n-1} = \frac{(1-t)^n}{(1-2t)^{(n-1)/2}}.$$

We minimize the last expression as a function of t; this is a calculus problem which, fortunately, has a simple solution, namely, $t = 1/(n+1)$. Then $a = 1 - \frac{1}{n+1}$ and b can be simplified to $\sqrt{1 + \frac{1}{(n+1)(n-1)}}$.

The calculation up until now can be regarded as heuristic; we should now verify carefully that not only the points e_1 and e_2, but all of the half-ball are contained in E'. By rotational symmetry, it suffices to verify that every point $(x_1, x_2, 0, \ldots, 0)$ with $x_1 \geq 0$ and $x_1^2 + x_2^2 \leq 1$ belongs to E'. We leave this calculation as an exercise.

The ratio $\mathrm{vol}(E')/\mathrm{vol}(B^n)$ equals ab^{n-1}. Using the well-known inequality $1 + x \leq e^x$, we have $a \leq \exp\frac{-1}{n+1}$ and $b \leq \exp\frac{1}{2(n+1)(n-1)}$, and so $ab^{n-1} \leq \exp(\frac{-1}{n+1} + \frac{1}{2(n+1)}) = e^{-1/(2n+2)}$—this is the value ρ claimed in the lemma.

It remains to deal with the general case, where E is an arbitrary ellipsoid with center s and H is an arbitrary half-space whose interior avoids s. First, by translation, we may assume $s = 0$. Second, we may assume that the

boundary hyperplane of H passes through the center of E (since translating H away from the origin makes $H \cap E$ smaller in inclusion).

We want to find a linear map $T\colon \mathbf{R}^n \to \mathbf{R}^n$ such that $T(B^n) = E$ and $T(H_0) = H$, where H_0 is the half-space $\{x_1 \geq 0\}$ considered in the first part of the proof. Indeed, if E_0' is an ellipsoid enclosing $B^n \cap H_0$ as in the lemma, then the ellipsoid $E' = T(E_0')$ encloses $E \cap H$ and has volume at most $\rho \cdot \operatorname{vol} E$, since affine maps change all volumes in the same ratio.

How can we compute the matrix of T? We may assume that E is represented by a linear map M with $E = M(B^n)$ (there is no translation since here we assume E centered at 0). The only problem is that M, in general, is not going to map the normal vector $\mathbf{e}_1$ of H_0 to the normal vector, call it $\mathbf{u}$, of H. To rectify this, let $\mathbf{v}$ be the preimage of $\mathbf{u}$ normalized so that $||\mathbf{v}|| = 1$; i.e., $\mathbf{v} = M^{-1}(\mathbf{u})/||M^{-1}(\mathbf{u})||$. According to the following exercise, we find a rotation R of $\mathbf{R}^n$ with $R(\mathbf{e}_1) = \mathbf{v}$, and we set $T = MR$ (first we rotate, then we apply M). This concludes the proof. □

Exercise 4.8. *Let $\mathbf{v} \in \mathbf{R}^n$ be a unit vector. Find a rotation R with $R(\mathbf{e}_1) = \mathbf{v}$ (write down the matrix of R w.r.t. the standard basis of $\mathbf{R}^n$).*

Implementing the iterations of the ellipsoid method. Now we consider issue (A) from the previous section, implementing the iterations. At first sight, it may seem that there is no problem and that we can just implement the method from the above proof. However, the formulas derived in that proof contain square roots, and so we cannot compute the matrices representing the ellipsoids E_k exactly in rational arithmetic.

The usual solution is to compute only approximately, with a suitable accuracy (which will generally depend on the size of the input—note that we can afford to compute with b-bit numbers, where b is a polynomial function of the input size).

In order to guarantee that the polyhedron P is still contained in E_k even after the numbers representing E_k are rounded, we first expand E_k slightly. Then one has to verify that rounding does not destroy the properties of the algorithm, and this is the somewhat lengthy and tedious part which we chose to omit in our treatment.

4.6.3 Solving linear programs using the relaxed feasibility problem

Here we explain how the algorithm for the relaxed feasibility problem (Problem 4.7) can be used to decide the feasibility of an arbitrary system $A\mathbf{x} \leq \mathbf{b}$ of linear inequalities, while preserving polynomial running time. This is issue (B) at the end of Section 4.6.1. For brevity, let us write $\varphi = \langle A \rangle + \langle \mathbf{b} \rangle$ for the encoding size of the considered system.

We will construct a new system $\hat{A}\mathbf{x} \leq \hat{\mathbf{b}}$, whose encoding size is polynomial in φ, and such that

(i) $\hat{P} \subset B(0, R)$, where $\langle R \rangle$ is polynomial in φ.

(ii) If $P \neq \emptyset$, then $\hat{P}$ contains an ε-ball, with $\varepsilon > 0$ and $\langle \varepsilon \rangle$ polynomial in φ.

(iii) If $P = \emptyset$, then $\hat{P} = \emptyset$.

As the reader has surely guessed, P and $\hat{P}$ denote the sets of all feasible solutions of $A\mathbf{x} \leq \mathbf{b}$ and of $\hat{A}\mathbf{x} \leq \hat{\mathbf{b}}$, respectively.

Boundedness. Making the polyhedron bounded is easy using Theorem 4.2, which tells us that if $P \neq \emptyset$, then $P \cap [-K, K]^n \neq \emptyset$ as well, where $K := 2^{C_1 \varphi}$, with a suitable constant C_1. So we simply add the inequalities

$$-K \leq x_j \leq K, \quad j = 1, 2, \ldots, n$$

to the system, obtaining a new system $A'\mathbf{x} \leq \mathbf{b}'$ with encoding size $\varphi' = O(n\varphi)$.

Thickening the polyhedron. The next lemma takes care of "thickening" the polyhedron so that, if it was nonempty, it will contain an ε-ball. The nontrivial part is to make sure that thickening an empty polyhedron cannot create a nonempty one.

Lemma 4.9. *Let $A\mathbf{x} \leq \mathbf{b}$ be a system of inequalities with rational coefficients and encoding size φ, let P be its polyhedron of feasible solutions, and let P_η be the polyhedron of the system $A\mathbf{x} \leq \mathbf{b} + \eta\mathbf{1}$, where $\mathbf{1}$ is the vector of all 1s. For $\eta = 2^{-C_2\varphi}$, with a sufficiently large constant C_2, the following hold.*

(a) *If $P \neq \emptyset$, then P_η contains an ε-ball for $\varepsilon = \eta/2^\varphi$.*

(b) *If $P = \emptyset$, then $P_\eta = \emptyset$.*

Proof. Part (a) is straightforward: we just observe that if $\mathbf{a}$ is one of the rows of A, we have $|\mathbf{a}^T(\mathbf{x}-\mathbf{x}')| \le ||\mathbf{a}|| \cdot ||\mathbf{x}-\mathbf{x}'|| \le 2^\varphi ||\mathbf{x}-\mathbf{x}'||$, and it follows that if $\mathbf{x} \in P$ and $||\mathbf{x}'-\mathbf{x}|| \le \varepsilon$, then $\mathbf{x}' \in P_\eta$.

Part (b) is proved using the Farkas lemma. Namely, if $A\mathbf{x} \le \mathbf{b}$ has no solution, then by Farkas lemma I (Lemma 4.4) there is $\mathbf{y}$ with $A^T\mathbf{y} = \mathbf{0}$ and $\mathbf{b}^T\mathbf{y} < 0$; by rescaling we can assume $\mathbf{b}^T\mathbf{y} = -1$.

Then Theorem 4.2 tells us that there also exists such a $\mathbf{y}$ with $\langle \mathbf{y} \rangle = O(\varphi)$. We claim that this $\mathbf{y}$ also witnesses infeasibility of $A\mathbf{x} \le \mathbf{b} + \eta\mathbf{1}$. Indeed, it suffices to check that $(\mathbf{b}+\eta\mathbf{1})^T\mathbf{y} < 0$, or in other words, that $-\mathbf{1}^T\mathbf{y} < 1/\eta$. But $|\mathbf{1}^T\mathbf{y}| \le \sum_{i=1}^m |y_i| \le 2^{O(\langle y \rangle)} = 2^{O(\varphi)}$, where the constant in $O(.)$ does not depend on C_2 in the definition of η, so a sufficiently large C_2 will do. □

Returning to the plan at the beginning of this section, given the original system $A\mathbf{x} \le \mathbf{b}$, the new system $\hat{A}\mathbf{x} \le \hat{\mathbf{b}}$ is

$$A\mathbf{x} \le \mathbf{b} + \eta\mathbf{1}, \quad -K - \eta \le x_j \le K + \eta, \quad j = 1, 2, \ldots, n$$

(of course, adding the tiny number η to the gigantic number K is ridiculous and unnecessary, but logically appropriate, since first we produce a system whose feasible polyhedron is guaranteed to be bounded, and then we thicken that polyhedron). The above discussion implies that conditions (i)—(iii) are satisfied, with $R = (K+\eta)\sqrt{n}$ and $\varepsilon = \eta/2^{\varphi'}$. The encoding size of the new system is at most $O(n\varphi) = O(\varphi^2)$ as needed.

Actually finding a solution. We may seem to be done, but the algorithm we have so far only *decides* whether $A\mathbf{x} \le \mathbf{b}$ has a solution, but it does not necessarily provide one—indeed, it returns a solution of the auxiliary system $\hat{A}\mathbf{x} \le \hat{\mathbf{b}}$, and this need not solve the original system.

This time we let the reader discover an algorithm in a guided exercise.

Exercise 4.9. *Suppose that we have an oracle that, given a system of linear inequalities, decides its feasibility (outputs YES or NO). Design an algorithm that computes a solution of a given system of linear equations and inequalities, provided that one exists, in polynomial time and with polynomially many calls of the oracle.*

(a) How can we proceed if there are only equations in the system?

(b) If there is at least one inequality, use the oracle to check if there is a solution satisfying that inequality with equality, and take appropriate actions depending on the outcome.

We conclude our treatment of the ellipsoid method with two potentially useful remarks.

Why ellipsoids? They are used in the ellipsoid method since they constitute probably the simplest class of n-dimensional convex sets that is closed under nonsingular affine maps. Popularly speaking, this class is rich enough to approximate all convex polyhedra including flat ones and needle-like ones. If desired, ellipsoids can be replaced by simplices, for example, but the formulas in the algorithm and its analysis become considerably more unpleasant than those for ellipsoids.

The ellipsoid method need not know all of the linear program. The system of inequalities $A\mathbf{x} \leq \mathbf{b}$ can also be given by means of a **separation oracle.** This is an algorithm (black box) that accepts a point $\mathbf{s} \in \mathbf{R}^n$ as input, and if $\mathbf{s}$ is a solution of the system, it returns the answer YES, while if $\mathbf{s}$ is not a solution, it returns one (arbitrary) inequality of the system that is violated by $\mathbf{s}$. (Such an inequality separates $\mathbf{s}$ from the solution set, and hence the name separation oracle.) The ellipsoid method calls the separation oracle with the centers $\mathbf{s}_k$ of the generated ellipsoids, and it always uses the violated inequality returned by the oracle for determining the next ellipsoid.

We talk about this since a separation oracle can be implemented efficiently for some interesting optimization problems even when the full system has exponentially many inequalities or even infinitely many.

Probably the most important example of a situation in which an infinite system of linear inequalities can be solved by the ellipsoid method is **semidefinite programming.** This is one of the most significant directions in optimization in the 1990s, similar to linear programming dominating the 1970s. But this is another story belonging to another course.

Addendum by E. Welzl[5]

4.7 Traveling Salesman

Given a graph $G = (V, E)$ with costs $c_e \in \mathbf{R}$, $e \in E$, a *tour* τ is a spanning cycle (also called Hamiltonian cycle), formally a subset E_τ of E such that (i) the graph (V, E_τ) is connected and (ii) every vertex is incident to exactly two edges in E_τ. The *cost* of tour τ is defined as $\sum_{e\in E_\tau} c_e$. An *optimal tour in* G is a tour of minimal cost. The task of computing an optimal tour for a given graph G is often called the *traveling salesman problem*,[6] a classical NP-complete problem.

For $S \subseteq V$, let $\delta(S) := \{e \in E : |e \cap S| = 1\}$, sometimes called the *boundary of* S or the edge set of the cut $(S, V \setminus S)$. For $v \in V$, we write $\delta(v)$ short for $\delta(\{v\})$, the set of edges incident to v. With this we can specify the characteristic vectors of edge sets of tours by the following constraints.

$$\begin{aligned} x &\in \{0,1\}^E \\ \sum_{e\in\delta(v)} x_e &= 2\,, \quad \text{for all } v \in V, \text{ and} \\ \sum_{e\in\delta(S)} x_e &\geq 1\,, \quad \text{for all } S \subseteq V \text{ with } \emptyset \neq S \neq V. \end{aligned} \tag{4.6}$$

Condition (4.6) says that every nontrivial cut must contain at least one edge, a characterization of connectivity (see exercise below). Note, however, that for the edge set of a tour, every such cut must indeed have at least two edges (follows from the fact that every cut must have even size). That is, we can substitute (4.6) by

$$\sum_{e\in\delta(S)} x_e \geq 2\,, \quad \text{for all } S \subseteq V \text{ with } \emptyset \neq S \neq V. \tag{4.7}$$

While these constraints are equivalent in the integer program, the resulting LP relaxation of the traveling salesman problem is more constrained and

[5] One source and inspiration for the text below is Jens Vygen, New approximation algorithms for the TSP, `http://www.or.uni-bonn.de/~vygen/files/optima.pdf`. Thanks also to Thomas Holenstein for discussions on the topic and to Manuel Wettstein for comments after reading a draft version.

[6] More recently, in the spirit of political correctness, renamed to traveling sales*person* problem.

therefore its optimal solution is hopefully closer to the integer solution.

Subtour LP for graph $G = (V, E), c \in \mathbf{R}^E$

$$\begin{array}{lrcll} \min & c^T x & & & \\ \text{subject to} & \sum_{e \in \delta(v)} x_e & = & 2\,, & \text{for all } v \in V, \\ & \sum_{e \in \delta(S)} x_e & \geq & 2\,, & \text{for all } S \subseteq V \text{ with } \emptyset \neq S \neq V, \text{ and} \\ & 1 \geq x_e & \geq & 0\,, & \text{for all } e \in E. \end{array}$$

This LP is called *Subtour LP* or *Held-Karp relaxation*, although it was first considered by Dantzig, Fulkerson and Johnson in 1954. Recall that an optimal solution $\tilde{x}$ to this LP is a lower bound for an optimal solution $x^* \in \{0,1\}^E$ to the underlying integer program, i.e.

$$c^T \tilde{x} \leq c^T x^* = \mathrm{OPT}_{\mathrm{tour}}\,.$$

with $\mathrm{OPT}_{\mathrm{tour}}$ the cost of the optimal tour in G with costs c. (Caveat: Is the LP and IP always feasible? See exercise below.)

A graph $G = (V, E)$ with costs c satisfies the *triangle inequality* if $E = \binom{V}{2}$ and

$$c_{\{u,w\}} \leq c_{\{u,v\}} + c_{\{v,w\}} \quad \text{for distinct } u, v, w \in V.$$

(For $|V| \geq 3$ it follows that $c_e \geq 0$ for all $e \in E$.) Provided the triangle inequality is satisfied, it can be shown that

$$c^T \tilde{x} \leq c^T x^* = \mathrm{OPT}_{\mathrm{tour}} \leq \frac{3}{2} c^T \tilde{x}\,.$$

The $3/2$ ratio reminds us of the Christofides approximation for the traveling salesman problem with triangle inequality. We will return to this point later and we will see that there is indeed a connection.

There is a worry looming: The Subtour LP has an exponential number of constraints ($n + (2^n - 2) + 2m$, for $n := |V|$ and $m := |E|$). So can we solve the LP efficiently? The dimension is m and given a vector x, we can find a violated cut constraint (4.7), if it exists, via a min-cut algorithm in polynomial time. This provides exactly the type of polynomial separation oracle we need for an efficient employment of the ellipsoid method.

Exercise 4.10. *Show that a graph $G = (V, E)$ is connected iff $\delta(S) \neq \emptyset$ for all $S \subseteq V$, $\emptyset \neq S \neq V$. Recall here the definition of "connected:" A graph G is* connected *if there is a path between any two vertices in G.*

Exercise 4.11. *(i) Give a graph* $G = (V, E)$ *for which the subtour LP is infeasible. (ii) Give a graph* $G = (V, E)$ *where the subtour LP is feasible but there is no feasible integer solution.*

Exercise 4.12. *Show that the constraints "*$1 \geq x_e$*" are redundant in the Subtour LP, i.e. every point* $x \in \mathbf{R}^E$ *that is feasible w.r.t. all other constraints does satisfy* $1 \geq x_e$ *for all* $e \in E$*.*

Exercise 4.13. *Show that all characteristic vectors* x *of tours of a graph* $G = (V, E)$ *are vertices of the feasible region of the Subtour LP. (You can use the following characterizations: (1) A point* $p \in \mathbf{R}^m$ *is a* vertex *of a convex polyhedron* $P \subseteq \mathbf{R}^m$ *if there exists a hyperplane* h *with* $P \cap h = \{p\}$*. (2) A point* $p \in \mathbf{R}^m$ *is a* vertex *of a convex polyhedron* $P \subseteq \mathbf{R}^m$ *if there exists a vector* $c \in \mathbf{R}^m$ *such that* p *is the unique point in* P *that maximizes* $c^T p$*.)*

Exercise 4.14. *Investigate the Subtour LP versus the Loose Subtour LP, where the cut constraint is written with "*≥ 1*" rather than "*≥ 2*". One concrete question to consider is the following: If* $\tilde{x}$ *is an optimal solution to the Subtour LP and* $\tilde{x}'$ *is an optimal solution to the Loose Subtour LP, how small can* $c^T\tilde{x}'$ *be compared to* $c^T\tilde{x}$*?*

4.8 Minimum Spanning Tree

Given a graph $G = (V, E)$ a subgraph $T = (V, E')$, $E' \subseteq E$, is called a *spanning tree of* G if (i) it is connected and (ii) it has no cycle. Given costs $c_e \in \mathbf{R}$ for $e \in E$, a minimum spanning tree is a spanning tree $T = (V, E')$ with minimal cost $\sum_{e \in E'} c_e$.

It is well-known that there are several equivalent characterizations, e.g.

T *is a tree iff (i) it has exactly* $n - 1$ *edges and (ii) it is connected.*

This suggests to express the characteristic vectors of spanning trees by the following constraints.

$$\begin{aligned} x &\in \{0,1\}^E \\ \sum_{e \in E} x_e &= n - 1\ , \text{ and} \\ \sum_{e \in \delta(S)} x_e &\geq 1\ , \text{ for all } S \subseteq V \text{ with } \emptyset \neq S \neq V. \end{aligned}$$

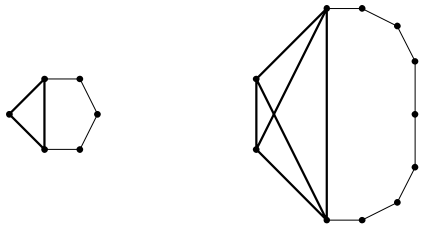


Figure 4.1: Graphs $G_{3,4}$ and $G_{4,8}$, "fractional trees" according to the Loose Spanning Tree LP: With weight $x_e := 1$ for fat edges e and weight $\frac{1}{2}$ for thin edges, the overall weight is the number of vertices minus 1 and all cuts have weight at least 1.

We consider the LP relaxation.

Loose Spanning Tree LP for graph $G = (V, E), c \in \mathbf{R}^E$

$$\begin{array}{lrcl} \min & c^T x & & \\ \text{subject to} & \sum_{e\in E} x_e & = & n-1 \\ & \sum_{e\in\delta(S)} x_e & \geq & 1\,, \text{ for all } S \subseteq V, \emptyset \neq S \neq V, \text{ and} \\ & 1 \geq x_e & \geq & 0\,, \text{ for all } e \in E. \end{array}$$

Let us investigate the behavior of the Loose Spanning Tree LP on the graph

$$G_{k,\ell} := \left(U \,\dot\cup\, \{v_0, v_1, \ldots v_\ell\}, \binom{U \cup \{v_0, v_\ell\}}{2} \cup \{\{v_0, v_1\}, \{v_1, v_2\}, \ldots \{v_{\ell-1}, v_\ell\}\} \right)$$

with $|U| = k-2$. That is, this graph has $k+\ell-1$ vertices; it consists of a k-clique with a path of length ℓ attached to two vertices of the clique. Now define $c_e := 0$ for all edges of the k-clique and, for some positive real number γ, $c_e := \gamma$ for all edges of the ℓ-path.

Lemma 4.10. *(i) For the graph $G_{k,\ell}$ with costs as described above a minimum spanning tree has cost $(\ell-1)\gamma$. (ii) The corresponding Loose Spanning Tree LP has a value of at most $\frac{\ell}{2}\gamma$ if $\ell = k(k-3)+4$.*

Proof. Every spanning tree of $G_{k,\ell}$ must contain at least $\ell-1$ of the edges of the ℓ-path, otherwise it cannot be connected. Hence, (i) holds.

For a proof of (ii) let $x_e := 1/2$ for the edges of the ℓ-path and $x_e := 1$ for the edges in the k-clique. This gives $\sum_{e\in E} x_e = \ell/2 + \binom{k}{2} = (k+\ell-1)-1$

for $\ell = k(k-3)+4$. Therefore the equality of the LP is satisfied, and it is easily seen that the cut constraints are met as well (follows from the fact that there is a spanning cycle where all edges have weight x_e at least $\frac{1}{2}$). The value of $c^T x$ equals $\frac{\ell}{2}\gamma$. □

We have seen that the Loose Spanning Tree LP allows a fractional solution that is roughly a factor $\frac{1}{2}$ smaller than the cost of a minimum spanning tree. A better LP is possible. For that we now switch to the alternative characterization

T is a tree iff (i) it has $n-1$ edges and (ii) it contains no cycle.

A graph has no cycle iff no set of k vertices induces a graph with k or more edges. This leads to the following new set of constraints for the spanning tree problem.

Tight Spanning Tree LP for graph $G = (V, E), c \in \mathbf{R}^E$

$$\begin{array}{lrcl} \min & c^T x & & \\ \text{subject to} & \sum_{e\in E} x_e & = & n-1 \\ & \sum_{e\in E\cap\binom{S}{2}} x_e & \leq & |S|-1\,, \quad \text{for all } S \subseteq V, \emptyset \neq S \neq V, \text{ and} \\ & 1 \geq x_e & \geq & 0\,, \quad \text{for all } e \in E. \end{array}$$

Here comes a surprise, although we have to leave it without a proof here and refer to the literature for that.

Theorem 4.11 (Edmonds, 1970). *Every basic feasible solution of the Tight Spanning Tree LP is integral. Therefore, the value of the Tight Spanning Tree LP equals the cost of the minimum spanning tree for every cost vector c.*

The Geometric View and the Spanning Tree Polytope. We are interested in a certain family of subsets of a ground set, here the family $\mathcal{T}$ of (edge sets of) spanning trees of a graph $G = (V, E)$. Let $S_{\mathcal{T}} \subseteq \{0,1\}^E$ be the set of characteristic vectors of the sets in $\mathcal{T}$. For $c \in \mathbf{R}^E$, we are interested in $\min_{x\in S_{\mathcal{T}}} c^T x$ (and in an $x^* \in S_{\mathcal{T}}$ with $c^T x^* = \min_{x\in S_{\mathcal{T}}} c^T x$).

We now design a linear program $Ax \leq b$ such that the polyhedron $\tilde{P} := \{x \in \mathbf{R}^E : Ax \leq b\}$ specifies our problem in the sense that $S_{\mathcal{T}} = \tilde{P} \cap \mathbf{Z}^E$. A solution $\min_{x\in\tilde{P}} c^T x$ to the linear program (with cost vector c) may obviously deviate from our desired $\min_{x\in S_{\mathcal{T}}} c^T x$. The better the polyhedron $\tilde{P}$ "embraces" $S_{\mathcal{T}}$, the better the LP solution will be.

There is a "perfect" polyhedron, namely $P^* = P^*_{\mathcal{T}} := \mathrm{conv}(S_{\mathcal{T}})$. We have $P^* \subseteq [0,1]^E$, therefore P^* is a polytope[7]; it is called the *spanning tree polytope* of the underlying graph G.

It is not hard to show that

$$\min_{x \in P^*} c^T x = \min_{x \in S_{\mathcal{T}}} c^T x \ .$$

A fundamental theorem in discrete geometry says that every polytope is the intersection of a finite number of halfspaces, that is, there is a linear program $A^* x \leq b^*$ such that P^* equals $\{x \in \mathbf{R}^E : A^* x \leq b^*\}$. Edmonds' Theorem tell us that the Tight Spanning Tree LP has exactly this property: The set of feasible solutions is P^*.

But we just learned that such an LP always exists, so what's the big deal. The problem is that for our algorithmic intentions (i.e. solving a optimization problem efficiently), the existence of the perfect LP is not enough, we need to get a hand on it, find some concrete description. This description needs not to be small, we do not even need it in some explicit form, but we need some form of separation oracle as we have discussed it for the ellipsoid method.

Exercise 4.15. *Show that the value of the Loose Spanning Tree LP for $G_{k,\ell}$ with the weights described above and with $\ell = k(k-3)+4$ is* exactly $\frac{\ell}{2}\gamma$.

Exercise 4.16. *Show that every feasible point of the Tight Spanning Tree LP is feasible in the Loose Spanning Tree LP – without using Theorem 4.11.*

Exercise 4.17. $m \in \mathbf{N}$. *Let $S \subseteq \mathbf{R}^m$ be finite and $P := \mathrm{conv}(S)$. Show that for every $c \in \mathbf{R}^m$ we have*

$$\min_{x \in P} c^T x = \min_{x \in S} c^T x \ .$$

(You may use the Separation Lemma: For $p \in \mathbf{R}^m$ and $C \subseteq \mathbf{R}^m$ a convex set, we have $p \notin C$ iff there exists a closed halfspace H with $p \in H$ and $H \cap C = \emptyset$.)

[7] Follows also from the fact that $S_{\mathcal{T}}$ is finite.

Exercise 4.18. *Consider the following linear program, almost the Tight Spanning Tree LP, it seems:*

Some LP *for graph* $G = (V, E), c \in \mathbf{R}^E$

$$\begin{array}{lrcl} \min & c^T x & & \\ \textit{subject to} & \sum_{e \in E} x_e & = & n \\ & \sum_{e \in E \cap \binom{S}{2}} x_e & \leq & |S| - 1\ , \quad \textit{for all } S \subseteq V, \emptyset \neq S \neq V, \textit{ and} \\ & 1 \geq x_e & \geq & 0\ , \quad \textit{for all } e \in E. \end{array}$$

What are the edge sets corresponding to vectors $x \in \{0,1\}^E$ *feasible in Some LP?*

4.9 Back to the Subtour LP

With the euphoria after having seen a "perfect" LP relaxation for the minimum spanning tree problem, we would of course like to know what the situation with the Subtour LP is. More concretely, for $\tilde{x}$ an optimal solution to the Subtour LP and for x^* the integral counterpart, do we always have equality $c^T\tilde{x} = c^T x^*$. We suspect this not to be true since otherwise we have a polynomial time algorithm for the traveling salesman problem, an NP-complete problem—but why not?.

If this equality does not always hold, we are still interested in how big the ratio $\frac{c^T x^*}{c^T \tilde{x}}$ can get. This ratio is called the *integrality ratio*[8] (Edmonds' Theorem 4.11 implies that the integrality ratio is always 1 for the Tight Spanning Tree LP).

Graphic Metrics. In order to get a hand on interesting cost functions, we make the following definitions. Let $G = (V, E)$ be a connected graph and let $d(u, v)$ be the length of the shortest path between u and v. This induces costs $c_{\{u,v\}} := d(u, v) \in \mathbf{N}$ on the edges of the complete graph $(V, \binom{V}{2})$. These costs always satisfy the triangle inequality. A cost function on the complete graph obtained in this way is called a *graphic metric*.[9]

Lemma 4.12. *Let the graphic metric* c *on* $\binom{V}{2}$ *be induced by the connected graph* $G = (V, E)$*. Then the optimal tour of* $G' = (V, \binom{V}{2})$ *with costs* c

[8] Sometimes also called *integrality gap*.

[9] The traveling salesman problem for a graphic metric is often called *graphic TSP* or *graph TSP*.

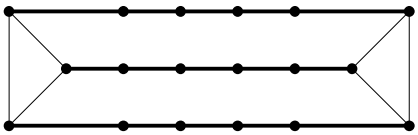


Figure 4.2: Graph G_6, a "fractional" cycle according to the Subtour LP: If fat edges e get weight $x_e := 1$ and thin edges get weight $x_e := \frac{1}{2}$, then every vertex is incident to edges of overall weight 2, and every cut has weight at least 2.

equals the length of the shortest closed walk[10] *in* G *visiting all vertices at least once.*

Proof. Let $(v_0, v_1, \ldots, v_{\ell-1}, v_\ell = v_0)$ be a closed walk of length ℓ in G visiting all vertices, i.e. there are indices $0 = i_1 < i_2 < \ldots i_n < \ell$ such that $V = \{v_{i_1}, v_{i_2}, \ldots, v_{i_n}\}$. Note that for $1 \leq j < k \leq \ell$, we have $d(v_{i_j}, v_{i_k}) \leq i_k - i_j$ and, therefore, the tour $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$ and back to v_{i_1} is a tour in G' of cost at most ℓ. In a similar fashion every tour of cost ℓ in G' can be turned into a closed walk of length ℓ visting all vertices of G. ◻

Now we consider a specific graph G_k with $n := 3k$ vertices. It consists of three disjoint paths of length $k-1$ (i.e. k vertices each), plus pairwise edges between the three starting points and also between the three end points of these paths. Let G'_k be the graph with the graphic metric induced by G_k.

Lemma 4.13. *Let* $k \in \mathbf{N}$ *and let* $n := 3k$ *(the number of vertices of* G_k *and* G'_k*). (i) The Subtour LP on* G'_k *has a feasible solution with value* n*. (ii) Every closed walk in* G_k *visiting all vertices has length at least* $\frac{4}{3}n - O(1)$ *and therefore every tour in* G'_k *has cost at least* $\frac{4}{3}n - O(1)$*.*

Proof. Note first that for every edge e of G_k we have $c_e = 1$ in G'_k. Now set $x_e := \frac{1}{2}$ for the six edges in the two 3-cliques of G_k, let $x_e := 1$ for the remaining $n - 3$ "path"-edges in G_k, and let $x_e := 0$ for all other edges in G'_k; hence, $\sum_e x_e = n$. The resulting vector x is feasible in the Subtour LP (not too difficult to check, argument omitted here) and $c^T x = \sum_e x_e = n$, since all edges with $x_e > 0$ have $c_e = 1$. Thus, (i) is shown.

For (ii) observe that every closed walk visiting all vertices in G_k has to use all or all but one edge on each of the three building paths of G_k:

[10] A *walk* in a graph $G = (V, E)$ is a sequence of (not necessarily distinct) vertices $(v_0, v_1, \ldots, v_\ell)$ with consecutive vertices adjacent in G. The walk is *closed* if $v_\ell = v_0$. The *length* of the walk is ℓ, and, for edge costs $c \in \mathbf{R}^E$, the *cost* of the walk is $\sum_{i=1}^{\ell} c_{\{v_{i-1}, v_i\}}$.

Skipping two edges disconnects some vertex from the rest and thus the walk cannot visit all vertices. Next consider a set of three edges $\{e_1, e_2, e_3\}$, with e_1 from the first path, e_2 from the second and e_3 from the third path. Removal of these three edges splits the graph into two parts. It follows that every closed walk has to use the edges in $\{e_1, e_2, e_3\}$ an even number of times; that is, if all three edges are indeed used at least once, then we have to use them at least 4 times altogether.

Now partition the edges of the three paths in triples, each triple containing one edge from each path. This gives $k-1$ such triples, where at least $(k-1)-3$ triples have to be used at least 4 times in a closed walk visiting all vertices. Therefore, the walk must have length at least $4(k-4) = \frac{4}{3}n - 16$. □

We can conclude that the Subtour LP is not perfect and that we have a family of graphs G'_k with costs c which exhibit an integrality ratio of roughly $\frac{4}{3}$:

$$\frac{c^T x^*}{c^T \tilde{x}} \to \frac{4}{3} \quad \text{as the size } n = 3k \text{ of the graphs grows.}$$

Now, clearly, we wish to know whether the integrality ratio $\frac{c^T x^*}{c^T \tilde{x}}$ has an upper bound. For that we will have to combine the insights we have collected about the Subtour LP, about the Tight Spanning Tree LP, and about the Christofides approximation algorithm for TSP.

Exercise 4.19. *Show that a shortest closed walk visiting all vertices in* G_k *has length* $4k - 2 = \frac{4}{3}n - 2$.

Exercise 4.20. *Let* $T = (V, E)$ *be a tree and let* G' *be the complete graph on* V *with the graphic metric induced by* T. *(i) What is the cost of an optimal tour in* G'*? (ii) What can you say about the value of the Subtour LP value for* G'*? (This is a very unspecified question, so you will have to decide for yourself in which direction to go.)*

4.10 Subtour LP versus Tight Spanning Tree LP

We now relate the Subtour LP to the Tight Spanning Tree LP with the goal of deriving an upper bound on the integrality ratio of the Subtour LP.

Lemma 4.14. *For a given graph $G = (V, E)$, if $x \in \mathbb{R}^E$ is a feasible solution of the Subtour LP, then $\frac{n-1}{n}x$ is a feasible solution of the Tight Spanning Tree LP.*

Proof. For x a feasible solution of the Subtour LP we have

$$\sum_{e \in E} x_e = \frac{1}{2} \sum_{v \in V} \sum_{e \in \delta(v)} x_e = \frac{1}{2} 2n = n$$

and therefore $\sum_{e \in E} \frac{n-1}{n} x_e = n - 1$ and the first constraint of the Tight Spanning Tree LP is satisfied for $\frac{n-1}{n}x$.

Next, for $S \subseteq V$, $\emptyset \neq S \neq V$, we have

$$\sum_{e \in E \cap \binom{S}{2}} x_e = \frac{1}{2} \left(\sum_{v \in S} \overbrace{\sum_{e \in \delta(v)} x_e}^{=2} - \overbrace{\sum_{e \in \delta(S)} x_e}^{\geq 2} \right) \leq \frac{1}{2}(2|S| - 2) = |S| - 1 \ .$$

We have shown that also the inequalities of the Tight Spanning Tree LP are satisfied. (Even $1 \geq \frac{n-1}{n} x_e \geq 0$ follows from $1 \geq x_e \geq 0$.) ◻

Corollary 4.15. *Given a graph G, if $\tilde{x}$ is an optimal solution of the Subtour LP and OPT_{mst} is the cost of the minimum spanning tree, then $c^T \tilde{x} \geq \frac{n}{n-1} OPT_{mst}$.*

Proof. We know that $\frac{n-1}{n}\tilde{x}$ is a feasible solution of the Tight Spanning Tree LP, and therefore $\frac{n-1}{n} c^T \tilde{x}$ is at least the value of the Tight Spanning Tree LP, which is attained by a basic feasible solution, which is known to be integral (by Theorem 4.11) and therefore the value equals OPT_{mst}. ◻

Recall that, under the assumption of the triangle inequality, we have

$$OPT_{tour} \leq 2 \cdot OPT_{mst} \tag{4.8}$$

(We can turn the closed walk that uses every edge of a minimum spanning tree twice into a tour by skipping vertices already visited; the triangle inequality ensures that the resulting tour has cost at most the cost of the initial closed walk, which is twice the cost of the minimum spanning tree.)

Corollary 4.16. *For a graph G whose costs c satisfy the triangle inequality the integrality ratio of the Subtour LP is at most 2.*

Proof. For $\tilde{x}$ an optimal solution to the Subtour LP and x^* an optimal integral solution, we have

$$c^T\tilde{x} \geq \frac{n}{n-1}OPT_{mst} \geq OPT_{mst} \geq \frac{1}{2}OPT_{tour} = \frac{1}{2}c^Tx^* ;$$

the first inequality is from Corollary 4.15, the second inequality uses that OPT_{mst} is non-negative (because of the triangle inequality), the third inequality is from (4.8) above, and the last equality is the basic property of the Subtour LP. Hence, $\frac{c^Tx^*}{c^T\tilde{x}} \leq 2$. □

This bound of 2 allows improvement along the familiar Christofides approximation, which generates a tour at most 3/2 times the cost of the optimal tour as follows: (1) Choose a minimum spanning tree (of cost OPT_{mst}). (2) For U the set of vertices of odd degree in this tree, compute the minimum weight matching (of cost $OPT_{match(U)}$) covering all vertices in U (this set is conveniently of even size). (3) The union of the minimum spanning tree and the matching is Eulerian (i.e. is connected and all vertices have even degree, if edges both in tree and matching are counted twice), thus a closed walk of cost $OPT_{mst} + OPT_{match(U)}$ visiting all vertices exists. (4) Such a walk can be turned into a tour of at most this cost (exploiting the triangle inequality).

We can can conclude that

$$OPT_{tour} \leq OPT_{mst} + OPT_{match(U)}$$

Besides $OPT_{mst} \leq c^T\tilde{x}$ one can also show $OPT_{match(U)} \leq \frac{1}{2}c^T\tilde{x}$ (proof omitted here). With these facts in our hands, we can conclude:

Theorem 4.17. *If G is a complete graph with costs satisfying the triangle inequality and if $\tilde{x}$ is an optimal solution to the Subtour LP for G, then*

$$c^T\tilde{x} \leq OPT_{tour} \leq \frac{3}{2}c^T\tilde{x} ,$$

(for OPT_{tour} the cost of an optimal tour in G).

For all $\varepsilon > 0$ there exists a graph with costs satisfying the triangle inequality such that $OPT_{tour} \geq (\frac{4}{3} - \varepsilon)c^T\tilde{x}$.

The situation between $\frac{4}{3}$ and $\frac{3}{2}$ has been open for a long time – and still is, to the best of my knowledge. Also, Christofides' polynomial 3/2-approximation is still the best known for triangle inequality TSP. However,

there was some progress on graphic TSP (with a graphic metric), where Mömke and Svensson, 2011, showed a 1.461-approximation, and later a 1.4-approximation by Sebő and Vygen, 2012.

For open problems in this context see Vygen's survey `http://www.or.uni-bonn.de/~vygen/files/optima.pdf`, Section 8 (page 22).