LINEAR PROGRAMMING

Karl Heinz Borgwardt

University of Augsburg, Institute for Mathematics, 86135 Augsburg, Universitaetsstrasse 14, Germany

Keywords:

Optimization, Linear Programming, Simplex Method, Ellipsoid Method, Interior Point Methods, Duality, Worst-Case Complexity, Average-Case Complexity, Polyhedra, Linear Inequalities.

Contents

Glossary

adjacent: Two vertices of a polyhedron are adjacent, if they are connected by an edge.
barrier function: A function tending to infinity at the boundary of the feasible region.
basic solution: A (feasible or infeasible) point where at least n restrictions are tight, whose gradients are linear independent.
basis-exchange: The replacement of one element of a basis by a nonbasic element.
capacity: The righthand-side value of a restriction.
central path: A set of points in a polyhedron, where certain measures of distance to the complete boundary are maximized. The measures depend on a parameter, the solutions for all parameters form the central path.
complementary slackness: The relation between a primal variable and a dual restriction or vice versa. In a weak version at least one of them is zero/tight, in a strong version exactly one

of them is zero/tight.
computation time: The time required by an algorithm to produce a solution, often measured in the number of arithmetical operations and of bits to be handled.
(convex) cone: A cone is a set of vectors such that each prolongated or reduced (positive) version of that vector is also in this set.
convexity: A set of points is called convex, if with each pair of points in that set the connecting line segment belongs to the set, too.
decision variable: One component of the vector describing the action which is to be carried out.
degenerate: A vertex is called degenerate, if more restrictions than necessary are tight there.
diagonal matrix: A matrix having nonzero entries only in the diagonal.
distance to centrality : A measure describing a distance to a specific point on the central path.
dual feasible: A basic solution is dual feasible, if it would be optimal in case that the violated constraints could be ignored.
duality: A relation between pairs of Linear Programs such that both can be solved simultaneously and that the sets of objective values bound each other and intersect only in optimal values.
edge: A boundary set of a polyhedron of dimension one.
ellipsoid: A set of points in $\mathbb{R}^n$ which is a generalized version of an ellipse from $\mathbb{R}^2$.
extremal subset or face: A subset of a polyhedron such that exactly there a linear functional becomes maximal.
facet: A face of a polyhedron of maximal dimension not coinciding with the polyhedron itself.
feasible set: The set of points in an LP satisfying all restrictions.
halfspace: A set of points satisfying a linear inequality of the type "$\leq$".
hyperplane: A set of points satisfying a linear equation (the boundary of a halfspace).
infeasibility: The situation that the restrictions of an LP cannot be satisfied altogether, i.e. that no point is feasible.
kernel: The solution set of a linear system $Ax = 0$ is called the kernel of the matrix A.
Newton Method : A numerical method for approximating the minimal points of functions or the zero-points of a function.
Linear Program (LP): A mathematical task to find the maximal or minimal value of a linear function at a point where certain restrictions in form of linear inequalities or linear equations are satisfied.
objective function: The function that shall be optimized, resp. the criterion for quality.
parametric optimization: Determination of all optimal points for arbitrary mixtures of two different objectives.
Phase I: A calculation procedure for determining a feasible point or a vertex of an LP.
Phase II: A calculation procedure for improving the objective until the optimal point is reached.
pivot step: The arithmetical handling of a basis-exchange.
polyhedron: A set of points satisfying a finite system of linear inequalities.
polynomiality: An algorithm is called polynomial if its calculation time is bounded by a polynomial in the encoding length of the problem-instance.
polytope: A bounded polyhedron.
predictor-corrector: An iterative algorithm which first extrapolates in order to improve the objective, and - after that - tries to get back into the neighbourhood of the central path.
reduced cost coefficients: Values indicating whether increasing a variable is worth while.

Revised Simplex Method: A version of the Simplex Method, designed for saving computation time, in particular suited for very large problems.
restriction: An equation or inequality that has to be satisfied by all decision vectors which may be considered.
rounding procedure: The determination of a close-by vertex from an approximating iteration point.
search direction: Instead of searching better points in the whole space, one concentrates on a certain line and determines the best point on that line. That line induces the search direction.
sensitivity analysis: A study how the optimal value and the optimal point change, if some data of the problem vary.
Simplex Path: A sequence of successively adjacent vertices such that the objective values improve.
sparse matrix: A matrix that has very few nonzero entries.
steepest descent: The direction where a function decreases the most.
strong polynomiality: An algorithm whose computation time is polynomial in the number of entries of the matrix of the problem, is called strongly polynomial.
tight/loose restriction: A $\leq$ restriction is called tight at a point x, if also the corresponding $=$ is satisfied, else it is called loose at x.
transformation: A modification of the mathematical description.
transformation-equivalent: A property of a pair of Linear Programs whose difference results from the application of feasible modifications.
unboundedness: Appears in connection with the feasible region and in connection with the objective. If the latter is unbounded, the problem does not contain an optimal point.
vertex: An extremal set of dimension 0 in a polyhedron.
vertex-exchange: The move from one vertex to another one in the process of the Simplex Method.

Summary

This survey has the purpose of giving an impression over the various fields of applications of Linear Programming in real-world decision making, of clarifying the essential mathematical terms and principal theoretical ideas in that mathematical field, and of explaining the most useful and efficient solution methods.
Linear Programming is a mathematical principle to simulate the well known real-world situation, that one achieves a certain goal, but has to consider certain constraints while searching for the best possible decision. In Linear Programming one assumes that the decisions have an impact on the objective and on the restrictions, that can be characterized by linear functions of the variables characterizing the decisions. And then it is a task of mathematics to calculate the optimal values of the decision variables. We present the mathematical theory of linear inequalities, of polyhedra and of duality. And then we show how this can be exploited to develop algorithms for solving such problems technically. The three methods in discussion are the Simplex Method, the Ellipsoid Method and Interior Point Methods. Their description finally leads to a comparison about their efficiency and complexity.

1 Linear Programming Problems

1.1 Formulation of Linear Programming Problems

In real life most of our decisions how to act or how to behave can be interpreted as attempts to optimize a certain goal or objective without violating certain restrictions, which may be given by nature, men or our own will, to follow existing rules.
For mathematics this means a challenge to modelize and to formalize this attempt mathematically, and to provide calculation methods for the determination of the best possible decision, if the objective and the restrictions are known. This mathematical field of developing tools for that purpose is of enormous importance in economy, engineering, administration, communication, and in all questions concerning technological development.

The mathematical approach for solving such problems is as follows:
Translate the possible decision set into a formal set of vectors of a finite number of decisions variables. Then find out which of these decision–vectors are feasible under the given restrictions. After that optimize, i.e. select the best decision-vector among the feasible ones. The criterion for good, better and best comes from a mathematical function describing the specific quality of the decision in question.

So a formal characterization of the mathematical problem is:

$$\begin{array}{ll} \text{maximize} & f(x), \quad \text{a function } f : \mathbb{R}^n \to \mathbb{R} \text{ defined on } x = (x^1, \ldots, x^n)^T \\ \text{subject to} & g_1(x) \leq \gamma_1, \ldots, g_m(x) \leq \gamma_m, \quad \text{and } h_1(x) = \kappa_1, \ldots, h_l(x) = \kappa_l. \end{array} \tag{1}$$

Here f is the objective function and the $g_i(x) \leq \gamma_i$ resp. $h_j(x) = \kappa_j$ are the $(m+l)$ restrictions. The g_i's and the h_j's are called the restriction functions. The values γ_i and κ_j are called capacities. The components of the vector x are called the decision variables.
The set of feasible vectors x will be denoted by X. So we can formalize our problem to

$$\begin{array}{ll} \text{Find a specific vector} & \overline{x} = (\overline{x}^1, \ldots \overline{x}^n)^T \in X \\ \text{such that} & f(\overline{x}) \geq f(x) \text{ for all } x \in X \end{array} \tag{2}$$

with $X = \{x \,|\, g_1(x) \leq \gamma_1, \ldots, g_m(x) \leq \gamma_m \text{ and } h_1(x) = \kappa_1, \ldots, h_l(x) = \kappa_l\}$.

Depending on the mathematical features of those functions, these problems are classified into certain categories.
What we have mentioned in (**??**) is a typical general Nonlinear Programming Problem (see Nonlinear Programming).
This survey is dedicated to a special form of that, namely to Linear Programming Problems or Linear Programs (LP).
Their general formulation is:

$$\begin{array}{rcll} \text{maximize} & c^T x & & \\ \text{subject to} & a_1^T x \leq b^1, \ldots, a_m^T x \leq b^m & \text{resp. } Ax \leq b, & \\ \text{and} & d_1^T x = p^1, \ldots, d_k^T x \leq p^k & \text{resp. } Dx = p, & \\ \text{where} & x, c, a_1, \ldots, a_m, d_1, \ldots, d_k \in \mathbb{R}^n,\ b \in \mathbb{R}^m,\ p \in \mathbb{R}^k. & & \end{array} \tag{3}$$

Here the vectors ${a_i}^T$ are supposed to be the row vectors of the matrix $A \in \mathbb{R}^{(m,n)}$ and the vectors ${d_i}^T$ form the matrix D.

Instead of the general functions $f(x)$, $g_i(x)$ and $h_j(x)$ we have now linear functions $c^T x$, $a_i^T x$ and $d_j^T x$. c, a_i, d_j are here the gradients of f, g_i, h_j.

An important property of these problems lies in the fact that the variables may attain all real values and that they may vary continuously. And the contribution of one variable to the objective function or to the restriction functions is proportional to its value in this specific type.

This is different in another type, which will repeatedly be mentioned in this text as a closely related, but discretely structured type, namely the Integer Linear Optimization Problem (see Combinatorial Optimization and Integer Programming), which is

$$\begin{array}{rcll} \text{maximize} & c^T x & & \\ \text{subject to} & a_1^T x \leq b^1, \ldots, a_m^T x \leq b^m & \text{resp. } Ax \leq b, & \\ \text{and} & d_1^T x = p^1, \ldots, d_k^T x \leq p^k & \text{resp. } Dx = p, & \\ \text{and} & x \in \mathbf{Z}^n & (\text{all } x^i\text{'s integer}) & \\ \text{where} & x, c, a_1, \ldots, a_m, d_1, \ldots, d_k \in \mathbb{R}^n,\ b \in \mathbb{R}^m,\ p \in \mathbb{R}^k. & & \end{array} \tag{4}$$

1.2 Examples

The variety of problems of that LP-type is overwhelming. Here we can only try to give a raw impression on the different application fields, where linear optimization problems are to be solved.

1. **Ingredient–Mixing**
   Assume that a product is composable out of a collection of different ingredients $I_1, \ldots, I_n$, that the costs of these ingredients are different, and that the quality can be controlled by varying the weights of the n ingredients. Now a certain level of quality shall be assured. This can be expressed in certain restrictions about the feasible mixtures. Then it is our aim to minimize the costs by choosing that feasible mixture, which has cheapest costs.

2. **Profit-Maximization in Investment**
   Assume that certain investment facilities are available for a given amount of money. How shall we distribute our money in order to maximize our profit when the investments are bounded, a certain level of risk shall not be exceeded and a certain diversification is required.

3. **Maximizing the delivered amount**
   Assume that several pipelines or cables are available to send goods or liquids or messages or data from a sender to a receiver continuously. Which pipelines or cables and to what extent shall be used in order to maximize the amount that arrives at the receiver's side per time unit?

4. **Production planning**
   Assume that the management of a company has to decide about the quantities in which certain products shall be manufactured. Then it is desirable to maximize the resulting profit without exceeding the capacity of available machine-time, personal labour force, financial credit etc.

5. **Transportation**
   Assume that m stores with certain stockpiles have to deliver goods to k shops, which have specific demands. And suppose that a delivery from store i to shop j causes costs of c_{ij} per unit. How shall the transport be organized, resp. how much of the goods shall be delivered from store i to shop j in order to minimize the costs and to satisfy the demands of all the shops.

These are typical linear optimization problems. But linear optimization techniques do not only help for these pure applications of Linear Programming. They also serve as extremely helpful tools as subroutines in the calculation of Integer Programming Problems as for example Staff Scheduling in Airline-Flights, Flight or Travel Scheduling, Route-Planning, Packing Problems, Location Problems etc. Here one uses Linear Programming repeatedly as a subroutine on subproblems where the integrity-condition is ignored. A systematic exploitation of the insights achieved in that way leads to the integer optimum (see Combionatorial Optimization and Integer Programming, Scheduling Problems, Routing Problems, Graph and Network Optimization).

Now we present a typical numerically specified problem-example.

$$
\begin{array}{lrcrcrcr}
\text{maximize} & & & 3x^2 & & & & \\
\text{subject to} & -1.03x^1 & + & 0.12x^2 & + & 0.06x^3 & \leq & 0.32 \\
& 0.05x^1 & - & 1.06x^2 & + & 0.06x^3 & \leq & 0.52 \\
& 0.011x^1 & + & 0.03x^2 & - & 1.09x^3 & \leq & 0.47 \\
& 1.14x^1 & + & 0.15x^2 & + & 0.11x^3 & \leq & 0.722 \\
& 0.2x^1 & + & 1.14x^2 & + & 0.075x^3 & \leq & 0.672 \\
& 0.14x^1 & + & 0.17x^2 & + & 1.01x^3 & \leq & 0.532 \\
& -0.22x^1 & - & 0.56x^2 & + & 0.67x^3 & \leq & 0.25 \\
& 1.1x^1 & - & 0.3x^2 & - & 1.31x^3 & \leq & 0.80 \\
& -1.1x^1 & + & 0.9x^2 & + & 0.9x^3 & \leq & 0.85
\end{array}
\tag{5}
$$

This problem can equivalently be written as

$$
\text{maximize } c^T x \quad \text{subject to } Ax \leq b, \quad \text{where} \tag{6}
$$

$$c = \begin{pmatrix} 0 \\ 3 \\ 0 \end{pmatrix} \quad \text{and} \quad \text{A} = \begin{pmatrix} -1.03 & 0.12 & 0.06 \\ 0.05 & -1.06 & 0.06 \\ 0.011 & 0.03 & -1.09 \\ 1.14 & 0.15 & 0.11 \\ 0.2 & 1.14 & 0.075 \\ 0.14 & 0.17 & 1.01 \\ -0.22 & -0.56 & 0.67 \\ 1.1 & -0.3 & -1.31 \\ -1.1 & 0.9 & 0.90 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 0.32 \\ 0.52 \\ 0.47 \\ 0.722 \\ 0.672 \\ 0.532 \\ 0.25 \\ 0.80 \\ 0.85 \end{pmatrix}.$$

Figure 1: Feasible Region

Figure 1 shows the feasible region for the example given above. The arrow shows the direction of the the second component x^2, our objective direction. The optimal point is the rightmost vertex $= (-0.258, 0.662, -0.416)^T$ and the optimal value is 1.986.

Geometrically, such a Linear Optimization Problem can be seen as follows:
Find a point $(x^1, \ldots, x^n)^T \in \mathbb{R}^n$, which maximizes the scalar product $c^T x$ on the feasible region (polyhedron) $X = \{x \mid Ax \leq b\}$. This means that among the isoclines of $c^T x$ that one shall be selected, which touches X without intersecting its interior. And the "touching point" is the optimal point we have been searching for.

From this view it becomes clear that there are four qualitatively different outcomes of an LP:

1. The feasible region X is bounded and the linear (continuous) function $c^T x$ attains its optimal value on X.
2. The feasible region X is unbounded, but the linear function $c^T x$ attains its optimal value on X anyway.
3. The feasible region X is unbounded, and the linear function $c^T x$ has no optimal value on X, because it is unbounded from above on X itself.
4. The constraints of $Ax \leq b$ are contradictory, which induces that $X = \emptyset$.

In algorithms this listing leads to the following general advice:
First, check whether X has feasible points. If not, then STOP because of INFEASIBILITY.
Else try to optimize. As soon as it is clear that the objective function is unbounded, then STOP also, but this time because of UNBOUNDEDNESS.
Else proceed with the optimization process until an optimal point is found. Then STOP because of OPTIMALITY.

1.3 Different Forms of Programs and Transformations

So far, we have presented only linear programs of the type

maximize $c^T x$ subject to $Ax \leq b$.

But all analytical and arithmetical insights about Linear Programs can easily be transfered to the "General Linear Programming Problem":

$$\begin{array}{lrcrcrcl} \text{maximize} & d^T x & + & e^T y & + & f^T z & & \\ \text{subject to} & Ax & + & By & + & Cz & \leq & a \\ & Dx & + & Fy & + & Gz & = & b \\ & Hx & + & Iy & + & Jz & \geq & c \\ & x & & & & & \geq & 0 \\ & & & & & z & \leq & 0 \end{array} \tag{7}$$

From the essence of a real problem it does not matter in which structural way some restrictions have been modelized. So we allow the following transformations and we regard their outcome as equivalent:

1. $a^T x = \beta \Longleftrightarrow a^T x \leq \beta$ AND $a^T x \geq \beta$
   an equation can be replaced by two inequalities.
2. $a^T x \geq \beta \Longleftrightarrow -a^T x \leq -\beta$
   an inequality can be replaced by the reverse negative inequality.
3. $a^T x \leq \beta \Longleftrightarrow a^T x + y = \beta, y \geq 0$
   an inequality can be replaced by an equation with a nonnegative slack variable.
4. $x = x_+ - x_-$ with ${x^i}_+ = Max\{x^i, 0\}$ and ${x^i}_- = Max\{-x^i, 0\}$
   a variable can be partitioned as a difference of two positive (minimal) parts.
5. $A_{\cdot i} x^i$ with $x^i \geq 0$ and $-A_{\cdot i} z^i$ with $z^i \leq 0$
   the contribution of a column and a variable can be represented by the negative column and the negative variable.

And, of course, it is allowed to replace a maximization problem by a minimization problem of the negative objective function (if we keep in mind that our result is negative then).

For the treatment of the different programs it is very important to have the following reduction opportunity: In every equivalence class according to the above mentioned transformations there is a problem as:

$$\text{maximize } c^T x \quad \text{s.t. } Ax \leq b \qquad \text{in canonical form,} \tag{8}$$

and as:

$$\text{minimize } c^T x \text{ s.t. } Ax = b,\ x \geq 0 \qquad \text{in standard form.} \tag{9}$$

2 Primal and Dual Programs and Polyhedra

2.1 Duality

Often it is helpful to get the information that a given point is optimal or that it is not optimal and how far it is away from optimality.

Such auxiliary knowledge can be derived from the theory of dual Linear Programs.

We allow the transformations mentioned above. This permission defines equivalence classes among linear programs. Then we define duality between two such classes, if in each class there is a representative, such that these two stand in the following relation.

Definition 1
The following two programs are (directly) dual to each other.

$$\begin{array}{lrcrcrcl} max & d^Tx & + & e^Ty & + & f^Tz & & \\ s.\ t. & Ax & + & By & + & Cz & \leq & a \\ & Dx & + & Fy & + & Gz & = & b \\ & Hx & + & Iy & + & Jz & \geq & c \\ & x & & & & & \geq & 0 \\ & & & & & z & \leq & 0 \end{array} \qquad \begin{array}{lrcrcrcl} min & a^Tu & + & b^Tv & + & c^Tw & & \\ s.\ t. & A^Tu & + & D^Tv & + & H^Tw & \geq & d \\ & B^Tu & + & F^Tv & + & I^Tw & = & e \\ & C^Tu & + & G^Tv & + & J^Tw & \leq & f \\ & u & & & & & \geq & 0 \\ & & & & & w & \leq & 0 \end{array} \tag{10}$$

If P is a program which is transformation-eqivalent to the left form and D is a program which is transformation-equivalent to the right form, then P and D are called dual to each other.

Here are two prominent dual pairs of programs.

$$P := \begin{array}{lrcl} \max & d^Tx & & \\ \text{s.t.} & Ax & \leq & a \\ & x & \geq & 0 \end{array} \qquad \text{and} \qquad D := \begin{array}{lrcl} \min & a^Tu & & \\ \text{s.t.} & A^Tu & \geq & d \\ & u & \geq & 0 \end{array} \tag{11}$$

are dual to each other as well as

$$P := \begin{array}{lrcl} \max & e^Ty & & \\ \text{s.t.} & By & \leq & a \end{array} \qquad \text{and} \qquad D := \begin{array}{lrcl} \min & a^Tu & & \\ \text{s.t.} & B^Tu & = & c \\ & u & \geq & 0 \end{array} \tag{12}$$

canonical form **standard form**

Remark 1
A dual program to a dual program of P is transformation-equivalent to P.

The tremendous value of such dual pairs comes from the following facts.

Theorem 1 *(Weak Duality)*
For all x satisfying $Ax \leq b$ and for all u satisfying $A^T u = c$, $u \geq 0$, we have $c^T x \leq b^T u$. Hence the objective values of a program $P := \max c^T x$ s.t. $Ax \leq b$ cannot be higher than those of its dual $D := \min b^T u$ s.t. $A^T u = c$, $u \geq 0$.

So any feasible point for D gives us an upper bound for all objective values in P.
And any feasible point for P gives us a lower bound for all objective values in D.

And this helps algorithmically. If we know a point $\bar{x}$ for P and a point $\bar{u}$ for D such that $d^T \bar{x} = a^T \bar{u}$, then both points must be optimal. Else, the difference of the values overestimates the distance to the respective optimal values. This will be formalized in the following theorem (which is given here only in its special form for canonical and standard programs).

Theorem 2 *(Strong Duality)*
For P (in canonical form) and D (in standard form) only four configurations are possible.

1. *P and D have feasible points. Then it is sure that both possess optimal points and that the optimal values are equal.*
2. *P is infeasible, but D is feasible. Then D has no optimal point, its objective is unbounded from below.*
3. *D is infeasible, but P is feasible. Then P has no optimal point, its objective is unbounded from above.*
4. *P and D are both infeasible.*

But we can profit even more, because duality will even help to identify the optimal points. This comes from the two following theorems.

Theorem 3 *(Weak Complementary Slackness)*
With P and D as above, $\tilde{x}$ and $\tilde{u}$ feasible for P resp. D, we know that the following statements are equivalent.

a) $\tilde{x}$ *is optimal for P and* $\tilde{u}$ *is optimal for D.*

b) $\tilde{u}^T(b - A\tilde{x}) = 0$.

c) For all components $\tilde{u}^i > 0$ *we know that* ${a_i}^T \tilde{x} = b$.

d) For all rows ${a_i}^T$ *of A with* ${a_i}^T \tilde{x} < b^i$ *we know that* $\tilde{u}^i = 0$.

So far, we observe that every pair of optimal points for P and D fulfils such a condition of complementary slackness as in b). But we know even more if we content ourselves with the existence of one pair of optimal points in the

Theorem 4 *(Strong Complementary Slackness)*
If P and D both have feasible points, then there exist optimal points $\tilde{x}$ and $\tilde{u}$ such that

$$\tilde{u}^i > 0 \Longleftrightarrow {a_i}^T \tilde{x} = b^i \quad \text{and} \quad {a_i}^T \tilde{x} < b^i \Longleftrightarrow \tilde{u}^i = 0.$$

We mention that these conditions enable us to conduct a kind of sensitivity analysis for the solution of LPs for the case that critical capacities change slightly (see Sensitivity Analysis).

2.2 Linear Inequalities and Polyhedra

The main tool in the treatment of linear inequalities is the famous

Lemma 1 *of Farkas*
Either there is a vector x *satisfying* $Ax = b$
or there exists a vector z *such that* $A^T z \leq 0,\ b^T z > 0$.

The consequence for the solution of LPs is given in

Theorem 5
Consider a point $\bar{x}$*, where* ${a_i}^T \bar{x} = b_i$ *for all* $i \in I$ *and* ${a_j}^T \bar{x} < b^j$ *for all* $j \notin I$*. Such an* $\bar{x}$ *is optimal if and only if there are nonnegative multipliers* $\rho_i \geq 0$ *for all* $i \in I$ *such that*

$$c = \sum_{i \in I} \rho_i a_i.$$

Now the task of algorithms is to identify points with that property, which can also serve as a guarantee for optimality.

Definition 2
A set $P \subset \mathbb{R}^n$ *is called a polyhedron, if there is a matrix* A *and a capacity-vector* b *such that* $P = \{x \,|\, Ax \leq b\}$*. It is called a polytope, if it is bounded.*

So geometrically, the feasible set of our m restrictions ${a_1}^T x \leq b^1, \ldots {a_m}^T x \leq b^m$
defines a polyhedron.
We observe that such a polyhedron is just the intersection set of a finite number (m) of halfspaces of the type $\{x \,|\, {a_i}^T x \leq b^i\}$, each of them bounded by the corresponding hyperplane $\{x \,|\, {a_i}^T x = b^i\}$.

A significant feature of such polyhedra is their convexity.

Definition 3
A set $M \subset \mathbb{R}^n$ *is called convex, if* $x_1 \in M$ *,* $x_2 \in M$ *and* $\lambda \in [0, 1]$ *implies that every convex combination* $y = \lambda x_1 + (1 - \lambda) x_2$ *is an element of* M*.*

And in such polyhedra our interest is mainly concentrated on the so-called extremal subsets.

Definition 4
A convex subset W *of a polyhedron is extremal, if for all* $x \in W$ *the following holds:*
for each pair $(y, z) \in M \times M$ *and for each* $\lambda \in [0, 1]$*, such that* $x = \lambda y + (1 - \lambda) z \ \in W$*, it is sure that* $y \in W$ *and* $z \in W$*.*
These extremal subsets of a polyhedron P *are called the faces of* P*.*

Remark 2
The faces of dimension 0 *are called vertices, faces of dimension 1 are edges, in dimension 2 we have planes and in general faces of dimension* $(Dim(P) - 1)$ *are called facets of* P*.*

Each of these faces in an LP-polyhedron is completely characterizable as the intersection set of P with all restriction hyperplanes bearing W completely.
In particular, each vertex is the intersection point of at least n restriction hyperplanes, each edge results from intersecting at least $n - 1$ such hyperplanes.

Another property of such faces is the following:
If a linear function attains its maximal value on a polyhedron P, then the optimal set is certainly one of the faces of P.
It is worthy to know that in any case, where P contains at least one vertex, then every face contains a vertex. And then, if there are optimal points, resp. if there is an optimal face, there is automatically an optimal vertex on that face.

This idea helps to reduce the linear optimizimation problem on polyhedra to a seemingly simpler problem, namely finding the best vertex and deciding whether it is already optimal in P or whether not (which would mean that there is no optimal point at all).
In particular this simplifies the handling of polytopes (bounded polyhedra), because there the possibility of unboundedness resp. nonexistence of optima can be ignored. Here the optimal vertex is directly the optimal point. The most famous and most efficient algorithm for solving LPs, the Simplex Method, is based on that idea.

3 The Simplex Method

In this section we will present two versions of the Simplex Method, an algorithm for solving LPs. Both versions have the following principles in common.

1. In a Phase I, a vertex of the feasible region X shall be calculated. This Phase has two possible outcomes:

   the infeasibility of X is proven and therefore we can STOP.

   a vertex x_0 of X is found, which is a basis for proceeding to Phase II.

2. In a Phase II, we start from the vertex x_0 and construct a sequence of vertices $x_0, \ldots, x_s$ such that successive vertices are adjacent (i.e. they are connected by an edge), and that the objective function is improving from vertex to vertex. The construction ends at a vertex x_s if at this vertex

   the unboundedness of the objective on X becomes obvious or if

   x_s is the optimal vertex for the objective on X.

Our first version is an algorithm which works on feasible regions as $X = \{x \mid Ax \leq b\}$. This version has the advantage of a very illustrative geometric explanation, since X can be regarded as a fulldimensional polyhedron, and it is easy to imagine a Simplex Path on such a figure.

The name of this version is "Restriction-oriented Simplex Method".

After that we shall talk about a second version, called the "Variable-oriented Simplex Method". This is specifically adapted to solve problems with $X = \{x \,|\, Ax = b, x \geq 0\}$. Its advantage lies in the fact that its solving technique is a bit closer to the Numerical Linear Algebra of solving systems of equations and therefore this version is more common in commercial software packages (see Numerical Linear Algebra).

But it should be mentioned clearly, that both are essentially equivalent, in a certain sense they are "dual" to each other.

3.1 The Restriction-oriented Simplex Method

Here we deal with problems

$$\text{maximize } c^T x \quad \text{s.t.} \quad Ax \leq b, \quad c, x \in \mathbb{R}^n,\ A \in \mathbb{R}^{(m,n)},\ b \in \mathbb{R}^m,\ m \geq n. \tag{13}$$

Since both Phases of the method run in a similar way, it makes sense to concentrate first on the more typical Phase II.
Suppose that we know a vertex $\bar{x}$ of X, which had been determined in Phase II.
In $\bar{x}$, we recognize that at least n of the restrictions ${a_i}^T x \leq b^i$ are tight, i.e. these are satisfied as equations $({a_i}^T \bar{x} = b^i)$. And the collection of those vectors a_i must contain a basis of $\mathbb{R}^n$.
Let I be the set of indices out of $\{1, \ldots, m\}$ corresponding to the tight restrictions and let $\Delta = \{\Delta^1, \ldots, \Delta^n\}$ be an n-element subset of I, such that $\{a_{\Delta^1}, \ldots, a_{\Delta^n}\}$ forms a basis of $\mathbb{R}^n$.
Then $\bar{x}$ is the unique solution point of the system of equations

$$a_{\Delta^1}^T x = b^{\Delta^1}, \ldots, a_{\Delta^n}^T x = b^{\Delta^n}. \tag{14}$$

Let A_Δ, A_I be the submatrices of A consisting of the respective rows. Then the following geometric insights are essential.

Lemma 2

1. *X is contained in the set described as $\bar{x} + cone(A_\Delta^{-1}(-e_1), \ldots, A_\Delta^{-1}(-e_n)) = \bar{x} + cone(z_1, \ldots, z_n)$.*
   Here the e_i are the unit vectors in $\mathbb{R}^n$, and the $z_i = A_\Delta^{-1}(-e_i)$ are the n unbounded edges of $\{z \,|\, A_\Delta z \leq 0\}$, the so-called recession cone of X at $\bar{x}$.
2. *For each $i \in \{1, \ldots, n\}$ resp. for each z_i, there is a value $= 0 \leq \delta_i \leq \infty$, such that exactly for $0 \leq \rho \leq \delta_i$ the points $\bar{x} + \rho_i z_i$ belong to X. And if $I = \Delta$, then all values δ_i are positive.*

Following these directions e_i gives us a chance to get to an adjacent vertex, which may improve the objective. This is formulated in the following

Lemma 3
If an interval $[\bar{x}, \bar{x} + \delta_i z_i]$ coincides with an edge of X for a value $\delta_i < \infty$, then it holds that

1. *$\tilde{x} := \bar{x} + \delta_i z_i$ is a vertex of X.*
2. *There is a $\tilde{j} \in \{1, \ldots, m\} \setminus \Delta$ such that*

   $$a_{\tilde{j}}{}^T(\bar{x} + \delta_i z_i) = b^{\tilde{j}} \quad \text{and} \quad a_{\tilde{j}}{}^T(z_i) > 0.$$

3. *Replacing i by $\tilde{j}$ produces a new basis $(a_{\Delta^1}, \ldots, a_{\Delta^{i-1}}, a_{\tilde{j}}, a_{\Delta^{i+1}}, \ldots, a_{\Delta^n})$.*

On each edge originating from x there are three possible configurations for δ_i

1. $\delta_i = 0$, i.e. $\bar{x} = \tilde{x}$
2. $0 < \delta_i < \infty$, i.e. $[\bar{x}, \tilde{x}]$ is a bounded edge of X.
3. $\delta_i = \infty$, i.e. $\bar{x} + \mathbb{R}^+ z_i$ induces an unbounded edge of X.

It depends on the objective $c^T x$, whether it will pay to move along an edge in direction z_i, because of

$$c^T z_i = c^T(A_\Delta^{-1}(e_i)) = (-e_i)^T (A_\Delta^{-1})^T c = -\xi^i. \tag{15}$$

Then we know that

1. If $\xi^i > 0$ then $\tilde{x}$ is worse than $\bar{x}$.
2. If $\xi^i < 0$ then $\tilde{x}$ is better than $\bar{x}$.
3. If $\xi^i = 0$ then $\tilde{x}$ is as good as $\bar{x}$.

The sequence of vertex-exchanges comes to an end as soon as one of the following two cases occurs:

1. $-\xi = c^T z_i \leq 0$ for all $i \in \Delta$, then $\bar{x}$ is optimal.
2. There is a z_i such that $Az_i \leq 0$ and $-\xi^i = c^T z_i > 0$,
   then the objective is unbounded from above on X.

So, our considerations lead to the following algorithm:

Algorithm 1
Initialization:
Let a vertex $x_\Delta = \bar{x}$, the set Δ and A_Δ^{-1} be given.
Typical iteration:

1. *Calculate $\xi = A_\Delta^{-1^T} c$.*
   If $\xi \geq 0$ then STOP because of OPTIMALITY.
   Else choose an index $\hat{i}$ such that $\xi^{\hat{i}} < 0$.

2. *Calculate $z_{\hat{i}} = A_\Delta^{-1}(-e_{\hat{i}})$ and $a_j^T z_{\hat{i}}$ for all $j \notin \Delta$.*
   If $a_j^T z_{\hat{i}} \leq 0$ for all $j \notin \Delta$, then STOP because $c^T x$ is UNBOUNDED. Else proceed.

3. *Determine a value $\delta_{\hat{i}}$ and a restriction which becomes tight only in $\bar{x} + \delta_{\hat{i}} z_{\hat{i}}$ and its corresponding index $\tilde{j}$ via the calculation*

$$\delta_{\hat{i}} := \delta_{\hat{i}}^{\tilde{j}} = Min \left\{ \frac{b^j - a_j^T \bar{x}}{a_j^T z_{\hat{i}}} \mid j \notin \Delta, a_j^T z_{\hat{i}} > 0 \right\}. \tag{16}$$

4. *Replace $\bar{x}$ by $\tilde{x} = \bar{x} + \delta_{\hat{i}} z_{\hat{i}}$ and Δ by $\Delta^* = \Delta \setminus \{\hat{i}\} \cup \{\tilde{j}\}$ and $c^T \bar{x}$ by $c^T \tilde{x}$.*

5. *Calculate $A_{\Delta^*}^{-1}$ i.e. update A_Δ^{-1}.*

6. *Set $\Delta := \Delta^*$ and go to 1.*

Figure 2: Simplex Path

Figure 2 shows a Simplex-Path in thick lines leading from the leftmost start vertex to the rightmost optimal vertex while improving the objective.

Remark 3
If I has more elements than Δ, then $\bar{x}$ is called degenerate. The whole problem (resp. polyhedron) is called nondegenerate, if all basic solutions (vertices) are nondegenerate.

Theorem 6
For nondegenerate problems the solution is achieved after at most $\binom{m}{n}$ pivot steps.

Proof: Here we have $\delta_{\hat{i}} > 0$ in each step, i.e. $c^T x$ improves strictly. And there are not more than $\binom{m}{n}$ different basic solutions, resp. vertices. Each of them is visited at most once.

Remark 4
A normal Simplex Algorithm may have difficulties in degenerate vertices (i.e. $\Delta \neq I$). There it may generate loops (a sequence of iterations returning to the same basis), while changing the bases, but not the vertex.
This danger can be avoided by using the special variant of Bland:
Choose in stage 1. of the Algorithm and (if necessary) in stage 3. each time the restriction

with minimal original index.
Application of Bland's Rule definitely prohibits loops and guarantees to finish the algorithm in at most $\binom{m}{n}$ *iterations.*

It remains to clarify how to start this algorithm.

Since we do not know a vertex $X = \{x \mid Ax \leq b\}$ in advance, our first problem will be to find such a vertex. Therefore we begin with solving a modified problem, called PI:

$$\begin{array}{lll} & \text{maximize} & x^{n+1} \\ PI: & \text{subject to} & Ax + x^{n+1}\mathbf{1} \leq b \\ & \text{and} & x^{n+1} \leq 0, \quad \text{where } (\mathbf{1}) := (1, \ldots, 1)^T. \end{array} \tag{17}$$

For this kind of problem we do know a feasible point, namely

$$w = \begin{pmatrix} 0 \\ \mid b^{min} \mid \end{pmatrix} \text{ with } \quad b^{min} = Min\{0, b^1, \ldots, b^m\}.$$

This point is not necessarily a vertex, but if we introduce $n+1$ additional auxiliary constraints

$$x^1 \geq 0, \ldots, x^n \geq 0, \text{ and } {a_j}^T x + x^{n+1} \leq b^j \text{ for the index where } b^j = b^{min}, \tag{18}$$

then w is a vertex of the restricted PI-polyhedron.
Now we try to get onto a vertex of purely original restrictions and to get rid of the auxiliary constraints. So we loosen these restrictions (one after the other). If the resulting edge is bounded, then we conduct a basis exchange, and if not we try the same in the opposite direction, which is allowed because of the provisional character of the auxiliary constraints.
After $n+1$ of these moves we finally are in a vertex, whose basis consists of original restrictions only.
At that moment we have a true vertex of the PI-polyhedron and we run an optimization process compatible with phase II, but this time for the PI-objective.

The outcome tells us whether X is feasible or not. If the optimal value of x^{n+1} turns out to be 0, then we have a feasible point of X, which can easily be transformed into a start vertex for Phase II. But if the optimal value is negative, then we can STOP because of INFEASIBILITY.

3.2 The Variable-oriented Simplex Method

Now we want to present the second version, called the "Variable-oriented Simplex Method". This form is the original Simplex Method that had been introduced by George Dantzig about 1947. It is specifically adapted to problems

$$\min c^T x \text{ s. t. } Ax = b \text{ and } x \geq 0,\ A \in \mathbb{R}^{(m,n)},\ c, x \in \mathbb{R}^n,\ b \in \mathbb{R}^m,\ m \leq n. \tag{19}$$

In this section we shall write $A_{\cdot i}$ resp. A_i for the i-th column of A. And we suppose that A has rank m.
In this context, we call a point a basic solution of the system if it satisfies $Ax = b$ and if at least $n - m$ of its components are 0, where the A-columns to the positive x-components are a linearly independent system.

Imagine such a partition of the indices $\{1, \ldots, n\}$ into two disjoint sets B and N (B for basic and N for nonbasic columns). In that way we also obtain a partition of the x-components $x = (x_B, x_N)$ for points solving the system $Ax = b$. Now we have in general

$$x = (x_B, x_N), \;\; b = A_B x_B + A_N x_N \text{ and } x_B = {A_B}^{-1} b - {A_B}^{-1} A_N x_N, \tag{20}$$

and since for basic solutions we shall have $x_N = 0$, it is clear that there $x_B = {A_B}^{-1} b$.
In the same way the effect on the objective function can be described:

$$c^T x = {c_B}^T x_B + {c_N}^T x_N \text{ implies that at the basic solution } \quad c^T x = c^T x_B \tag{21}$$

$$\text{and in general } \quad c^T x = c_B^T x_B + c_N^T x_N = {c_B}^T {A_B}^{-1} b + ({c_N}^T - {c_B}^T {A_B}^{-1} A_N) x_N. \tag{22}$$

So we see that the so-called "reduced cost-coefficient-vector" $({c_N}^T - {c_B}^T {A_B}^{-1} A_N)$ gives the information whether it pays to increase a component $x^i, i \in N$. This is the case, if

$$(c^i - {c_B}^T {A_B}^{-1} A_i) < 0. \tag{23}$$

Increasing exactly one such nonbasic variable, induces geometrically a move along an edge of $X = \{x \,|\, Ax = b,\, x \geq 0\}$.

Now the question arises as in the last section, how far that increment can be driven.
All we have to maintain is feasibility, i.e. $x \geq 0$ and $Ax = b$. x_N will stay at 0 (with the exception of x^i), so we have to concentrate on $x_B(x^i)$ only. But this is

$$x_B(x^i) = {A_B}^{-1} b - {A_B}^{-1} A_N x_N(x^i) = {A_B}^{-1} b - {A_B}^{-1} A_i x^i. \tag{24}$$

In order to keep that vector nonnegative, we should choose

$$x^i = Min \left\{ \frac{({A_B}^{-1} b)^j}{({A_B}^{-1} A_i)^j} \,|\, j \in B,\; {A_B}^{-1} {A_{\cdot i}}^j > 0 \right\}. \tag{25}$$

If there is no such index j with positive entry, than we can run the complete ray, increase x^i without any limit and by the way decrease $c^T x$ below any limit. In this case we could STOP because of unboundedness.
Else we find such a maximal value for x^i and a corresponding (basic) variable x^j. Now a realization of that (maximal possible) move makes x^i a basic and x^j a nonbasic variable. Then the new basis-nonbasis structure will be:
$B^{new} := B^{old} \cup \{i\} \setminus \{j\}$ and $N^{new} := N^{old} \cup \{j\} \setminus \{i\}$.
Algorithmically, this looks as follows:

Algorithm 2
Initialization
Let a vertex $\bar{x}$, the index-partition (B, N), and ${A_B}^{-1}A$ be given.
Typical step

1. *Check whether ${c_N}^T - {c_B}^T {A_B}^{-1} A_N \geq 0$.*
   If YES, then STOP because of OPTIMALITY.
   Else, select a nonbasic variable x^i with negative reduced cost coefficient $(c^i - {c_B}^T {A_B}^{-1} A_i)$ and use A_i as the pivot column for a basis-exchange.

2. *Check ${A_B}^{-1} A_i$. If this vector is completely nonpositive, then we STOP because of UNBOUNDEDNESS of the objective.*

3. *Conduct a quotient-comparison and determine the (resp. a) nonbasic variable $x_{\tilde{j}}$ which induces*

$$\frac{({A_B}^{-1}b)^{\tilde{j}}}{({A_B}^{-1}A_i)^{\tilde{j}}} = Min\left\{\frac{({A_B}^{-1}b)^j}{({A_B}^{-1}A_i)^j} \,|\, j \in B, {A_B}^{-1}{A_{\cdot i}}^j > 0\right\}.$$

   The column $A_{\tilde{j}}$ is bound to leave the basis.

4. *Perform a pivot step such that*
   $B^{new} := B^{old} \setminus \{\tilde{j}\} \cup \{i\}$ and $N^{new} := N^{old} \setminus \{i\} \cup \{\tilde{j}\}$
   $c^T x^{new} = c^T x^{old} + x^i(c^i - {c_B}^T {A_B}^{-1} A_i) < c^T x^{old}$.

5. *Update $\bar{x}$, (B, N), ${A_B}^{-1}A$ and go to 1.*

So far, we have dealt with Phase II only. Analogously to above, one sees that in all nondegenerate cases not more than $\binom{n}{m}$ iterations are required to solve the problem.

In the degenerate case again, it is sufficient to apply Bland's Rule in the following form:
In degenerate vertices one should choose i in stage 1 and $\tilde{j}$ in stage 3 least possible. This will avoid loops and assures the upper bound $\binom{n}{m}$ also under degeneracy.

It remains to find a starting vertex, i.e. we have to discuss Phase I.
Our first observation is that in $Ax = b$ we can provide a nonnegative capacity vector $\bar{b}$ by premultiplication with (-1) of all rows of A, b, where $b^i < 0$. The result will be a new system $\bar{A}x = \bar{b}$, which is equivalent to the old one.
Now for the new system we can introduce slack variables $u^i \geq 0,\ u \in \mathbb{R}^m$ and we can formulate a Phase I-problem

$$\begin{array}{lllll} \text{minimize} & \mathbf{0}^T x + \mathbf{1}^T u & & & \\ \text{subject to} & \bar{A}x + u & = \bar{b} & & (26) \\ & x \geq 0 & u \geq 0. & & \end{array}$$

For this problem we know the feasible point

$$\begin{pmatrix} x \\ u \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \bar{b} \end{pmatrix}. \tag{27}$$

It is very convenient that this feasible point is even a vertex of the modified PI-polyhedron and that this allows us to start immediately with the optimization process (compatible to Phase II). As in the last section, our optimization result will tell us whether our problem is feasible or not.

- If the optimal value for PI is positive, then the original problem is infeasible and X is empty.
- If the optimal value for PI is 0, then a simple transformation of the optimal vertex leads to a vertex of X, which is suitable for starting phase II.

3.3 Modifications of Methods and Problems

In a final section on solving LPs using the Simplex Method we try to present and to explain some additional topics, which are standard in LP-theory.

1. The Revised Simplex Method
   Since many LPs get extremely large, it is reasonable to save computation time wherever this is possible. This aim is realized in the Revised Simplex Method by an efficient use of Numerical Linear Algebra. We cannot describe details here, but we mention some saving ideas as guidelines.
   First, not all reduced cost coefficient components (resp. all impacts of edge-moves on the objective) need to be known. It suffices to have one such item showing a chance for improvement.
   Another resource for saving comes from the fact that it is not recommended to calculate the upcoming inverse matrices explicitly. Since the required information can be seen as the result of solving a system of equations, a stepwise approach according to the methods of Numerical Linear Algebra is more appropriate. For example: Factorization in upper and lower triangle matrices, representation of the current matrix as a product of the original matrix with a sequence of so-called "Eta-matrices" (describing the effect of basis-exchanges) and reinversions, i.e. new calculation of a correct current data set from time to time, are strongly recommended. These tricks pay in particular, if the original matrix is sparse (i.e. most of its entries are 0.)

2. Dual Simplex Algorithms
   We have learned that the Restriction–oriented method is directly adapted to problems in canonical form and that the Variable-oriented method is adapted to problems in standard form. In both cases the algorithms produce a sequence of feasible points (vertices) while improving the value of the objective function, until the optimal vertex is found. This is the way a primal problem is solved.
   But as we have noticed in our section on duality theory, such an algorithm also solves -

in a certain sense - the dual problem. So imagine, that we use our two algorithms crosswise (i.e. Restriction–oriented for standard problems and Variable–oriented for canonical problems). In both cases we may start at a basic solution, which in general is not a feasible point, but which is "dual feasible". That means that at the basic solution, the objective direction is in the respective cone of the gradients to those restrictions which are tight at the current point. In other words: A basic solution $\bar{x}$ is dually feasible if it satisfies the following (somehow artificial) condition: If we drop all the violated restrictions at $\bar{x}$, then $\bar{x}$ will be optimal on the (now larger) feasible region $\bar{X}$.
Then both algorithms move from such dual basic solutions to adjacent ones. The objective for the problem under consideration gets worse in each step. But eventually we reach the primal feasible region or we recognize that X is empty. And (according to the explanation above) in that moment that all restrictions are satisfied, we are in the total primal optimum. This behaviour is also called "outer" algorithm. The purpose of achieving feasibility a posteriori can also be used in Post-Optimization, where certain data of the LP have been changed.

3. Post-Optimization
For many purposes (also for the solution of integer or combinatorial problems) it is very useful to consider the following configuration: Suppose that a problem has been solved, and that an optimal vertex is available, but that afterwards some data of the original problem change or turn out to be false.
For example the capacity vector may differ from the old one, or our objective may have changed, or some entries of the matrix may be modified. Then it would be advantageous to use the current optimum as a start for the correction algorithm. This would be preferred against a complete re-solution of the problem.
The method for such a correction is called Post-Optimization. Mainly it makes use of the techniques of primal or dual (internal or external) algorithms as described in the section before.

4. Parametric Optimization
In practice often a similar, but not identical question arises. Suppose that a part of our data set is considerable in two different versions (and at the moment we cannot decide which one shall be used). Then it may be interesting to learn about all potential optimal solutions not only for the two mentioned extremal versions, but also for all mixtures of these two versions.
For example, let two objectives ${c_1}^T x$ resp. ${c_2}^T x$ be in discussion. Then it is interesting to know all optima corresponding to the convex combinations of both objectives. That means that we solve an optimization problem for each mixed objective
$\lambda {c_1}^T x + (1 - \lambda c_2)^T x$ with $0 \leq \lambda \leq 1$.
Interestingly, the set of these resulting optimal vertices forms a Simplex Path. That means, that if we start at the optimal vertex for $c_1 (\equiv \lambda = 1)$, then a certain variant will lead us over all optima for decreasing λ to the optimum for $c_2 (\equiv \lambda = 0)$. On this path ${c_2}^T x$ has been successively improved. This variant is called the parametric variant of the Simplex Method. If we have this sequence of optima, we may and can make the decision about the objective (which is to be realized) afterwards, i.e. in view of the calculated optima.
This concept can be generalized to e.g. a finite number of objectives. Then we speak of "Multiobjective Programming" (see Multicritera Decision Making).

3.4 The Complexity of the Simplex Method

We have already learned that both in nondegenerate and in degenerate problems as well the Simplex Method will (e.g. for the canonical form) require not more than $\binom{m}{n}$ iterations, because this is the maximal possible number of different bases in a system $Ax \leq b$ with $A \in \mathbb{R}^{(m,n)}$. But the true effort for solving such problems may be significantly less. In this paragraph we will briefly discuss some insights on that question.
In order to get a formally justified general measure for the complexity of algorithms, we consider the encoding length of the data set for the problem that is to be solved. In case of an LP these data are A, b, c and the encoding length (resp. the number of binary bits to store all numbers of the data set) will be denoted by $\langle\,\rangle$(see Complexity Theory).
So we observe that

$$\langle LP\rangle = \langle (A,b,c)\rangle = \langle A\rangle + \langle b\rangle + \langle c\rangle$$

After having classified all LPs according to their encoding length, and thus forming classes $\Pi(N)$ of problem-instances with equal encoding length N (which does depend on the dimensions and on the digit-precision of the single numbers), it is possible to attribute a worst case running time to each such class.
To formalize the term running time, we consider all the arithmetical operations which have to be carried out until a problem is solved. In each operation two numbers are involved and we sum up the two encoding lengths. So we have the number of bits for that calculation. Finally, we sum this figure up over all such calculations and regard this sum as a formal computing time.
Now we assume that all problems in the class of encoding length N are solved using an algorithm Alg and we consider the highest running time which has occured in that class. Regarding all these results for all $N \in \mathbb{N}$, we have a function $t_{algorithm} : \mathbb{N} \to \mathbb{N}$ defined as

$$t_{algorithm}(N) := Sup\{t_{algorithm}(I)\,|\,I \in \Pi(N)\}, \quad \text{where } \Pi(N) = \{I\,|\,\langle I\rangle = N\}. \tag{28}$$

Such a "worst-case behaviour function" can be judged according to its growth in N.

Definition 5
An algorithm Alg is said to have polynomial worst case complexity, if there is a polynomial $\tilde{P}$ such that $t_{algorithm}(N) \leq \tilde{P}(N)$ for all $N \in \mathbb{N}$.
Analoguosly one speaks of linear ($\leq CN$), quadratic ($\leq CN^2$), cubic ($\leq CN^3$) or of exponential ($\leq Cexp(N)$) complexity resp. behaviour.

Now it is a simple consequence of Linear Algebra (see Linear Algebra) that every number occurring in the Simplex Method during the solution process is between

$$2^{(-\langle A\rangle+\langle b\rangle+\langle c\rangle)-2n} \quad \text{and} \quad 2^{\langle A\rangle+\langle b\rangle+\langle c\rangle-2n}. \tag{29}$$

Since not more than $O(mn)$ numbers or tableau-entries have to be updated in each step (with one addition and one multiplication per iteration), it is clear that the effort for one pivot step is polynomial in $L := \langle A\rangle + \langle b\rangle + \langle c\rangle = \langle (A,b,c)\rangle$.

It remains to clarify, how many pivot steps are necessary until we can stop. Although the practical experience with the computation time of the Simplex Method had been excellent, there was a great disappointment, when Klee and Minty 1972 made the following observation for one variant of the Simplex Method (the analogous results were achieved later for all usual variants):

Theorem 7
There is a certain family of LPs with $m = 2n$ restrictions in $\leq$-form and with n variables that forces the given variant of the Simplex Method to visit all the available 2^n vertices of X. This means, that an exponential number of pivot steps has to be carried out.

The special form of these LPs is :

$$\begin{array}{lccccc} \text{maximize} & {e_n}^T x & & & & \\ \text{subject to} & 0 & \leq & x^1 & \leq & 1 \\ & \epsilon x^i & \leq & x^{i+1} & \leq & 1 - \epsilon x^i \quad \text{for } i = 1, \ldots, n-1 \\ & & \text{where} & \epsilon \in (0, \frac{1}{2}). & & \end{array} \tag{30}$$

Such Klee-Minty-polytopes resp. closely related objects show that all well-known variants of the Simplex Method have an exponential worst-case behaviour (nonpolynomial). But until now it could not be proven that there is no polynomial variant of the method.

Much better is the observation and the impression, when we care about the average-case behaviour. (Among others) the author of this survey has (around 1980) clarified that for a certain stochastic distribution of randomly generated LP-instances, namely the so-called "Rotation-Symmetry-Model", the expected number of required pivot steps is polynomial in both dimensions m and n. This model generates problems of the type

$$\text{maximize } c^T x \quad \text{s.t.} \quad a_1^T x \leq 1, \ldots, a_m^T x \leq 1, \quad \text{where} \quad x, c, a_1, \ldots, a_m \in \mathbb{R}^n. \tag{31}$$

And it assumes a distribution of the linear programming problems with the following properties:

$$\begin{array}{l} \text{The vectors } a_1, \ldots, a_m, c \text{ are distributed on } \mathbb{R}^n \backslash \{0\} \text{ independently,} \\ \text{identically and symmetrically under rotations.} \end{array} \tag{32}$$

Under these conditions it can be proved that

Theorem 8
*For all distributions according to our rotation-symmetry-model (**??**) we have the following upper bound for the expected number of pivot steps required for solving problems in n variables and m restrictions:*

$$E_{m,n}(s) \leq m^{\frac{1}{n-1}} \cdot n^2 \cdot Const. \tag{33}$$

This gives a much more helpful statement about the quality of the Simplex Method in practical applications. It shows that the impact of the very bad examples is rather small and that they are quite artificial and seldom in reality.

4 Polynomial Solution Methods for LPs

4.1 The Ellipsoid Method

In 1979, the Russian Mathematician Khachiyan developed and analyzed an algorithm for LPs which turned out to be - in a certain sense - polynomial in the encoding length (in contrast to what is known about the Simplex Method).
In principle, this had been a method for deciding whether an inequality-system has a solution or not.
If L is the encoding length for the system (A, b), then the analysis is based on a certain perturbation lemma.

Lemma 4

1. *$Ax \leq b$ has a solution x if and only if there is also a solution for*
2. *$2^L Ax < 2^L b + \mathbf{1}$.*

The volume of the solution set of 2. can be bounded from below.

Lemma 5
If 1. is solvable, then there is a solution point $\bar{x}$ with $\|\bar{x}\| \leq 2^L - 2^{-L}$, such that a ball of radius 2^{-2L} around $\bar{x}$ belongs to the solution set of 2.

A further insight is that such a ball around $\bar{x}$ of radius 2^{-2L} cannot contain more than one vertex, which would enable us to "round" exactly, if we had to search only in that ball.

Khachiyan's algorithm constructs a sequence of ellipsoids and combines that with a stopping criterion.

Algorithm 3
Initialization
Let be given: $x_0 := 0,\ B_0 := E2^{2L},\ k := 0,\ Ell_0 := \{x \mid (x - x_k)^T {B_k}^{-1}(x - x_k) \leq 1\}$.
(E is the unit matrix and a special ellipsoid Ell_0 is implicitly defined by B_0. This start ellipsoid is a ball with radius 2^L).
Typical Step

1. *Check whether the point x_k satisfies all the restrictions ${a_i}^T x_k \leq b^i + 2^{-L}$.*
   If YES, then STOP because of FEASIBILITY. Else proceed.
2. *Choose the gradient a_i to one of the violated constraints and set $a := a_i$.*
   (The hyperplane $\{x \mid {a_i}^T x_k = a^T x\}$ divides Ell_k in two half-ellipsoids. In one of these two every point violates the i-th restriction. It remains to check the other (the interesting) half-ellipsoid.)

3. *Now calculate a new point and a new matrix*

$$x_{k+1} := x_k - \frac{1}{n+1}\frac{B_k a}{\sqrt{a^T B_k a}} \text{ and } B_{k+1} := \frac{n^2}{n^2-1}\left[B_k - \frac{2}{n+1}\frac{B_k a (B_k a)^T}{a^T B_k a}\right].$$

(These figures help to construct Ell_{k+1}, the smallest ellipsoid with center x_{k+1} that contains the interesting half-ellipsoid of Ell_k from 2.)

4. *If $k+1 \geq 6n(n+1)L$ then STOP because of INFEASIBILITY.*
Else set $k := k+1$ and go to 1.

Figure 3: Ellipsoid Method

Figure 3 shall illustrate how the Ellipsoid Method works. The polygon symbolizes the feasible region. The first (horizontal) ellipsoid has an infeasible center (x). Therefore we construct a second ellipsoid, containing the complete right half of the first one. The separating line is parallel to the edge of the polygon standing for a violated constraint. The volume of the second ellipsoid is slightly smaller and still the polygon is a subset. In our example, the center of the second ellipsoid (o) is feasible. So we are ready.

The construction of the ellipsoids in that algorithm is done in such a way that their volumes shrink from step to step by a factor of at least $exp(-\frac{1}{2(n+1)})$. On the other hand it provides that the small feasible ball in the polyhedron – if there are feasible points at all – is contained in each of the ellipsoids.
But after conducting $6Ln(n+1)$ iterations, the volume of the ellipsoid Ell_k has become so small, that it is less than the volume of the mentioned small ball. And this would cause a contradiction if the polyhedron would be feasible.
Now it becomes clear that the algorithm must have stopped with a feasible point x_k in any case, where there are feasible points at all, and this will happen before step $6Ln(n+1)$. The opposite event $k > 6Ln(n+1)$ implicitly proves infeasibility.

So far, we have talked about solving a system of inequalities only. But this can be used to solve LPs, too, e.g. with binary search. Here we define systems which (in addition to feasibility) require that the objective gets better than a given lower bound. By checking a polynomial number of such bounds we are able to find the optimal value and the optimal point.
Another, more direct way is to formulate an extended system which comprises the feasibility restrictions of the primal and the dual problem and makes use of the fact of the Theorem of Weak Duality, that the primal and dual objective values can coincide only in optimal points. Such a system reads

$$\begin{array}{rcrcl} c^T x & - & b^T y & \leq & 0 \\ Ax & & & \leq & b \\ & - & A^T y & \leq & -c \\ -x & & & \leq & 0 \\ & & -y & \leq & 0. \end{array} \tag{34}$$

So the ellipsoid method is in fact polynomial in the encoding length L, but not purely in the dimensions (m,n). To distinguish both concepts, the term "strong polynomiality" has been

introduced. So the Ellipsoid Method is polynomial but not strongly polynomial.
The fact that the encoding length does not only influence the necessary precision of the calculations, but also the number of iterations, is a non-desired side-effect. But, in addition, the Ellipsoid Method did in practice not stand the competition with the Simplex Method, because the condition numbers of the matrices B_k became too small and the ellipsoids became too degenerate (see Numerical Linear Algebra). So this success of polynomiality remained a theoretical one.

4.2 Interior Point Methods

Since 1984 the so called interior point methods have been serious competitors for the Simplex Method. This originated in a proposal of Narendra Karmarkar to produce a sequence of inner points in a way that they converge to the optimal vertex (in this case the minimal vertex). So far, such an idea had been present, but mostly it did not work for some of the following reasons. Usually in Nonlinear Programming (see Nonlinear Programming) a better point is searched in the following manner: first one determines a certain direction (from the current point), which promises to lead to better points resp. lower values.

Figure 4: Difficulty of moving in the interior

In the example of figure 4 one tries to reach the uppermost vertex. But if our iteration point is the left one, then a move strictly upward –along the steepest descent– will soon be stopped. It is preferable to use a barrier function, which grows rapidly at the border. This leads to a rotation of the gradient such that a move along the new steepest descent is much more promising.

Then one tries to find the minimal objective value or the minimum feasible point on the ray defined by the current point and the search direction. But this search is restricted to the feasible segment of the ray. So although we may be successfull in finding this minimum, what we do may turn out to be highly inefficient, if we hit the boundary of X in our move very soon. This does - of course - depend strongly on the choice of the search direction. Taking the steepest descent as the search determinator may be much slower than choosing a less descending (but long feasible staying) direction.

4.2.1 Minimizing Potential Functions

Karmarkars principal idea (and the progress in comparison to the situation before) was to apply an iteration specific projective transformation on the polyhedron depending on the current iteration point, which moves the iteration point into the center and which deformates the polyhedron in such a way that it becomes quite "thick" and "round", such that the center is – in any direction – far from the boundary. All these transformations can be inverted and so we produce (as before) a sequence of original points in the original space, but with a measurable and significant progress in the auxiliary space. A similar effect can be achieved by avoiding that complicated projective transformation and by applying only a so-called scaled transformation. This works by premultiplying all points of the original space with a diagonal matrix

$$X_k{}^{-1} = Diag((x_k^1)^{-1}, \ldots, (x_k^n)^{-1}). \tag{35}$$

This transformation sends the iteration point x_k into the image-point $(1,\dots,1)^T$, which is equally distant from every bounding hyperplane of the positive orthant (and thus quite central).

The algorithmic tool for improving the objective is the method of

Scaled Steepest Descent
We try to minimize a function $\varphi(x)$ subject to $Ax = b$ and to $x \geq 0$.
This algorithm transforms each x_k into the point $(1,\dots,1)^T$.
Instead of $\varphi(x)$ we now work with $\bar{\varphi}(y) := \varphi(X_k y)$. To find the search direction h, we project the (negative) gradient of $\bar{\varphi}(y)$ at $y = \mathbf{1}$ on the kernel of AX_k.
Then we try to find the point of the form $y^* = \mathbf{1} + \lambda h, y^* \geq 0$, which minimizes $\bar{\varphi}(y)$.
This point is then inverted into the original space, resp. in $x_{k+1} = X_k y^*$.

Now we should think about an appropriate function, which can play the role of φ if an LP has to be solved. As mentioned before, a linear function would not work as desired.

Let us attack the LP

$$\text{minimize } c^T x \;\text{ subject to }\; Ax = 0,\;\; \mathbf{1}^T x = 1,\;\; x \geq 0. \tag{36}$$

Every LP in general form can with limited effort be represented or transformed in that way.

Now it turns out that this problem can be solved by considering (instead of $c^T x$) the so-called potential function

$$\varphi(x) = n \log(c^T x) - \sum_{i=1}^{n} \log(x^i). \tag{37}$$

The term $-\sum_{i=1}^{n} \log x^i$ is a barrier function on the bounded feasibility region (and on the positive orthant). So it tends to ∞ at the boundary and it remains bounded from below on X. This enables us to show that as soon as we achieve that the first term $n \log(c^T x)$ gets less than $-nL$, then $c^T x < \exp(-L) < 2^{-L}$. (And this is just what we want).

Now back to our method of Scaled Steepest Descent. It can be proven that this algorithm does in each iteration decrease the value of the above-mentioned potential function by at least 0.1. Since φ has an initial value of less than $2nL$, it suffices to carry out $30nL$ steps to achieve the desired precision (see Steepest Descent, Barrier Functions, Nonlinear Programming).

So, an upper bound of $30nL$ iterations is proved and an analysis of the arithmetical effort per iteration leads to the result that in total $O(n^{3.5}L)$ arithmetical operations are required. If we conduct them with precision in L bits, this would mean an effort of $O(n^{3.5}L^2)$. And so we have a polynomial behaviour.

4.2.2 Following the Central Path

Another type of such interior point algorithms makes use of the fact that bounded polyhedra contain a so-called Central Path in the polyhedron of P resp. in the pair of dual feasible sets

to (P, D).

Suppose that we have to deal with the following dual pair of problems and assume that both have feasible inner points

$$P := \begin{array}{lrcl} \min & c^T x & & \\ \text{s.t.} & Ax & = & b \\ & x & \geq & 0 \end{array} \quad \text{and} \quad D := \begin{array}{lrcl} \max & b^T y & & \\ \text{s.t.} & A^T y + z & = & c \\ & z & \geq & 0 \end{array} \tag{38}$$

standard form canonical form

Then we would have a solution for both if we could solve the system

$$(S_\mu) := \begin{array}{rcl} Ax & = & b \\ A^T y + z & = & c \\ x, z & \geq & 0 \\ Xz & = & \mu \mathbf{1} \end{array} \tag{39}$$

with $X = Diag(x^1, \ldots, x^n)$ for the special value $\mu = 0$.
But it could be helpful to observe the total set of such solutions and systems (for general $\mu > 0$). Perhaps this will lead us to the desired solution of S_0.
This system has for each S_μ a unique solution $\begin{pmatrix} x(\mu) \\ y(\mu) \\ z(\mu) \end{pmatrix}$.
And it turns out that the first section of this solution vector, namely $x(\mu)$, is just the minimal point of the logarithmic barrier function

$$f(x, \mu) = \frac{(c^T x)}{\mu} - \sum_{i=1}^{n} \log x^i. \tag{40}$$

So we recognize a so-called central path for P and D.

Definition 6
The central path on (P) is the set $\{x(\mu) \,|\, \mu \geq 0\}$.
And the central path on (D) is the corresponding set $\{y(\mu) \,|\, \mu \geq 0\}$.

We make the following observations.

Lemma 6
For all values of $\mu > 0$ *we have*

$$c^T x(\mu) - b^T y(\mu) = x(\mu)^T (c - A^T y(\mu)) = x(\mu)^T z(\mu) = n\mu. \tag{41}$$

And for decreasing values of μ *,* $c^T x(\mu)$ *is decreasing and* $b^T y(\mu)$ *is increasing.*

The main advantage of this central path will be that it serves as a reliable guide how to find the points $x(\mu')$ for $\mu' < \mu$, if we are in or close to $x(\mu)$.
To understand this, one needs a measure for the distance to the central path.

Definition 7
The measure of x to centrality under the parameter μ to is defined as

$$\delta(x,\mu) := Min\{\|\frac{1}{\mu}Xz - \mathbf{1}\| \,|\, y,\ z \text{ such that } A^T y + z = c\}. \tag{42}$$

This measure results from the attempt to solve a quadratic optimization problem on $\{(y,z) \,|\, A^T y + z = c\}$. The unique solutions for that are pairs $(y(x,\mu), z(x,\mu))$.

Now one should try to get as close as possible to centrality. Thus we employ the Newton-Method on $\{x \,|\, Ax = b\}$ to minimize the mentioned function $f(x,\mu)$. And we can calculate which move is done by the Newton method when x, μ are at hand (see Newton-Method).
If we denote the move (the difference between successive iteration points) of the Newton-Method by a vector $p(x,\mu)$, then this vector turns out to be

$$p(x,\mu) = X\Pi_{AX}(\mathbf{1} - \frac{1}{\mu}Xc), \tag{43}$$

where $X = Diag(x^1, \ldots, x^n)$ and Π_{AX} is the projection on the kernel of AX.

Then it is important to know that

$$\delta(x,\mu)^2 = p(x,\mu)^T X^{-2} p(x,\mu). \tag{44}$$

Being close to the central path will allow us to improve the value of μ significantly without losing contact to the central path. This comes from two facts.

Lemma 7

1. *If $\delta(x,\mu) < 1$ then the dual partner of x, namely $y(x,\mu)$, is a feasible point of D and we know that*

   $$\mu(n - \delta\sqrt{n}) \leq c^T x - b^T y \leq \mu(n + \delta\sqrt{n}) \tag{45}$$

2. *If $\delta(x,\mu) < 1$ then the point $x^* = x + p(x,\mu)$ is strictly feasible for P and*

   $$\delta(x^*,\mu) \leq \delta(x^*,\mu)^2. \tag{46}$$

From this idea one can develop algorithms which try to reduce the value of μ while staying close to the central path. The fundamental property is

Lemma 8
Let $\delta(x,\mu) \leq \frac{1}{2}$. If we then move from x to $x^{new} := x + p(x,\mu)$ and if we consider a value $\mu^{new} := \mu(1 - \frac{1}{6\sqrt{n}})$ then the resulting combination will satisfy $\delta(x^{new}, \mu^{new}) \leq \frac{1}{2}$.

This property is exploited in an algorithm with "short steps", working as follows:

Algorithm 4
Initalization
Given a pair (x_0, μ_0) such that $\delta(x_0, \mu_0) \leq \frac{1}{2}$, $\mu_0 = O(2^L)$. Set $k := 0$.
Typical Step

1. *If $n\mu \leq \exp(-L)$ then STOP.*
2. *Set $x_{k+1} = x_k + p(x,\mu)$ and $\mu_{k+1} := (1 - \frac{1}{6\sqrt{n}})\mu_k$.*
3. *Set $k = k + 1$ and go to 1.*

Using our knowledge from above, the following Theorem can be proven.

Theorem 9
The algorithm with short steps stops after at most $6\sqrt{n}L$ iterations with a precision that is sharp enough to identify the best vertex uniquely.

Other concepts try to realize longer steps such that μ can be reduced faster. But than it cannot be guaranteed that we stay in the neighbourhood of $\delta(x,\mu) < \frac{1}{2}$. However, it is possible to make such a long step and – afterwards – reduce iteratively the value of $\delta(x^{new}, \mu_{new})$ until we are at least in a $(\delta = \frac{1}{2})$–neighbourhood. Then we reduce μ again. This type of algorithm is called a "predictor-corrector algorithm". It can be implemented in a way such that again the complexity is of order $O(\sqrt{n}L)$.

The following insights are fundamental for the success of such a predictor-corrector-algorithm.

Lemma 9
Let $\delta(x,\mu) \geq \frac{1}{2}$. If we then move from x to $x + \frac{1}{1+\delta}p(x,\mu)$, then

$$f(x + \frac{1}{1+\delta}p(x,\mu), \mu) \leq f(x,\mu) - \frac{1}{12}. \tag{47}$$

And if $\delta(x,\mu) \leq \frac{1}{2}$, then

$$f(x,\mu) - f(x(\mu),\mu) \leq \frac{1}{3}. \tag{48}$$

Figure 5: Comparison of short and long step method

Figure 5 compares the method of short steps with that of long steps. In the left method with short steps one iterates until the sufficient neighbourhood of the central path is reached. The

following iterations reduce the distance to the optimum by short steps close to the central path, which are all made very cautious such that one stays in that neighbourhood.
In the right picture with long steps one has again to reach the neighbourhood, but then we try to reduce the parameter significantly. But this may lead us far away from the central path. Before we proceed, we have to make correction steps in order to get back to the central path.

Then this algorithm works as follows.

Algorithm 5
Initalization
Given a pair (x_0, μ_0) *such that* $\delta(x_0, \mu_0) \leq \frac{1}{2}$, $\mu_0 = O(2^L)$. *Set* $k := 0$.
Typical Step

1. *If* $n\mu \leq \exp(-L)$ *then STOP.*
2. *If* $\delta(x_k, \mu_k) \leq \frac{1}{2}$ *then go to 5.*
3. $x_k := x_k + \frac{1}{1+\delta} p(x_k, \mu_k)$.
4. *Go to 2.*
5. *Set* $x_{k+1} = x_k$ *and* $\mu_{k+1} := (1 - \frac{1}{\sqrt{n}})\mu_k$.
6. *Set* $k = k + 1$ *and go to 1.*

Here we have inner iterations (2.–4.) and outer iterations (1.–5.). And finally we arrive again at a complexity of $O(n^3 L)$, having in mind that we need $O(n^{2.5})$ for the single step.

Again, we observe theoretical polynomiality, but not strong polynomiality.
But these algorithms are serious competitors for the Simplex Method. They can be implemented in a numerically stable way and they turn out to be superior to the Simplex Method in those configurations, where the matrix is very sparse and where the number of variables is extremely high. In the remaining cases, still the Simplex Method seems to be the best choice, in particular because of its clearness and because it delivers the optimal points directly without a need to make final roundings.

References

[1] Bazaraa, M.-S., Sheraly, H.D., Shetty, C.M.(1993): *Nonlinear Programming*, Wiley, New York. [A book telling all the necessary knowledge about Nonlinear Programming and its use for Linear Programming]

[2] Bland, R. G., Goldfarb, D., Todd, M. J.(1981): *The Ellipsoid Method: A Survey*, in: Operations Research, Vol. **29** No. 6, 1981, 1089-1091. [All the fundamentals of the Ellipsoid Method]

[3] Borgwardt, K. H.(1987): *The Simplex Method – A Probabilistic Analysis*, Springer Verlag, Berlin, Heidelberg. [A detailed monography about the calculation of the Average-Case-Complexity of LPs]

[4] Borgwardt, K. H.(1999) : *A Sharp Upper Bound For The Expected Number Of Shadow Vertices In LP-Polyhedra Under Orthogonal Projection on Two-Dimensional Planes*, Mathematics of Operations Research, Vol.**24** No. 4, 925–984. [An article leading to the sharpest results in the Average-Case-Analysis]

[5] Chvátal, V.(1983):*Linear Programming*, Freeman, New York. [A very instructive book on LP, including applications and presentation of the Revised Simplex Method]

[6] Dantzig, G. B.(1974): *Linear Programming and Extensions*, Princeton University Press, Princeton. [THE fundamental work on Linear Programming]

[7] Gill, P., Murray, W., Wright, M.(1991): *Numerical Linear Algebra and Optimization*, Vol. 1, Addison-Wesley Publishing Company, Redwood City. [Fundamental knowledge of Numerical Linear Algebra as far as it is useful for LP]

[8] Grötschel, M., Lovász, L., Schrijver, A.(1988): *Geometric Algorithms and Combinatorial Optimization*, Springer Verlag, Berlin, Heidelberg. [A book on the use of the Ellipsoid Method for the Complexity of Linear Programming and for Combinatorial Optimization]

[9] den Hertog, D.(1994): *Interior Point Approach to Linear, Quadratic and Convex Programming*, Kluwer Academic Publishers, Dordrecht. [A broad survey over the various forms and purposes of Interior Point Methods]

[10] Karmarkar, N.(1984):*A New Polynomial-Time Algorithm for Linear Programming*, in: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing (Washington, D. C., 1984), The Association for Computing Machinery, New York, 1984, 302-311 (revised version: Combinatorica **4** (1984), 373-395). [The original and first article on Interior Point Methods]

[11] Klee, V., Minty, G. J.(1972): *How Good is the Simplex Algorithm?*, in: Shisha, O. (ed.), Inequalities III, Academic Press, New York, 1972, 159-175. [The article where the Klee-Minty polytopes were presented and the exponential behaviour of the Simplex Method was shown]

[12] Luenberger, D. G.(1989): *Linear and Nonlinear Programming*, Addison-Wesley, Reading, Mass. [A book providing a lot of useful numerical methods for linear and nonlinear programming]

[13] Mangasarian, O. L.(1969): Nonlinear Programming, McGraw-Hill, New York. [A fundamental insight into the theory of Nonlinear Programming]

[14] Murty, K. G.(1983):*Linear Programming*, John Wiley & Sons, New York. [An extensive work on the whole world of LPs]

[15] Padberg, M.(1995):*Linear Optimization and Extensions*, Algorithms and Combinatorics 12, Springer Verlag, Berlin, Heidelberg, New York. [A book showing the tremendous use of LP for real world problems and for Combinatorial Optimization]

[16] Rockafeller, R. T.(1970):*Convex Analysis*, Princeton University Press, Princeton. [A book presenting the fundamentals of Convexity and of Polyhedra]

[17] Saigal, R.(1995): *Linear Programming. A Modern Integrated Analysis*, International Series in Operations Research & Management Sciences, Kluwer Academic Publishers, Dordrecht. [A book showing the modern state of the art in Interior Point Methods]

[18] Schrijver, A.(1986): *Theory of Linear and Integer Programming*, Wiley. [A fundamental book on the theory of linear programming]

[19] Stoer, J., Witzgall, C.(1970): Convexity and Optimization in Finite Dimensions I, Springer Verlag, New York. [A book telling the theory behind Duality, Convexity and Polyhedra]