Combinatorial Optimization: The Interplay of Graph Theory, Linear and Integer Programming Illustrated on Network Flow

Annegret K Wagler

Combinatorial optimization: The interplay of graph theory, linear and integer programming illustrated on network flow

Annegret K. Wagler

Abstract Combinatorial Optimization is one of the fields in mathematics with an impressive development in recent years, driven by demands from applications where discrete models play a role. Here, we intend to give a comprehensive overview of basic methods and paradigms, in particular the beautiful interplay of methods from graph theory, geometry, linear and integer programming related to combinatorial optimization problems. To understand the underlying framework and the interrelationships more clearly, we illustrate the theoretical results and methods with the help of flows in networks as running example. This includes on the one hand a combinatorial algorithm for finding a maximum flow in a network, combinatorial duality and the Max Flow/Min Cut-Theorem as one of the fundamental combinatorial min-max relations. On the other hand, we discuss solving the network flow problem as linear program with the help of the Simplex Method, linear programming duality and the dual program for network flow. Finally, we address the problem of integer network flows, ideal formulations for integer linear programs and consequences for the network flow problem.

1 Introductory remarks on Combinatorial Optimization

Combinatorial optimization problems occur in a great variety of contexts in science, engineering and management. All such problems have the goal to find the best of something. In mathematical terms, this is expressed with the help of an *objective function:*

$$\max \text{ or } \min c(\mathbf{x}),\ \mathbf{x} \in \mathbf{R}^n.$$

Annegret K. Wagler
Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS) / CNRS
Université Blaise Pascal (Clermont-Ferrand II)
BP 10125, 63173 Aubière Cedex, France
e-mail: Annegret.WAGLER@univ-bpclermont.fr

In practical settings, finding the best of something typically includes some *side constraints*. In mathematical terms, this can be expressed with the help of some function(s) $f : \mathbf{R}^n \to \mathbf{R}$. The functions involve certain *variables* $\mathbf{x} \in \mathbf{R}^n$. This leads to the following classical optimization problem:

$$\begin{aligned} \max \text{ or } \min & \, c(\mathbf{x}) \\ \text{subject to } & f_1(\mathbf{x}) \leq b_1 \\ & \vdots \; \vdots \; \vdots \\ & f_k(\mathbf{x}) \leq b_k \\ & \mathbf{x} \in \mathbf{R}^n \end{aligned}$$

The points $\mathbf{x} \in \mathbf{R}^n$ satisfying all side constraints $f(\mathbf{x}) \leq b$ are called *feasible*. The set of all feasible points is called the *feasible region* of the optimization problem. If all side constraints are linear functions, the above optimization problem is a linear program and the feasible region is a convex set, which allows to solve the problem in polynomial time.

If the studied objects are entities as workers, planes,... which cannot be divided, it is necessary to use integral variables $\mathbf{x} \in \mathbf{Z}^n$ or decision variables $\mathbf{x} \in \{0,1\}^n$ which makes the corresponding integer linear programs computationally more demanding.

This is typically the case for combinatorial optimization problems, where the goal is to search for an optimum object in a finite collection of certain objects. Hereby, the objects have a concise representation within a discrete structure (like a graph or a network), but their number is huge such that scanning all objects to select the best one among them is not an option. The aim of Combinatorial Optimization is to find more efficient solution methods.

The first step towards solving a problem is always to build a mathematical model: it helps to correctly formalize the problem, that is, to decide which conditions are crucial to describe the problem, and how to formalize them appropriately. This can reveal relationships by gaining structural insight of the problem, for instance in terms of bounds for the objective function value arising from dual combinatorial objects. The second step is to develop methods for finding a feasible solution, and to certify optimality (without knowing the optimal solution before). In addition, it is important to study the complexity of the problem, that is, to answer the question how hard or easy the studied problem is.

In this chapter, we shall discuss how to model and solve combinatorial optimization problems, illustrated with the help of the well-studied network flow problem as running example.

Problem 1 (Network Flow Problem). Find a maximal flow, that is, transport the maximal amount of certain goods (or water, electricity, cars, ...), through a given transportation network (consisting of pipelines, streets, etc.).

In Section 2, we first address the Network Flow Problem from a combinatorial point of view. This includes to model the problem with the help of an appropriate discrete structure (a network) and the studied combinatorial objects therein (a flow).

We discuss combinatorial duality and the Max Flow/Min Cut-Theorem as one of the fundamental combinatorial min-max relations. Moreover, we present Ford-Fulkersons's combinatorial algorithm for finding a maximum flow in a network.

In Section 3, we introduce linear programs and show how to formulate the Network Flow Problem in this context. Next, we discuss the geometry of the feasible region of linear programs and its impact on solving linear programs with the help of the Simplex Method. Furthermore, we address linear programming duality and consider the dual program for network flow.

Finally, in Section 4, we introduce integer linear programs, linear programming relaxations for integer linear programs and ways to strengthen them. We conclude with the problem of integer network flows, discuss ideal formulations for integer linear programs related to totally unimodular matrices, and consequences for the network flow problem.

2 A combinatorial algorithm for network flow

The combinatorial formulation of the Network Flow Problem involves both an appropriate discrete structure to model the input of that problem and a combinatorial object therein to describe the desired output:

- *Model:* construct a directed graph with transportation ways (pipes, streets, ...) as directed arcs, their crossing points (connections, swivel valves, ...) as nodes, and arc weights as capacities;
- *Task:* find a maximal flow through the network (respecting the arc capacities).

We first introduce the underlying discrete structures. For that, consider a *digraph* $D = (V,A)$ with node set V and arc set A where each arc $a = (u,v) \in V \times V$ is an ordered pair. We say that $a = (u,v)$ is the arc *outgoing* from u and *ingoing* to v and denote by

$$\delta^-(v) = \{a \in A : a = (u,v)\}$$

the set of arcs ingoing to v and by

$$\delta^+(v) = \{a \in A : a = (v,u)\}$$

the set of arcs outgoing from v. A *directed path* is a subgraph of D with (distinct) nodes $v_1, \ldots, v_k \in V$ and (exactly) the arcs $(v_i, v_{i+1}) \in A$ for $1 \leq i < k$, and is called (v_1, v_k)-path if it links v_1 with v_k. Figure 1 shows a digraph with a directed path.

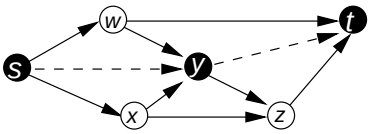

Fig. 1 A digraph with a directed path (induced by the black nodes and the dashed arcs).

A digraph together with a source/sink pair and arc capacities becomes a network (see Figure 2). More formally:

Definition 1. We call $N = (D;s,t;c)$ a *network* if $D = (V,A)$ is a digraph with two specified nodes, a source $s \in V$ with $\delta^-(s) = \emptyset$ and a sink $t \in V$ with $\delta^+(t) = \emptyset$, and arc capacities c_a for all $a \in A$.

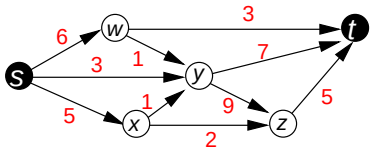

Fig. 2 A network consisting of a digraph with source s, sink t and arc capacities.

Networks are the studied combinatorial structures to model flows therein:

Definition 2. For a network $N = (D;s,t;c)$ with digraph $D = (V,A)$, an (s,t)-*flow* is a function $f : A \to \mathbf{N}_0$ satisfying

- capacity constraints $0 \le f(a) \le c_a$ for all arcs $a \in A$, and
- flow conservation constraints $(\delta f)(v) = \sum_{a\in\delta^-(v)} f(a) - \sum_{a\in\delta^+(v)} f(a) = 0$ for all nodes $v \in V \setminus \{s,t\}$.

We denote by

$$val(f) := \sum_{a\in\delta^-(t)} f(a) = \sum_{a\in\delta^+(s)} f(a)$$

the value of the (s,t)-flow f. For illustration, Figure 3 shows a network with an (s,t)-flow f and its value $val(f)$.

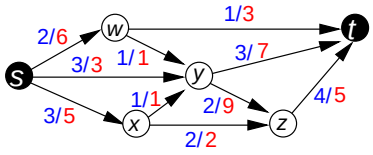

Fig. 3 A network with (s,t)-flow f of value $val(f) = 8$ (on each arc $a \in A$, its flow value and capacity are indicated by $f(a)/c_a$).

This enables us to combinatorially formulate the Network Flow Problem:

Problem 2 (Maximum Network Flow Problem (Combinatorial Formulation)). Given a network $N = (D;s,t;c)$ with digraph $D = (V,A)$, find an (s,t)-flow $f : A \to \mathbf{N}_0$ with maximal value $val(f)$.

The existence of an (s,t)-flow in a given network $N=(D;s,t;c)$ is ensured as soon as there exists an (s,t)-path in the underlying digraph D (which can be easily checked with the help of Breadth First Search techniques starting in s). We will next address the question whether and how we can find an upper bound for its possible value (without knowing the optimum before). For that, we look for the combinatorial structure in a digraph being dual to flows.

Definition 3. Let $N=(D;s,t;c)$ be a network with digraph $D=(V,A)$. An (s,t)-*cut* (V_s,V_t) is a partition $V=V_s\cup V_t$ of V into subsets V_s and $V_t=V\setminus V_s$ with $s\in V_s$ and $t\in V_t$.

The capacity of an (s,t)-cut (V_s,V_t) is

$$c(V_s,V_t)=\sum_{u\in V_s,v\in V_t} c_{(u,v)},$$

see Figure 4 for illustration.

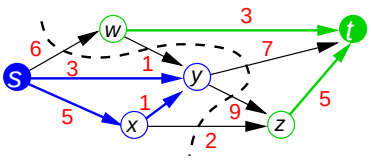

Fig. 4 A network with an (s,t)-cut $V_s=\{s,x,y\}$, $V_t=\{t,w,z\}$ and capacity $c(V_s,V_t)=24$ (as sum of the capacities of all forward arcs crossing the dashed line).

Let $N=(D;s,t;c)$ be a network with digraph $D=(V,A)$ and consider an (s,t)-flow f as well as an (s,t)-cut (V_s,V_t) in N. The flow across the (s,t)-cut (V_s,V_t) is

$$f(V_s,V_t)=\sum_{u\in V_s,v\in V_t} f((u,v))-\sum_{u\in V_s,v\in V_t} f((v,u)).$$

Obviously, $val(f)\leq c(V_s,V_t)$ holds for any (s,t)-cut in a network. We even have:

Theorem 1 (Max-Flow Min-Cut Theorem (Ford & Fulkerson [16])). *For any network $N=(D;s,t;c)$ with digraph $D=(V,A)$ and $s\neq t\in V$, we have*

$$\max\{val(f): f\ (s,t)\text{-flow in } N\}\ =\ \min\{c(V_s,V_t): (V_s,V_t)\ (s,t)\text{-cut in } N\}.$$

The Max-Flow Min-Cut Theorem is one of the fundamental theorems in Combinatorial Optimization. It ensures that the minimum capacity of all (s,t)-cuts in a network always equals the maximum value of an (s,t)-flow. The next question is how to construct such a maximum flow in a network. To state the corresponding combinatorial algorithm, we first have to introduce the following notions.

Definition 4. Let $N=(D;s,t;c)$ be a network with digraph $D=(V,A)$, f an (s,t)-flow, and $P=\{s=v_0,v_1,\ldots,v_k=t\}$ an (undirected) (s,t)-path.

- The residual capacity of an arc a of P is

  $\Delta f(a) = c_a - f(a)$ if $a = (v_i, v_{i+1})$ is a forward arc,
  $\Delta f(a) = f(a)$ if $a = (v_{i+1}, v_i)$ is a backward arc.

- The residual capacity of the path P is

  $\Delta f(P) = \min\{\Delta f(a) : a \text{ arc of } P\}$

  and P is called *f-augmenting path* if $\Delta f(P) > 0$.

Finding f-augmenting paths can be done with Breadth First Search techniques starting in s, where a node u is considered as "neighbor" of the active node v if there is an arc a with $\Delta f(a) > 0$ linking v and u (or u and v), see Figure 5.

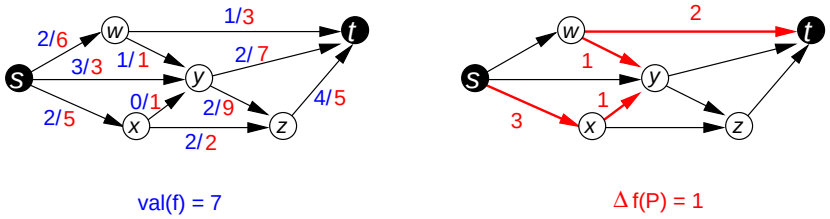

Fig. 5 A network with (s,t)-flow f and augmenting (s,t)-path P with residual capacity $\Delta f(P) = 1$ (resulting as minimum value of the residual capacities of its arcs).

With the help of an f-augmenting path, we can increase the value of f as follows:

Lemma 1. *Let P be an f-augmenting (s,t)-path in a network N with (s,t)-flow f. There exists an (s,t)-flow f' in N with $val(f') = val(f) + \Delta f(P)$. We obtain f' by modifying f on the arcs of P as follows:*

$f'(a) = f(a) + \Delta f(a)$ *for any forward arc a of P,*
$f'(a) = f(a) - \Delta f(a)$ *for any backward arc a of P.*

For illustration, Figure 6 shows f and the resulting flow f' after augmentation using the f-augmenting path from Figure 5.

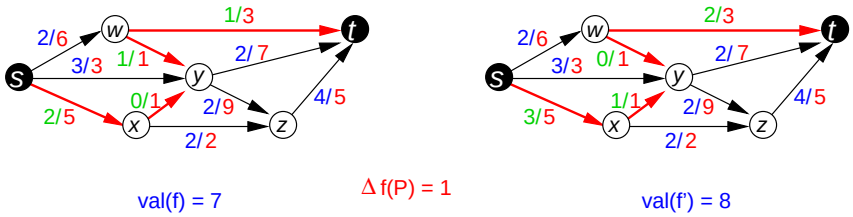

Fig. 6 A network with a (s,t)-flow f and the flow f' obtained by augmentation.

This augmentation can be repeated until no further augmenting path for the current flow can be found. An optimally criterion from [16] guarantees that this leads indeed to the studied maximum flow:

Theorem 2 (Ford & Fulkerson [16]). *An (s,t)-flow f in a network $N = (D;s,t;c)$ has maximal value if and only if there is no f-augmenting (s,t)-path in N.*

Therefore, we arrived at the following combinatorial algorithm for computing maximum flows due to Ford & Fulkerson [16]:

Max-Flow Algorithm (Ford & Fulkerson [16])

Input: Digraph $D = (V,A)$ with arc weights $c \in \mathbf{Z}_+^{|A|}$, source $s \in V$, sink $t \in V$.

Output: Maximum (s,t)-flow f.

STEP 1: Initialize f with $f(a) := 0$ for all arcs $a \in A$.

STEP 2: Find an f-augmenting path P.
IF such a path P exists:
Augment f by
$f(a) := f(a) + \Delta f(a)$ if a is a forward arc of P,
$f(a) := f(a) - \Delta f(a)$ if a is a backward arc of P.
Iterate STEP 2.
ELSE STOP.

Remark.

- The Max-Flow Algorithm by Ford & Fulkerson [16] terminates correctly due to the characterization of maximum flows by augmenting paths (Theorem 2). Note that at this final step, the algorithm finds the shore V_s of an (s,t)-cut (V_s,V_t) such that all arcs outgoing from V_s are saturated as the capacity of this cut equals the value of the current flow which, therefore, cannot be improved further. Hence, the capacity of this (s,t)-cut gives a certificate for the maximality of the obtained flow.
- In the worst case, the algorithm performs $val(f^*)$ augmentation steps using each time an f-augmenting path P with $\Delta f(P) = 1$, where f^* is a maximum flow. Finding an augmenting path and augmenting the flow in STEP 2 takes $O(|V|+|A|)$ time. The *overall running time* of the Max-Flow Algorithm is therefore $O(val(f^*) \cdot (|V|+|A|))$.
- A variant of the Max-Flow Algorithm by Edmonds & Karp [13] determines in STEP 2 an augmenting path of minimal combinatorial length by Breadth First Search techniques. It terminates after $|V| \cdot (|A|+1)$ augmentations and has *polynomial* running time $O(|V| \cdot |A|^2)$.

An example how to perform the Max-Flow Algorithm is presented in Figure 7. More information on network flows can be found in [17, 27].

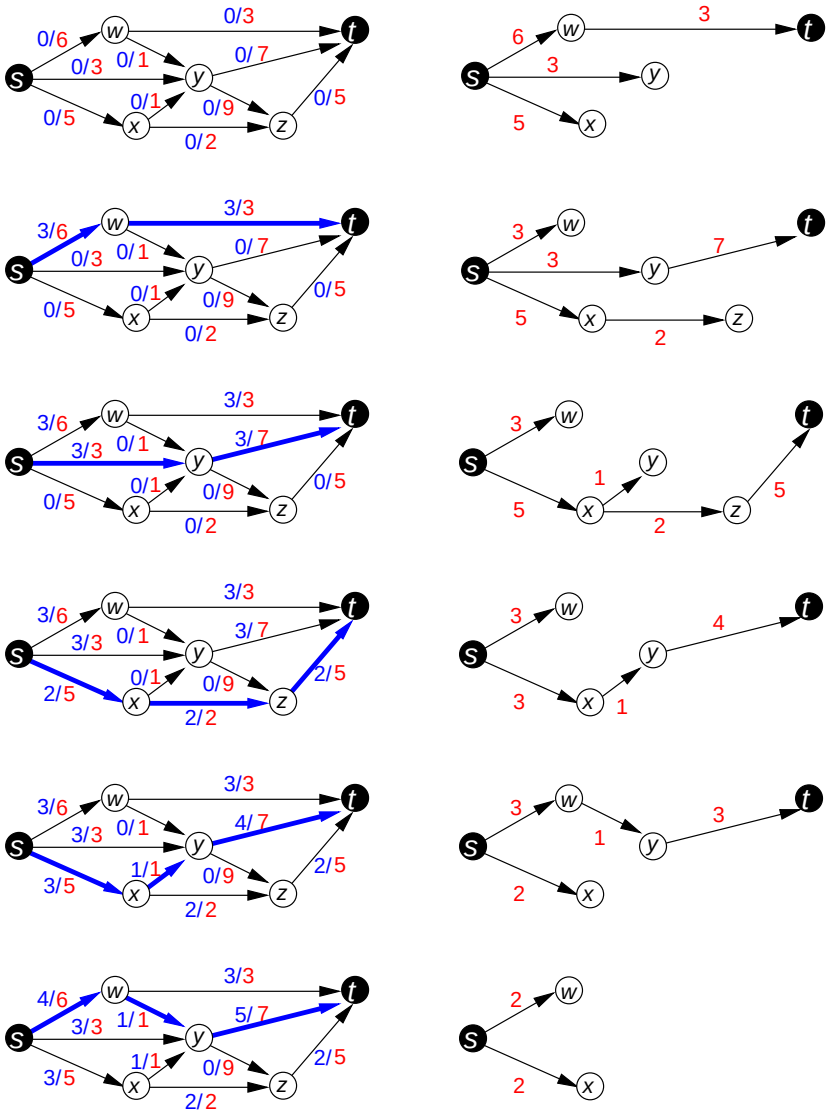

Fig. 7 The Max-Flow Algorithm starts with a flow f with $f(a) := 0$ for all $a \in A$. For each current flow f, a Breadth First Search is performed that, starting in s, adds a node u as neighbor of the active node v if there is an arc a with $\Delta f(a) > 0$ linking v and u (or u and v), until t is reached. This results in a unique f-augmenting path P and f is augmented along P to f'. The procedure is repeated until no augmenting path can be found anymore since the Breadth First Search tree consists in one shore V_s of a (s,t)-cut (V_s, V_t) where all arcs outgoing from V_s are saturated.

3 Solving network flow by linear programming techniques

"From an economic point of view, Linear Programming has been the most important mathematical development in the 20th century." Martin Grötschel

In this section we discuss the following questions about Linear Programming:

- What is a linear program and how it is possible to model a real problem (for instance network flow) as linear program?
- How does the feasible region of a linear program look from a geometric point of view?
- What are the consequences for solution techniques for Linear Programming?

3.1 Modeling a problem as a linear program

We first address the question what a linear program is.

Definition 5. A *linear program (LP)* is as follows:
Maximize/Minimize the value of $\mathbf{c}^T\mathbf{x}$
among all vectors $\mathbf{x} \in \mathbf{R}^n$ satisfying $A\,\mathbf{x} \leq \mathbf{b}$
$\mathbf{x} \geq \mathbf{0}$ (optional)
where $A \in \mathbf{R}^{m\times n}$ is a given constraint matrix, $\mathbf{b} \in \mathbf{R}^m$ a given right hand side vector, and $\mathbf{c} \in \mathbf{R}^n$ a given objective function vector.

We illustrate this formal definition with the help of a small example:

Example 1. This example shows a linear program given explicitly as well as in matrix formulation:

$$\begin{array}{lrll} \max & x_1 + x_2 & & \text{is the linear objective function } \mathbf{c}^T\mathbf{x} \\ s.t. & -x_1 + x_2 & \leq 1 & \\ & x_1 & \leq 3 & \text{form the linear constraints } A\mathbf{x} \leq \mathbf{b} \\ & x_2 & \leq 2 & \\ & x_1, x_2 & \geq 0 & \text{are the non-negativity constraints } \mathbf{x} \geq \mathbf{0} \end{array}$$

Figure 8 gives the graphical interpretation of the constraints and the feasible region, i.e. the set of all feasible solutions $\mathbf{x} \in \mathbf{R}^n_+$ satisfying $A\,\mathbf{x} \leq \mathbf{b}$.

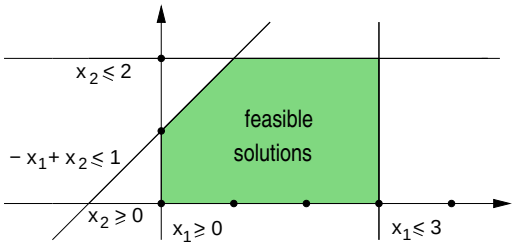

Fig. 8 The graphical interpretation of the constraints and the feasible region (the shaded region) of the linear program given in Example 1.

We next discuss the reformulation of the Network Flow Problem as linear program. Given a network $N = (D; s,t; \mathbf{c})$ with $D = (V,A)$, the problem of finding an (s,t)-flow $f : A \to \mathbf{R}$ maximizing the value $val(f)$ can be encoded as follows:

- the required variables are x_a to express the flow $f(a)$ on each arc $a \in A$;
- the objective function is max $\sum_{a \in \delta^+(s)} x_a$ to maximize the flow leaving the source s (or, equivalently, max $\sum_{a \in \delta^-(t)} x_a$ as flow entering the sink t);
- the flow conservation constraints read as $\sum_{a \in \delta^-(v)} x_a = \sum_{a \in \delta^+(v)} x_a \ \ \forall v \in V \setminus \{s,t\}$;
- the capacity constraints lead to $x_a \leq c_a \ \ \forall a \in A$;
- in addition, non-negativity $x_a \geq 0 \ \ \forall a \in A$ is required for all variables.

Thus, the Maximum Network Flow Problem of finding an (s,t)-flow $f : A \to \mathbf{R}$ maximizing the value $val(f)$ reads as linear program:

Problem 3 (Maximum Network Flow Problem (LP Formulation)). Given a network $N = (D; s,t; c)$ with digraph $D = (V,A)$, solve the following linear program:

$$\begin{array}{rll} \max & \sum_{a \in \delta^+(s)} x_a & \\ s.t. & \sum_{a \in \delta^-(v)} x_a = \sum_{a \in \delta^+(v)} x_a & \forall v \in V \setminus \{s,t\} \\ & x_a \leq c_a & \forall a \in A \\ & x_a \geq 0 & \forall a \in A \end{array}$$

Indeed, every vector $\mathbf{x} \in \mathbf{R}^A$ satisfying all the above constraints corresponds to a valid (s,t)-flow f, an optimal solution of this linear program corresponds to a maximal flow.

Example 2. The Maximum Network Flow Problem with the network from Figure 2 reads as explicit linear program:

$$\begin{array}{rrrrrrrrrrrrr} \max & x_{sw} & +x_{sx} & +x_{sy} & & & & & & & & & \\ s.t. & x_{sw} & & & -x_{wt} & -x_{wy} & & & & & & = & 0 \\ & & x_{sx} & & & & -x_{xy} & -x_{xz} & & & & = & 0 \\ & & & x_{sy} & & +x_{wy} & +x_{xy} & & -x_{yt} & -x_{yz} & & = & 0 \\ & & & & & & & x_{xz} & & +x_{yz} & -x_{zt} & = & 0 \\ & x_{sw} & & & & & & & & & & \leq & 6 \\ & & x_{sx} & & & & & & & & & \leq & 5 \\ & & & x_{sy} & & & & & & & & \leq & 3 \\ & & & & x_{wt} & & & & & & & \leq & 3 \\ & & & & & x_{wy} & & & & & & \leq & 1 \\ & & & & & & x_{xy} & & & & & \leq & 1 \\ & & & & & & & x_{xz} & & & & \leq & 2 \\ & & & & & & & & x_{yt} & & & \leq & 7 \\ & & & & & & & & & x_{yz} & & \leq & 9 \\ & & & & & & & & & & x_{zt} & \leq & 5 \\ & x_{sw}, & x_{sx}, & x_{sy}, & x_{wt}, & x_{wy}, & x_{xy}, & x_{xz}, & x_{yt}, & x_{yz}, & x_{zt}, & \geq & 0 \end{array}$$

3.2 Geometry of the feasible region

For a given linear program

$$\max \mathbf{c}^T \mathbf{x} \; s.t. \; A\mathbf{x} \leq \mathbf{b}, \; \mathbf{x} \geq \mathbf{0}$$

the task is to find one vector $\mathbf{x}$ maximizing the objective function value within the feasible region described by the constraint system $A\mathbf{x} \leq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$. In general, a linear program can have the following sets of optimal solutions: a unique optimum, infinitely many optima, or no optimal solutions at all due to infeasibility or unboundedness, see Figure 9.

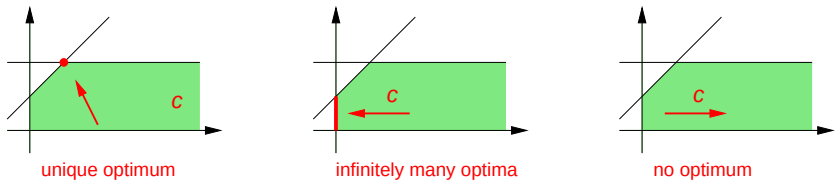

Fig. 9 The different situations for sets of optimal solutions of a feasible linear program: a unique optimum, infinitely many optima, or no optimal solution due to unboundedness (in all cases, the feasible region of the linear program is shaded and the arrows indicate the direction of the objective function vector).

In particular, whenever an optimal solution exist for a linear program, it is attained at the boundary of its feasible region. This is a central issue for Linear Programming (see, e.g. [26] for a proof):

Theorem 3 (Linear Programming Theorem). *If a linear program has a (bounded) optimal solution, then there exists an "extremal" point on the boundary of the feasible region which is optimal.*

Hence, as a first step towards finding an optimal solution, we shall describe the feasible region of a linear program more formally and study its boundary (in particular the extremal points). For that, we need to introduce the following notations.

Let $\mathbf{x}^1, \ldots, \mathbf{x}^k \in \mathbf{R}^n$ be points and $\lambda_1, \ldots, \lambda_k \in \mathbf{R}_+$ with $\sum_{i \leq k} \lambda_i = 1$. The point $\mathbf{x} = \sum_{i \leq k} \lambda_i \mathbf{x}^i \in \mathbf{R}^n$ is a *convex combination* of $\mathbf{x}^1, \ldots, \mathbf{x}^k$. A set $C \subseteq \mathbf{R}^n$ is *convex* if for any two points $\mathbf{x}, \mathbf{x}' \in C$, also any of their convex combinations

$$\lambda \mathbf{x} + (1-\lambda)\mathbf{x}', \lambda \in (0,1)$$

belongs to C. For a subset $D \subseteq \mathbf{R}^n$, its *convex hull* $\mathrm{conv}(D)$ consists of all points in $\mathbf{R}^n$ being a convex combination of points in D.

A subset $C^0 \subseteq C$ of a convex set $C \subseteq \mathbf{R}^n$ is an *extremal set* if C^0 is convex, for all $\mathbf{x}, \mathbf{x}' \in C$ and $\lambda \in (0,1)$ with $\lambda \mathbf{x} + (1-\lambda)\mathbf{x}' \in C^0$, we have $\mathbf{x}, \mathbf{x}' \in C^0$. Note that the empty set and C itself are trivial extremal sets of C. Special extremal sets are *extreme*

points in C which cannot be obtained as proper convex combination of some other points in C, see Figure 10 for examples.

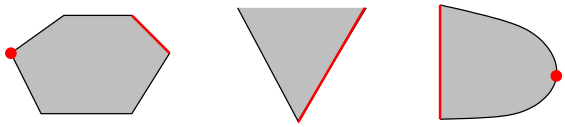

Fig. 10 Extremal sets of convex sets.

It turns out that the feasible regions of linear programs are special convex sets: For $\mathbf{a} \in \mathbf{R}^n$ and $b \in \mathbf{R}$, the set

- $\left\{\mathbf{x} \in \mathbf{R}^n : \mathbf{a}^T\mathbf{x} = b\right\}$ is a *hyperplane* of $\mathbf{R}^n$,
- $\left\{\mathbf{x} \in \mathbf{R}^n : \mathbf{a}^T\mathbf{x} \leq b\right\}$ is a *closed half-space* of $\mathbf{R}^n$.

A *polyhedron* $P \subseteq \mathbf{R}^n$ is the intersection of finitely many closed half-spaces and/or hyperplanes in $\mathbf{R}^n$. A bounded polyhedron is called *polytope*.

Every polyhedron is a convex set, as hyperplanes and half-spaces are convex, and the intersection of convex sets yields a convex set again.

The dimension $\dim(P)$ of a polyhedron $P \subseteq \mathbf{R}^n$ is the smallest dimension of an affine subspace containing P, or the largest d for which P contains points $\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^d$ s.t. the vectors $\mathbf{x}^0 - \mathbf{x}^1, \ldots, \mathbf{x}^0 - \mathbf{x}^d$ are linearly independent.

The extremal sets of a polyhedron P are called *faces*, and in particular faces of dimension

- 0 are extreme points,
- 1 are edges,
- $\dim(P) - 1$ are facets.

Figure 11 illustrates different faces of a polytope.

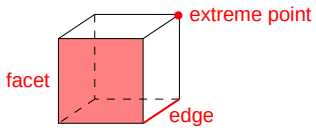

Fig. 11 A polytope and different extremal sets (of dimension 0, 1 and 2).

A bounded polyhedron, i.e. a polytope, has besides its description as intersection of finitely many closed half-spaces and/or hyperplanes a second representation [24, 28]:

Theorem 4 (Weyl-Minkowski Theorem). *A bounded polyhedron is the convex hull of its extreme points.*

For $A \in \mathbf{R}^{m\times n}$ and $\mathbf{b} \in \mathbf{R}^m$ as constraint matrix and right hand side vector, let

$$P(A,\mathbf{b}) = \{\mathbf{x} \in \mathbf{R}^n : A\mathbf{x} \leq \mathbf{b}\}$$

denote the polyhedron defined by the corresponding half-spaces $A_{i\cdot} \leq b_i$ or hyperplanes $A_{j\cdot} = b_j$. We can characterize its extreme points as follows (see, e.g. [26] for a proof):

Theorem 5. *For a polyhedron $P = P(A,\mathbf{b}) \subseteq \mathbf{R}^n$ and $\mathbf{x}^\star \in P$, the following assertions are equivalent:*

- *$\mathbf{x}^\star$ is an extreme point of P;*
- *$\{\mathbf{x}^\star\}$ is a 0-dimensional face of P;*
- *$\mathbf{x}^\star$ is not a convex combination of other points in P;*
- *$P \setminus \{\mathbf{x}^\star\}$ is still convex;*
- *$\exists \mathbf{c} \in \mathbf{R}^n \setminus \{\mathbf{0}\}$ s.t. $\mathbf{x}^\star$ is unique optimum of $\max \mathbf{c}^T\mathbf{x}$, $\mathbf{x} \in P$.*

The drawback of the above characterization is that none of the conditions characterizing $\mathbf{x}^\star$ as an extreme point is easy to check. This changes in the special case where the studied polyhedron is given by hyperplanes only. For $A \in \mathbf{R}^{m\times n}$ and $\mathbf{b} \in \mathbf{R}^m$, let

$$P^{=}(A,\mathbf{b}) = \{\mathbf{x} \in \mathbf{R}^n : A\mathbf{x} = \mathbf{b}\}.$$

Then we have the following (see, e.g. [26] for a proof):

Theorem 6. *For a polyhedron $P = P^{=}(A,\mathbf{b}) \subseteq \mathbf{R}^n$ and $\mathbf{x}^\star \in P$, the following assertions are equivalent:*

- *$\mathbf{x}^\star$ is an extreme point of P;*
- *The columns $A_{\cdot j}$ of A with $j \in supp(\mathbf{x}^\star)$ are linearly independent.*

As extreme points of the feasible region P of a linear program are crucial and can be easily detected if P is of the special form $P^{=}(A,\mathbf{b})$, we consider linear programs given in the so-called *equational form*:

$$\begin{aligned} \max \mathbf{c}^T\mathbf{x} \;\; s.t. \;\; A\mathbf{x} &= \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned}$$

Remark:

- Linear programs in equational form are also called linear programs given in standard form.
- Note that any linear program can be transformed into equational form, namely, by introducing so-called *slack variables* $\mathbf{y} \in \mathbf{R}^m$:

$$\begin{aligned} \max \mathbf{c}^T\mathbf{x} \;\; s.t. \;\; A\mathbf{x} &\leq \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned} \Rightarrow \begin{aligned} \max \mathbf{c}^T\mathbf{x} \;\; s.t. \;\; A\mathbf{x} + \mathbf{y} &= \mathbf{b} \\ \mathbf{x},\;\; \mathbf{y} &\geq \mathbf{0} \end{aligned}$$

- For linear programs in equational form, we assume that the equation system $A\mathbf{x} = \mathbf{b}$ has at least one solution (i.e. that $P^{=}(A,\mathbf{b}) \neq \emptyset$ holds), and that the rows of the matrix A are linearly independent (i.e. no redundant constraints occur).

We are interested in special feasible solution of a linear program:

Definition 6. A *basic feasible solution* of the linear program in equational form

$$\begin{aligned} \max \mathbf{c}^T\mathbf{x}\ s.t.\ A\mathbf{x} &= \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned}$$

with $A \in \mathbf{R}^{m\times n}, \mathbf{b} \in \mathbf{R}^m$ is a feasible solution $\mathbf{x}^\star \in \mathbf{R}^n$ for which there exists an m-element subset $B \subseteq \{1,\ldots,n\}\ s.t.$ the (square) matrix A_B is non-singular (i.e., the columns of A indexed by B are linearly independent), and $x_j^\star = 0$ for all $j \notin B$.

Example 3. The vector $\mathbf{x}^\star = (0,2,0,1,0)$ is a basic feasible solution of the equation system

$$\begin{aligned} x_1 + 5x_2 + 3x_3 + 4x_4 + 6x_5 &= 14 \\ x_2 + 3x_3 + 5x_4 + 6x_5 &= 7 \end{aligned}$$

with $B = \{2,4\}$.

In fact, basic feasible solutions are crucial for Linear Programming due to the following reason:

Theorem 7. *Consider a linear program in equational form:*

$$\max \mathbf{c}^T\mathbf{x}\ s.t.\ A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}.$$

- *If there is at least one feasible solution and the objective function is bounded from above on $P^=(A,\mathbf{b}) \cap \mathbf{R}^n_+$, then there always exists an optimal solution.*
- *If an optimal solution exists, then there is also a* basic *feasible solution which is optimal.*

In addition, basic feasible solutions are easy to detect:

Theorem 8. *A feasible solution* $\mathbf{x}$ *of a linear program* $\max \mathbf{c}^T\mathbf{x}\ s.t.\ A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$ *is* basic *if and only if the columns of the matrix A_K are linearly independent, where*

$$K = \left\{ j \in \{1,\ldots,n\} : x_j > 0 \right\}.$$

This opens the possibility to solve linear programs with the help of basic feasible solutions.

A rather naive approach to solve linear programs would be: For a given linear program $\max \mathbf{c}^T\mathbf{x}\ s.t.\ A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$,

- find all extreme points of $P^=(A,\mathbf{b})$, i.e., all basic feasible solutions (there are at most $\binom{m}{n}$ if $A \in \mathbf{R}^{m\times n}$).
- select the best one among them (i.e. this $\mathbf{x}$ with $\mathbf{c}^T\mathbf{x}$ maximal).

Is there a more clever idea to solve linear programs?

3.3 The Simplex Method for solving linear programs

Given a matrix $A \in \mathbf{R}^{m\times n}$ and vectors $\mathbf{b} \in \mathbf{R}^m, \mathbf{c} \in \mathbf{R}^n$, consider the linear program

$$\begin{aligned} \max \mathbf{c}^T\mathbf{x}\ s.t.\ A\mathbf{x} &\leq \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0}. \end{aligned}$$

To solve the linear program with the help of the Simplex Method, one takes advantage of the following previously stated results: If a linear program has a bounded optimal solution, then there exists an *extreme point* on the boundary of the feasible region which is optimal (Main Theorem of Linear Programming). For a linear program given in *equational form*

$$\max \mathbf{c}^T\mathbf{x}\ s.t.\ A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$$

we have even more:

- If $P^{=}(A,\mathbf{b})$ is non-empty and bounded, there *exists* always an optimal solution.
- Among all optimal solutions, there is always a *basic* feasible solution.
- Basic feasible solutions are easy to detect: A feasible solution $\mathbf{x}$ is *basic* if and only if the columns of the matrix A_B are linearly independent, where

$$B = \left\{ j \in \{1,\ldots,n\} : x_j > 0 \right\}.$$

The idea of the Simplex Method is to start with an arbitrary basic feasible solution and, as long as the current solution is not optimal, to move to a "neighbored" basic feasible solution with a better objective function value.

We first shall illustrate this method with the help of an introductory example (the linear program from Example 1) before stating it formally.

Example 4. Given the following linear program:

$$\begin{array}{lrl} \max & x_1 + x_2 & \\ s.t. & -x_1 + x_2 & \leq 1 \\ & x_1 & \leq 3 \\ & x_2 & \leq 2 \\ & x_1, x_2 & \geq 0 \end{array}$$

As the linear program is not in equational form, we have to transform it by introducing *slack variables* in order to turn the inequalities into equations. The resulting equational form of the above linear program (with slack variables in bold) is:

$$\begin{array}{lrcl} \max & x_1 + x_2 & & \\ s.t. & -x_1 + x_2 + \mathbf{x_3} & = & 1 \\ & x_1 + \mathbf{x_4} & = & 3 \\ & x_2 + \mathbf{x_5} & = & 2 \\ & x_1\ ,\ x_2\ ,\ \mathbf{x_3}\ ,\ \mathbf{x_4}\ ,\ \mathbf{x_5} & \geq & 0 \end{array}$$

From the linear program in equational form, we easily get $\mathbf{x}^0 = (0,0,1,3,2)^T$ as *initial basic feasible solution* by taking the slack variables as basis $B^0 = \{3,4,5\}$ and the original variables as non-basis $N^0 = \{1,2\}$.

We next rewrite the linear program as so-called *simplex tableau*, having the basic variables as left hand side (in bold) and an additional row for the objective function value $z = \mathbf{c}^T\mathbf{x}$:

$$\begin{array}{rl} \mathbf{x_3} = & 1 + x_1 - x_2 \\ \mathbf{x_4} = & 3 - x_1 \\ \mathbf{x_5} = & 2 \qquad - x_2 \\ \hline z = & \qquad x_1 + x_2 \end{array}$$

Considering the simplex tableau associated with $\mathbf{x}^0 = (0,0,1,3,2)^T$, we obviously have $z = 0$ as objective function value.

In order to improve z, we can increase the value of x_1 or x_2, w.l.o.g. say x_2 (keeping $x_1 = 0$). How much depends on the tableau and the non-negativity constraints: from $x_3 = 1 + x_1 - x_2$, $x_1, x_2, x_3 \geq 0$ we infer $x_2 \leq 1$, from $x_5 = 2 - x_2$ and $x_2, x_5 \geq 0$ we infer $x_2 \leq 2$. Together, we conclude that $x_2 = 1$ is possible.

We update the tableau accordingly by rewriting the first row (to have x_2 as left hand side) and substituting this expression for x_2 in the other rows. The resulting tableau (with changes in bold) is

$$\begin{array}{rl} \mathbf{x_2} = & \mathbf{1} + \quad x_1 - \mathbf{x_3} \\ x_4 = & 3 - \quad x_1 \\ x_5 = & \mathbf{1} - \quad \mathbf{x_1} + \mathbf{x_3} \\ \hline z = & \mathbf{1} + \mathbf{2x_1} - \mathbf{x_3} \end{array}$$

associated with the basic feasible solution $\mathbf{x}^1 = (0,1,0,3,1)^T$, $B^1 = \{2,4,5\}$ and with objective function value $z = 1$.

Improving z further is possible by increasing the value of x_1 only (as increasing x_3 would decrease z).

From the tableau and non-negativity constraints we see that no restriction comes from $x_2 = 1 + x_1 - x_3$, the second row $x_4 = 3 - x_1$ and $x_1, x_4 \geq 0$ show $x_1 \leq 3$, but $x_5 = 1 - x_1 + x_3$ and $x_1, x_3, x_5 \geq 0$ result in $x_1 \leq 1$. Hence, $x_1 = 1$ is possible.

We update the tableau accordingly by rewriting the third row (to have x_1 as left hand side) and substituting this expression for x_1 in the other rows. We get the new tableau (with changes in bold)

$$\begin{array}{rl} x_2 = & \mathbf{2} - \quad \mathbf{x_5} \\ x_4 = & \mathbf{2} + \quad \mathbf{x_5} - \mathbf{x_3} \\ \mathbf{x_1} = & 1 - \quad \mathbf{x_5} + x_3 \\ \hline z = & \mathbf{3} - \mathbf{2x_5} + \mathbf{x_3} \end{array}$$

associated with $\mathbf{x}^2 = (1,2,0,2,0)^T$, $B^2 = \{1,2,4\}$ and $z = 3$.

Now, improving z is possible only by increasing the value of x_3 (as increasing x_5 would decrease z). From the tableau and non-negativity we see that $x_4 = 2 + x_5 - x_3$

and $x_3, x_4, x_5 \geq 0$ result in $x_3 \leq 2$, while the row $x_1 = 1 - x_5 + x_3$ does not restrict the value of x_3. Hence, $x_3 = 2$ is possible.

We update the tableau accordingly by rewriting the second row (to have x_3 as left hand side) and substituting this expression for x_3 in the other rows. The resulting tableau (again with changes in bold) is

$$\begin{aligned} x_2 &= 2 - x_5 \\ \mathbf{x_3} &= 2 + x_5 - \mathbf{x_4} \\ x_1 &= \mathbf{3} + \mathbf{0} - \mathbf{x_4} \\ \hline z &= \mathbf{5} - \mathbf{x_5} - \mathbf{x_4} \end{aligned}$$

associated with $\mathbf{x}^3 = (3,2,2,0,0)^T$, $B^3 = \{1,2,3\}$ and $z = 5$. In this situation, we cannot increase a non-basic variable further without decreasing z (as x_5 and x_4 appear with negative signs).

So, we are stuck. But $\mathbf{x}^3$ is the *optimal solution*: Any feasible solution $\tilde{\mathbf{x}}$ with $\mathbf{c}^T\tilde{\mathbf{x}} = \tilde{z}$ has to satisfy

$$\tilde{z} = 5 - \tilde{x}_5 - \tilde{x}_4$$

which implies $\tilde{z} \leq 5$ (together with non-negativity). Hence, $\mathbf{x}^3$ is optimal!

In fact, $\mathbf{x}^3$ is the unique optimal solution (as $z = 5$ requires $x_4 = x_5 = 0$ and the equations uniquely determine the values of x_1, x_2 and x_3).

The geometric interpretation is as follows (see Figure 12): Starting with the initial basic feasible solution $\mathbf{x}^0 = (0,0)$ (in the original variables only), the simplex method moves along the edges of the feasible region from one basic feasible solution to another, while the objective function value grows until it reaches the optimum.

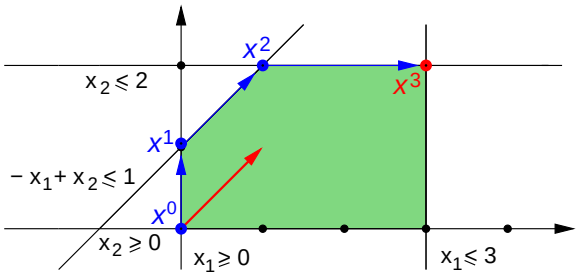

Fig. 12 The geometric interpretation of the basis exchanges performed in Example 4.

The previous example illustrated the solution method for linear programs found by Dantzig [7] (see also [8, 9]), now we state it formally:

The Simplex Method (Dantzig [7])
Input: a matrix $A \in \mathbf{R}^{m \times n}$ and vectors $\mathbf{b} \in \mathbf{R}^m, \mathbf{c} \in \mathbf{R}^n$, defining a linear program $\max \mathbf{c}^T\mathbf{x}$ *s.t.* $A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$

Output: a vector $\mathbf{x}^*$ maximizing the objective function

1. Transform the program into equational form (if necessary).
2. Find an initial basic feasible solution $x^0 \in \mathbf{R}^n$ and the corresponding basis $B^0 \subseteq \{1,\ldots,n\}$ *s.t.* A_{B^0} is non-singular and $x^0_j = 0 \forall j \notin B^0$.
   Generate the corresponding simplex tableau $T(B^0)$.
3. Move from one basic feasible solution x^i with basis B^i to a basic feasible solution x^{i+1} with basis B^{i+1} and higher objective function value by selecting $j \in B^i$ and $\ell \in \{1,\ldots,n\} \setminus B^i$ and setting $B^{i+1} := B^i \setminus \{j\} \cup \{\ell\}$ s.t. $c(x^{i+1}) \geq c(x^i)$ holds.
4. Stop if no further improvement is possible.

We will next discuss all the necessary steps of the Simplex Method in detail.

STEP 1 (Transformation). As we need linear programs given in equational form

$$\max \mathbf{c}^T\mathbf{x} \; s.t. \; A\mathbf{x} \; = \; \mathbf{b}, \mathbf{x} \; \geq \mathbf{0},$$

inequalities and variables without sign restrictions are disturbing and the following transformation becomes necessary: If the given (in)equality system has a

- row $A_{i\cdot}\mathbf{x} \; \leq \; b_i$, introduce a *slack variable* $x_{n+i} \geq 0$ and replace the row by

$$A_{i\cdot}\mathbf{x} \; + \; x_{n+i} = \; b_i$$

- row $A_{j\cdot}\mathbf{x} \; \geq \; b_j$, introduce a *slack variable* $x_{n+j} \geq 0$ and replace the row by

$$-A_{j\cdot}\mathbf{x} \; + \; x_{n+j} = - \; b_j$$

- variable x_ℓ without sign restriction, introduce two new variables $y_\ell \geq 0$ and $z_\ell \geq 0$, and substitute x_ℓ everywhere by $y_\ell - z_\ell$.

After applying an according transformation, the original linear program is in equational form, as required for the next step.

STEP 2 (Initial basic feasible solution). Consider a linear program in equational form. We distinguish the following two cases.

If the original linear program was given in inequality form $\max \mathbf{c}^T\mathbf{x}$ *s.t.* $A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$, then the transformation in STEP 1 into equational form with the help of slack variables $x_{n+1},\ldots,x_{n+m}$ yields

$$\begin{aligned} \max \mathbf{c}^T\mathbf{x} \; s.t. \; \overline{A}\overline{\mathbf{x}} &= \mathbf{b} \\ \overline{\mathbf{x}} &\geq \mathbf{0} \end{aligned}$$

with $\overline{A} = (A, I)$ and $\overline{\mathbf{x}} = (x_1,\ldots,x_n,x_{n+1},\ldots,x_{n+m})$. By the structure of $\overline{A}$, an obvious basic feasible solution of the transformed linear program is

$$\mathbf{x}^0 = \begin{pmatrix} \mathbf{0} \\ \mathbf{b} \end{pmatrix}$$

with basis $B^0 = \{x_{n+1}, \ldots, x_{n+m}\}$ (i.e. containing all slack variables).

If the linear program is already given in equational form $\max \mathbf{c}^T\mathbf{x}$ *s.t.* $A\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$, there is no obvious initial basic feasible solution (as $\mathbf{x} = \mathbf{0}$ is infeasible if $\mathbf{b} \neq \mathbf{0}$). For each row of $A\mathbf{x} = \mathbf{b}$, we introduce an *auxiliary variable* $x_{n+i} = b_i - A_{i\cdot}^T\mathbf{x}$ and find values for $x_1, \ldots, x_n$ *s.t.* $x_{n+i} = 0$ holds for all $1 \leq i \leq m$ by solving the auxiliary linear program ALP

$$\max - \sum_{i \leq m} x_{n+i} \; s.t. \; \overline{A}\overline{\mathbf{x}} = \mathbf{b}, \overline{\mathbf{x}} \geq \mathbf{0}$$

with $\overline{A} = (A, I)$ and $\overline{\mathbf{x}} = (x_1, \ldots, x_n, \ldots, x_{n+1}, \ldots, x_{n+m})$ if $\mathbf{b} \geq \mathbf{0}$ (otherwise, we multiply the equations with $b_i < 0$ by -1). This works, since we have:

Lemma 2. *The original linear program is feasible if and only if every optimal solution* $\overline{\mathbf{x}}$ *of ALP satisfies* $x_{n+1} = \ldots = x_{n+m} = 0$. *For any such optimal solution, its basic vector* $\overline{\mathbf{x}}_B = (x_1 \ldots, x_n)$ *is a basic feasible solution of the original linear program.*

The *simplex tableau* $T(B^0)$ determined by B^0 is a system of $m+1$ linear equations in variables $x_1, \ldots, x_n$ and z that has the same set of solutions as the original system $A\mathbf{x} = \mathbf{b}$, $z = \mathbf{c}^T\mathbf{x}$. In matrix notation, $T(B^0)$ reads as

$$\begin{aligned} \mathbf{x}_{B^0} &= \overline{\mathbf{b}} - \overline{A}\mathbf{x}_N \\ z &= z_0 + \overline{\mathbf{c}}^T\mathbf{x}_N \end{aligned}$$

where $\mathbf{x}_{B^0}$ is the vector of basic variables, $\mathbf{x}_N$ the vector of non-basic variables and $N = \{1, \ldots, n\} \setminus B^0$, and $\overline{\mathbf{b}} \in \mathbf{R}^m$, $\overline{\mathbf{c}} \in \mathbf{R}^{n-m}$, $\overline{A} \in \mathbf{R}^{m \times (n-m)}$, $z_0 \in \mathbf{R}$.

This always works, since we have in general:

Lemma 3. *For each feasible basis B, there exists exactly one simplex tableau* $T(B)$

$$\begin{aligned} \mathbf{x}_B &= \overline{\mathbf{b}} - \overline{A}\mathbf{x}_N \\ z &= z_0 - \overline{\mathbf{c}}^T\mathbf{x}_N \end{aligned}$$

with $\overline{A} = A_B^{-1}A_N$, $\overline{\mathbf{b}} = A_B^{-1}\mathbf{b}$, $\overline{\mathbf{c}} = \mathbf{c}_N - (\mathbf{c}_B^T A_B^{-1} A_B)^T$ *and* $z_0 = \mathbf{c}_B^T A_B^{-1}\mathbf{b}$.

For the *initial* basic feasible solution $\mathbf{x}^0$, we often have $A_{B^0} = I$ which simplifies the construction of the first tableau by

$$\overline{A} = A_N, \overline{\mathbf{b}} = \mathbf{b}, \overline{\mathbf{c}} = \mathbf{c}_N - (\mathbf{c}_B^T A_N)^T \text{ and } z_0 = \mathbf{c}_B^T\mathbf{b}.$$

Note that from any tableau $T(B)$, we can read off immediately the basic feasible solution $\mathbf{x}^0$ by

$$x_i^0 = \overline{b}_i \; \forall i \in B \text{ and } x_i^0 = 0 \; \forall i \in N,$$

and the objective function value $\mathbf{c}^T\mathbf{x}^0 = z^0 = z^0 + \overline{\mathbf{c}}^T\mathbf{0}$.

STEP 3 (Basis exchanges). In each basis exchange (called *pivot step*) of the Simplex Method, we go from the current basis B and its tableau $T(B)$ to a new basis B' and its tableau $T(B')$. Thereby, a nonbasic variable x_ℓ with $\ell \in N = \{1,\ldots,n\} \setminus B$ has to be exchanged by a basic variable x_k with $k \in B$ in order to obtain the new basis

$$B' = (B \setminus \{k\}) \cup \{\ell\}.$$

We say that x_k *leaves* the basis and x_ℓ *enters* the basis. This leads to the following questions:

- Which conditions have x_k and x_ℓ to satisfy?
- How to select them if there is no unique choice?
- How to obtain the new tableau $T(B')$?

We first discuss the conditions for entering and leaving variables. A nonbasic variable x_ℓ with $\ell \in N$ may enter the basis if and only if its coefficient $\overline{c}_\ell$ in the last row of the tableau $T(B)$

$$\begin{aligned} \mathbf{x}_B &= \overline{\mathbf{b}} \; - \overline{A}\,\mathbf{x}_N \\ z \;\; &= z_0 + \overline{\mathbf{c}}^T\mathbf{x}_N \end{aligned}$$

is *positive*, i.e., if $\overline{c}_\ell^T > 0$ holds (as only incrementing such non-basic variables can increase the value z of the objective function). For chosen x_ℓ with $\ell \in N$, the leaving basic variable must correspond to a row of the tableau which limits the increment of x_ℓ *most strictly*:

- All nonbasic variables x_i with $i \in N \setminus \{\ell\}$ should remain zero, hence the j-th row of the tableau together with non-negativity yields

$$x_j = \overline{b}_j - \overline{a}_{j\ell}x_\ell \geq 0.$$

- If $\overline{a}_{j\ell} \leq 0$, this inequality does not restrict the increase of x_ℓ in any way.
- For any $\overline{a}_{j\ell} > 0$, we have $x_\ell \leq \frac{\overline{b}_j}{\overline{a}_{j\ell}}$.

Thus, we can choose x_k with $\overline{a}_{k\ell} > 0$ and $\frac{\overline{b}_k}{\overline{a}_{k\ell}}$ minimal.

This leads to the following fundamental theorem which in addition shows how to detect two exceptional cases: *unboundedness* (i.e. the case where the linear program does not have a finite optimal solution) and *degeneracy* (i.e. the case where *several* bases correspond to a *single* basic feasible solution). In degenerate basic feasible solutions, some basic variables are zero: e.g., for the basic feasible solution $\mathbf{x}^0 = (0,0,0,2)^T$, the following bases

$$B = \{1,4\} \quad \text{or} \quad B' = \{2,4\} \quad \text{or} \quad B'' = \{3,4\}$$

are possible.

Theorem 9 (Basis Exchange Theorem). *Let* $\mathbf{x}$ *be a basic feasible solution with basis* B *and simplex tableau* $T(B)$

$$\begin{aligned} \mathbf{x}_B &= \overline{\mathbf{b}} - \overline{A}\,\mathbf{x}_N \\ z &= z_0 + \overline{\mathbf{c}}^T \mathbf{x}_N \end{aligned}$$

and let $\ell \in N$ with $\overline{c}_\ell > 0$. Then we have the following:

- *If $\overline{A}_{\cdot\ell} \leq 0$, then the linear program is unbounded.*
- *If $\overline{A}_{\cdot\ell} \not\leq 0$, we get a new basis $B' = (B \setminus \{k\}) \cup \{\ell\}$ where $k \in B$ with $\overline{a}_{k\ell} > 0$ and*

$$\frac{\overline{b}_k}{\overline{a}_{k\ell}} = \min\left\{ \frac{\overline{b_j}}{\overline{a}_{j\ell}} : j \in B, \overline{a}_{j\ell} > 0 \right\}.$$

- *If B is non-degenerate (as $\mathbf{x}_B = \overline{\mathbf{b}} > \mathbf{0}$), then $\mathbf{c}^T\mathbf{x}' > \mathbf{c}^T\mathbf{x}$ holds where $\mathbf{x}'$ is the basic feasible solution associated with the new basis B'.*

Remark. The geometric view may illustrate the basis exchanges. Basic feasible solutions correspond to extreme points of the polyhedron $P^{=}(A, \mathbf{b})$. Pivot steps (i.e. basis exchanges) of the Simplex Method move from one extreme point to another along an edge (i.e. an 1-dimensional face) of the polyhedron:

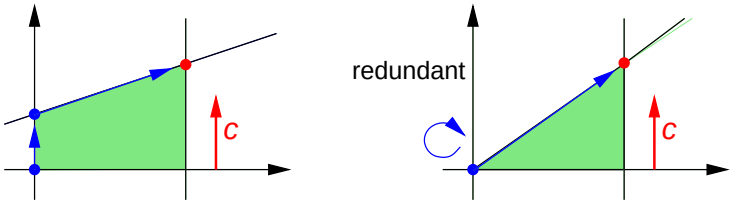

Fig. 13 Basis exchanges in the non-degenerate and in the degenerate case.

Exceptions are *degenerate* pivot-steps, where we stay at the same extreme point $\mathbf{x}^0$ as only the feasible basis changes. Possible reasons are superfluous variables or redundant inequalities (whose removal resolves degeneracy) or geometric reasons (e.g. that more than $\dim(P^{=}(A, \mathbf{b}))$ hyperplanes meet in $\mathbf{x}^0$). The resulting difficulty is so-called *cycling*:

- If degeneracy occurs, longer runs of degenerate bases exchanges (without improvement in the objective function value) may be necessary.
- It may even happen that some tableau is *repeated* in a sequence of degenerate exchange steps (called cycling) s.t. the algorithm passes through an *infinite* sequence of tableaux and, thus, fails.

To finish a basis exchange, updating the simplex tableau according to the new basis is required. For the new basis B', one can calculate the new tableau $T(B')$

$$\begin{aligned} \mathbf{x}_{B'} &= \overline{\mathbf{b}} - \overline{A}\,\mathbf{x}_{N'} \\ z &= z_0 + \overline{\mathbf{c}}^T \mathbf{x}_{N'} \end{aligned}$$

by $\overline{A} = A_{B'}^{-1} A_{N'}$, $\overline{\mathbf{b}} = A_{B'}^{-1}\mathbf{b}$, $\overline{\mathbf{c}} = \mathbf{c}_{N'} - (\mathbf{c}_{B'}^T A_{B'}^{-1} A_{N'})^T$, $z_0 = \mathbf{c}_{B'}^T A_{B'}^{-1}\mathbf{b}$ from the original matrix A and the vectors $\mathbf{b}$ and $\mathbf{c}$.

In computer implementations of the Simplex Method, however, this is never done (as it is inefficient). Note that for the next basis exchange, we only need the vector $\overline{\mathbf{c}}$ (to select the next entering variable $\ell \in N'$ with $\overline{c}_\ell > 0$), and for the chosen $\ell \in N'$, the column $\overline{A}_{\cdot\ell}$ and the vector $\overline{\mathbf{b}}$ (to find the next leaving variable $k \in B'$). For that, the matrix $A_{B'}^{-1}$ is computed (which is required to calculate all needed entries). This procedure is known as *Revised Simplex Algorithm*, see [10].

Step 4 (Testing for Optimality) The Simplex Method stops if an optimal solution is found. To detect this situation, we have the following *optimality criterion* of a simplex tableau:

Lemma 4. *Consider a feasible basis B and its simplex tableau $T(B)$*

$$\begin{aligned} \mathbf{x}_B &= \overline{\mathbf{b}} - \overline{A}\,\mathbf{x}_N \\ z &= z_0 + \overline{\mathbf{c}}^T \mathbf{x}_N \end{aligned}$$

If the basic feasible solution $\mathbf{x}^0$ corresponding to B is non-degenerate (i.e., if $\overline{\mathbf{b}} > \mathbf{0}$), then we have: $\mathbf{x}^0$ is the optimal solution if and only if $\overline{\mathbf{c}} \leq \mathbf{0}$.

Indeed, $\mathbf{x}^0 = \binom{\overline{\mathbf{b}}}{\mathbf{0}}$ has the objective function value equal to z_0, while for any other feasible solution $\tilde{\mathbf{x}}$, we have $\tilde{\mathbf{x}}_N \geq \mathbf{0}$ and $\mathbf{c}^T\tilde{\mathbf{x}} = z_0 + \overline{\mathbf{c}}^T\tilde{\mathbf{x}}_N \leq z_0$ (by $\overline{\mathbf{c}} \leq \mathbf{0}$).

It is left to discuss the efficiency of the Simplex Method and pivoting. The number of pivot steps (i.e. basis exchanges) for solving a linear program by the Simplex Method strongly depends on the choices which variables should leave or enter the basis: Figure 14 shows an example where, starting from an initial basic feasible solution, the optimal solution could be reached in three or two steps.

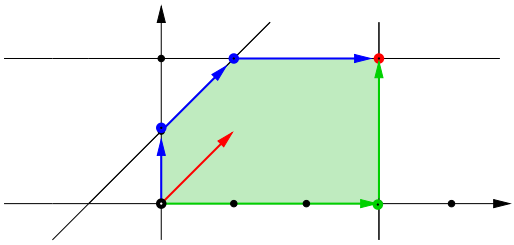

Fig. 14 Different basis exchanges towards the optimal solution.

We do not know in advance which choices will be good if there are several possibilities of *improving variables* (i.e. nonbasic variables x_j with $j \in N$ from the current tableau with $\overline{c}_j > 0$). We denote the index set of the improving variables by N^+.

A *pivot rule* is a rule how to select the entering variable among the improving ones (some rules also specify the choice of the leaving variable, if necessary).

Some well-known pivot rules are:

- **Largest Coefficient Rule:** choose an improving variable x_ℓ such that $\overline{c}_\ell = \max\{\overline{c}_j : j \in N^+\}$ (to maximize the improvement of z per unit increase of x_ℓ)
- **Largest Increase Rule:** choose an improving variable that yields the maximal improvement in z (this rule is computationally more expensive, but locally maximizes the progress)
- **Steepest Edge Rule:** choose an improving variable maximizing the value

$$\frac{\mathbf{c}^T(\mathbf{x}_{new} - \mathbf{x}_{old})}{||\mathbf{x}_{new} - \mathbf{x}_{old}||}$$

  (to move the current basic feasible solution into a direction closest to the one of the objective function $\mathbf{c}$)
- **Bland's Rule:** choose the improving variable x_ℓ with the smallest index $\ell \in N^+$; if there are several possibilities for the leaving variable, also take the one with the smallest index.

The Largest Coefficient Rule is the original rule by Dantzig [8], whereas the Steepest Edge Rule is the champion in practice, and Bland's rule is particularly important, since we have:

Theorem 10 (Bland [4]). *The Simplex Method with Bland's rule is always finite, as cycling is impossible.*

Using other pivot rules, the Simplex Method may cycle (and theoretically, this is the only possibility how it may fail). In fact, for (almost) all pivot rules, there are worst case examples known that require an exponential number of pivot steps (e.g. for Dantzig's rule one in n variables and inequalities requiring $2^n - 1$ pivot steps by Klee & Minty [23]). Note that in practice, most implementations of the Simplex Method try to circumvent cycling via different perturbation techniques.

In theory, the best known worst case bound for the running time of the Simplex Method is, therefore, $e^{c\sqrt{n \ln n}}$ for linear programs with n variables and constraints, using a simple randomized pivot rule (randomly permute the indices of the variables, then apply Bland's rule).

In practice, however, the Simplex Method performs very satisfactory even for large linear programs. Computational experiments indicate that it reaches, for linear programs with m equations, an optimal solution in something between $2m$ and $3m$ pivot steps, with about $O(m^2)$ arithmetic operations per pivot step, such that the *expected running time* is about $O(m^3)$.

3.4 Linear Programming Duality

In this subsection, we address the problem to obtain bounds for the objective function value of a linear program, e.g. an upper bound for the value of an optimal

solution of a maximization problem, without knowing the optimum before. To this end, we shall start with an introductory example.

Example. Consider the following linear program:

$$\begin{array}{lrcl} \max & 2x_1 + 3x_2 & & \\ s.t. & 4x_1 + 8x_2 & \leq & 12 \\ & 2x_1 + & x_2 & \leq 3 \\ & x_1, & x_2 & \geq 0 \end{array}$$

Without computing the optimum z^*, we can infer from the first inequality and non-negativity that $z^* \leq 12$ holds as

$$2x_1 + 3x_2 \leq 4x_1 + 8x_2 \leq 12.$$

We obtain a better bound by scaling the inequality by a factor 2:

$$2x_1 + 3x_2 \leq 2x_1 + 4x_2 \leq 6.$$

Adding the two inequalities and scaling by a factor 3 even yields:

$$2x_1 + 3x_2 \leq 2x_1 + 3x_2 \leq 5.$$

How good can a so-obtained *upper bound* $u \geq \mathbf{c}^T\mathbf{x}$ for all feasible solutions $\mathbf{x}$ of the studied linear program be? To answer this question, we shall derive an inequality of the form $d_1x_1 + d_2x_2 \leq u$, where $d_1 \geq 2$, $d_2 \geq 3$, and u is as small as possible. Then, for all $x_1, x_2 \geq 0$, we indeed have

$$2x_1 + 3x_2 \leq d_1x_1 + d_2x_2 \leq u.$$

For that, we combine the two inequalities of the linear program with some non-negative coefficients y_1 and y_2, obtain

$$(4y_1 + 2y_2)x_1 + (8y_1 + y_2)x_2 \leq 12y_1 + 3y_2.$$

and infer that $d_1 = 4y_1 + 2y_2$, $d_2 = 8y_1 + y_2$, and $u = 12y_1 + 3y_2$ holds. For choosing the best coefficients d_1 and d_2, we must ensure $d_1 \geq 2$, $d_2 \geq 3$ and u being minimal under these constraints. This leads to

$$\begin{array}{lrcl} \min & 12y_1 + 3y_2 & & \\ s.t. & 4y_1 + 2y_2 & \geq & 2 \\ & 8y_1 + & y_2 & \geq 3 \\ & y_1, & y_2 & \geq 0 \end{array}$$

the linear program being *dual* to the original linear program we started with. Every of its feasible solutions yields an upper bound for the objective function value of the original (*primal*) linear program.

We now shall formalize this process. Given a matrix $A \in \mathbf{R}^{m\times n}$ and vectors $\mathbf{b} \in \mathbf{R}^m, \mathbf{c} \in \mathbf{R}^n$. Consider the *primal linear program (P)*

$$\begin{aligned} \max\ & \mathbf{c}^T\mathbf{x} \\ s.t.\ & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

To determine an upper bound $u \geq \mathbf{c}^T\mathbf{x}$ for all $\mathbf{x} \in P(A,\mathbf{b})$, combine the m inequalities of $A\mathbf{x} \leq \mathbf{b}$ with non-negative coefficients $y_1,\ldots,y_m$ s.t. the resulting inequality has the jth coefficient at least c_j, for $1 \leq j \leq m$, the right hand side is as small as possible. This leads to the *dual linear program (D)*

$$\begin{aligned} \min\ & \mathbf{b}^T\mathbf{y} \\ s.t.\ & A^T\mathbf{y} \geq \mathbf{c} \\ & \mathbf{y} \geq \mathbf{0}. \end{aligned}$$

The primal and the dual linear program are related as follows:

Theorem 11 (Weak Duality Theorem). *Consider the dual linear programs*

$$\begin{aligned} &\max \mathbf{c}^T\mathbf{x}\ s.t.\ A\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\ \ (P) \\ &\min \mathbf{b}^T\mathbf{y}\ s.t.\ A^T\mathbf{y} \geq \mathbf{c},\ \mathbf{y} \geq \mathbf{0}\ \ (D) \end{aligned}$$

- *For each feasible solution* $\mathbf{y}$ *of (D), the value* $\mathbf{b}^T\mathbf{y}$ *provides an* upper bound *for the maximum objective function value of (P), i.e., we have* $\mathbf{c}^T\mathbf{x} \leq \mathbf{b}^T\mathbf{y}$ *for each feasible solution* $\mathbf{x}$ *of (P).*
- *If (P) is unbounded, then (D) is infeasible.*
- *If (D) is unbounded (from below), then (P) is infeasible.*

Theorem 12 (Strong Duality Theorem). *For the dual linear programs*

$$\begin{aligned} &\max \mathbf{c}^T\mathbf{x}\ s.t.\ A\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\ \ (P) \\ &\min \mathbf{b}^T\mathbf{y}\ s.t.\ A^T\mathbf{y} \geq \mathbf{c},\ \mathbf{y} \geq \mathbf{0}\ \ (D) \end{aligned}$$

exactly one of the following possibilities occurs:

- *Neither (P) nor (D) has a feasible solution.*
- *(P) is unbounded and (D) has no feasible solution.*
- *(P) has no feasible solution and (D) is unbounded.*
- *Both (P) and (D) have a feasible solution. Then both linear programs have an optimal solution, say* $\mathbf{x}^*$ *of (P) and* $\mathbf{y}^*$ *of (D), and* $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}^*$ *holds.*

Proofs of the two duality theorems can be found in [26], for instance.

The two duality theorems are valid for all kinds of linear programs, we only have to construct the dual program properly: For a maximization problem with constraint matrix $A \in \mathbf{R}^{m\times n}$, right hand side vector $\mathbf{b} \in \mathbf{R}^m$, objective vector $\mathbf{c} \in \mathbf{R}^n$, the dual program has

- variables $y_1, \ldots, y_m$ where y_i corresponds to the ith constraint and satisfies

$$y_i \left\{ \begin{array}{l} \geq 0 \\ \leq 0 \\ \in \mathbf{R} \end{array} \right\} \quad \text{if} \quad A_{i\cdot}\mathbf{x} \left\{ \begin{array}{l} \leq \\ \geq \\ = \end{array} \right\} b_i$$

- n constraints, where the jth constraint corresponds to x_j and reads

$$A_{\cdot j}\mathbf{y} \left\{ \begin{array}{l} \geq \\ \leq \\ = \end{array} \right\} c_j \quad \text{if} \quad x_j \left\{ \begin{array}{l} \geq 0 \\ \leq 0 \\ \in \mathbf{R} \end{array} \right\}$$

- the objective function $\mathbf{b}^T\mathbf{y}$ which is to be minimized.

We can summarize these conditions as the following "Dualization Recipe":

	Primal linear program	Dual linear program
Variables	$x_1, x_2, \ldots, x_n$	$y_1, y_2, \ldots, y_m$
Matrix	$A \in \mathbf{R}^{m\times n}$	$A^T \in \mathbf{R}^{n\times m}$
Right-hand side	$\mathbf{b} \in \mathbf{R}^m$	$\mathbf{c} \in \mathbf{R}^n$
Objective function	$\max \mathbf{c}^T\mathbf{x}$	$\min \mathbf{b}^T\mathbf{y}$
Constraints	ith constraint has $\leq$	$y_i \geq 0$
	$\geq$	$y_i \leq 0$
	$=$	$y_i \in \mathbf{R}$
	$x_j \geq 0$	jth constraint has $\geq$
	$x_j \leq 0$	$\leq$
	$x_j \in \mathbf{R}$	$=$

The implications for the solvability of two dual linear programs are due to the Farkas Lemma [14, 15] (see also [26] for a proof):

Theorem 13 (Farkas Lemma). *For $A \in \mathbf{R}^{m\times n}$ and $\mathbf{b} \in \mathbf{R}^m$, exactly one of the following two possibilities occurs:*

1. *There is a vector $\mathbf{x} \in \mathbf{R}^n$ satisfying $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$.*
2. *There is a vector $\mathbf{y} \in \mathbf{R}^m$ s.t. $\mathbf{y}^T A \geq \mathbf{0}^T$ and $\mathbf{y}^T\mathbf{b} < \mathbf{0}$.*

Remark. The Farkas Lemma has several variants for the different types of linear programs, which can be summarized as follows:

	The system $A\mathbf{x} \leq \mathbf{b}$	The system $A\mathbf{x} = \mathbf{b}$
has a solution $\mathbf{x} \geq \mathbf{0}$ if and only if	$\mathbf{y} \geq \mathbf{0}$ and $\mathbf{y}^T A \geq \mathbf{0}$ imply $\mathbf{y}^T \mathbf{b} \geq \mathbf{0}$	$\mathbf{y}^T A \geq \mathbf{0}^T$ implies that $\mathbf{y}^T \mathbf{b} \geq \mathbf{0}$
has a solution $\mathbf{x} \in \mathbf{R}$ if and only if	$\mathbf{y} \geq \mathbf{0}$ and $\mathbf{y}^T A = \mathbf{0}$ imply $\mathbf{y}^T \mathbf{b} \geq \mathbf{0}$	$\mathbf{y}^T A = \mathbf{0}^T$ implies that $\mathbf{y}^T \mathbf{b} = \mathbf{0}$

That is: if the primal and the dual linear program are neither infeasible nor unbounded, then the maximum of the primal program (P) equals the minimum of the dual program (D).

This leads to duality-based Simplex Methods to solve a linear program: the Dual Simplex Method and so-called primal-dual methods:

- To solve a linear program, we can apply the Simplex Method either to the primal linear program or to its dual linear program. The *Dual Simplex Method* solves the dual linear program by starting with a dual feasible basis and trying to attain primal feasibility while maintaining dual feasibility throughout. This can be substantially faster if
  - the dual linear program has less constraints than the primal linear program, or
  - an initial (dual) basic feasible solution is easy to obtain, or
  - the dual linear program is less degenerate.
- *Primal-Dual Methods* solve a linear program by iteratively improving a feasible solution of the dual linear program:
  - Consider a primal linear program given by $\max \mathbf{c}^T \mathbf{x}$ *s.t.* $A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$.
  - For a feasible dual solution $\mathbf{y}$, define $J = \{j \in \{1,\ldots,n\} : A_{\cdot j}\mathbf{y} = c_j\}$.
  - A dual solution $\mathbf{y}$ is optimal if and only if there is a feasible primal solution $\mathbf{x}$ with
$$x_j = 0 \ \forall j \in \{1,\ldots,n\} \setminus J.$$

In addition to the aforementioned relations between primal and the dual linear programs, we have even more: If a primal linear program is a formulation for a combinatorial optimization problem, then its dual linear program has also an interpretation as a combinatorial optimization problem, related to the combinatorial object being dual to the original studied one.

We shall illustrate this relation with the help of our running example, the Network Flow Problem.

Example 5 (Dualization of Maximum Network Flow). Given a network $N = (D; s, t; \mathbf{c})$ with digraph $D = (V, A)$ and capacities $\mathbf{c} \in \mathbf{Z}^A$. Recall from Problem 3 that the linear programming formulation of the Maximum Network Flow Problem is:

$$\begin{array}{rll} \max & \sum_{a\in\delta^+(s)} x_a & \\ s.t. & \sum_{a\in\delta^-(v)} x_a = \sum_{a\in\delta^+(v)} x_a & \forall v \in V\setminus\{s,t\} \\ & x_a \leq c_a & \forall a \in A \\ & x_a \geq 0 & \forall a \in A \end{array}$$

With $V' = V \setminus \{s,t\}$ denoting the set of internal nodes of the digraph, let

- $F \in \mathbf{Z}^{A\times V'}$ be the matrix of the flow conservation constraints,
- $\mathbf{d} \in \mathbf{Z}^A$ with $d_a = 1$ if $a \in \delta^+(s)$, $d_a = 0$ otherwise be the objective vector.

Then the *primal linear program (P)* encoding the Maximum Flow Problem reads in matrix notation:

$$\begin{array}{rll} \max & \mathbf{d}^T\mathbf{x} & \\ s.t. & F\ \mathbf{x} = \mathbf{0} & \\ & I\ \mathbf{x} \leq \mathbf{c} & \\ & \mathbf{x} \geq \mathbf{0} & \end{array}$$

For the dualization, we use one variable

- z_v for the flow conservation constraint of $v \in V'$,
- y_a for the capacity constraint of $a \in A$.

This leads to the following *dual linear program (D)*

$$\begin{array}{rl} \min & \mathbf{c}^T\mathbf{y} \\ s.t. & F^T\mathbf{z} + I^T\mathbf{y} \geq \mathbf{d} \\ & \mathbf{y} \geq \mathbf{0} \end{array}$$

A closer look to the dual program shows that the dual program has

- one variable $z_v \in \mathbf{R}$ corresponding to the flow conservation for each $v \in V'$:

$$x(\delta^-(v)) - x(\delta^+(v)) = 0$$

- one variable $y_a \geq 0$ corresponding to the capacity constraint for each $a \in A$,
- for each primal variable $x_a, a \in A$, one constraint $F_{\cdot a}\ \mathbf{z} + I_{\cdot a}\ \mathbf{y} \geq d_a$ which reads, for $a = (u,v) \in A$

$$\begin{array}{rl} z_v - z_u + y_a \geq 0 & \text{if } u \neq s, v \neq t \\ z_v \quad + y_a \geq 1 & \text{if } u = s, v \neq t \\ -z_u + y_a \geq 0 & \text{if } u \neq s, v = t \end{array}$$

- the objective function $\mathbf{c}^T\mathbf{y}$ which is to be minimized.

What is the combinatorial interpretation of the dual program? For a network $N = (D;s,t;\mathbf{c})$ with $D = (V,A)$, consider the dual program

$$\begin{array}{rl} \min & \mathbf{c}^T\mathbf{y} \\ s.t. & z_v - z_u + y_a \geq 0 \ \forall\, a = (u,v) \in A \\ & z_s = 1 \\ & z_t = 0 \\ & y_a \geq 0 \ \forall\, a = (u,v) \in A \end{array}$$

Recall that for a partition of $V = V_s \cup V_t$ with $s \in V_s$ and $t \in V_t$, the subset of arcs $\delta^+(V_s) = \{(u,v) \in A : u \in V_s, v \in V_t\}$ is an (s,t)-cut. Hence, each (s,t)-cut $\delta^+(V_s)$ of a network $N = (D;s,t;\mathbf{c})$ with $D = (V,A)$ corresponds to a feasible solution $(\mathbf{z},\mathbf{y})^T \in \mathbf{R}^{V'} \times \mathbf{R}^A_+$ of the dual program with

$$\begin{array}{ll} z_v = 1 \text{ if } v \in V_s, & z_u = 0 \text{ if } u \in V_t \\ y_a = 1 \text{ if } a \in \delta^+(V_s), & y_a = 0 \text{ if } a \notin \delta^+(V_s). \end{array}$$

Recall further that the *flow* across the (s,t)-cut (V_s,V_t) is

$$f(V_s,V_t) = \sum_{u \in V_s, v \in V_t} f(uv) - \sum_{u \in V_s, v \in V_t} f(vu)$$

and its *capacity* is

$$c(V_s,V_t) = \sum_{u \in V_s, v \in V_t} c(uv).$$

Obviously, $val(f) \leq c(V_s,V_t)$ holds for any (s,t)-cut. We have even more: Since every (s,t)-flow f satisfies the capacity constraints, we have that $f(V_s,V_t) \leq c(V_s,V_t)$ and thus

$$val(f) \leq c(V_s,V_t)$$

holds for any (s,t)-cut. This upper bound for the maximum flow in a network also follows from the Weak Duality Theorem (Theorem 11), and the Max-Flow Min-Cut Theorem (Theorem 1), is a famous special case of the Strong Duality Theorem (Theorem 12), which implies:

$$\begin{array}{rl} \max & \mathbf{d}^T\mathbf{x} \\ s.t. & F\ \mathbf{x} = \mathbf{0} \\ & I\ \mathbf{x} \leq \mathbf{c} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \quad = \quad \begin{array}{rl} \min & \mathbf{c}^T\mathbf{y} \\ s.t. & F^T\mathbf{z} + I^T\mathbf{y} \geq \mathbf{d} \\ & \mathbf{y} \geq \mathbf{0} \end{array}$$

In particular, the linear programming formulation for Maximum Network Flow from Problem 3 is the "right" formulation as it does not only properly encode the primal problem, but also its dual linear program has an interpretation as a Minimum Cut Problem, the combinatorial problem being dual to the original studied Network Flow Problem.

4 Integer Programming and the Network Flow Problem

In the previous section, we considered a linear program, i.e. the problem to

$$\begin{array}{ll} \text{maximize/minimize the value of} & \mathbf{c}^T\mathbf{x} \\ \text{among all vectors } \mathbf{x} \in \mathbf{R}^n \text{ satisfying} & A\,\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

where $A \in \mathbf{R}^{m\times n}$ is a given matrix and $\mathbf{b} \in \mathbf{R}^m, \mathbf{c} \in \mathbf{R}^n$ are given vectors.

If in some practical settings, the studied objects are entities as workers, goods or planes which cannot be divided, we do not consider variables $\mathbf{x} \in \mathbf{R}^n$ but $\mathbf{x} \in \mathbf{Z}^n$. This leads to an *Integer Linear Optimization Problem*. In this section we discuss

- how linear programs and integer linear programs are related,
- why integer linear programs are hard to solve in general, and
- what is special for solving Integer Network Flow Problems.

4.1 Integer linear programs and their linear relaxations

We first address the question what an integer linear program is.

Definition 7. An *integer linear program (ILP)* is as follows:

$$\begin{array}{ll} \text{maximize/minimize the value of} & \mathbf{c}^T\mathbf{x} \\ \text{among all vectors } \mathbf{x} \in \mathbf{Z}^n \text{ satisfying} & A\,\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

where $A \in \mathbf{R}^{m\times n}$ is a given constraint matrix, $\mathbf{b} \in \mathbf{R}^m$ a given right hand side vector, and $\mathbf{c} \in \mathbf{R}^n$ a given objective function vector (typically, also the entries of $A, \mathbf{b}, \mathbf{c}$ are integral in this case).

We illustrate this formal definition with the help of a small example:

Example 6. This example shows an integer linear program given explicitly as well as in matrix formulation:

$$\begin{array}{lrll} \max & x_1 + x_2 & & \text{is the linear objective function } \mathbf{c}^T\mathbf{x} \\ s.t. & -x_1 + x_2 & \leq 1 & \\ & x_1 & \leq 3 & \text{form the linear constraints } A\mathbf{x} \leq \mathbf{b} \\ & x_2 & \leq 2 & \\ & x_1, x_2 & \geq 0 & \text{are the non-negativity constraints } \mathbf{x} \geq \mathbf{0} \\ & x_1, x_2 & \in \mathbf{Z} & \text{are the integrality constraints } \mathbf{x} \in \mathbf{Z}^2 \end{array}$$

Figure 15 gives the graphical interpretation of the constraints and the resulting polyhedron $P(A, \mathbf{b})$.

In an integer linear program

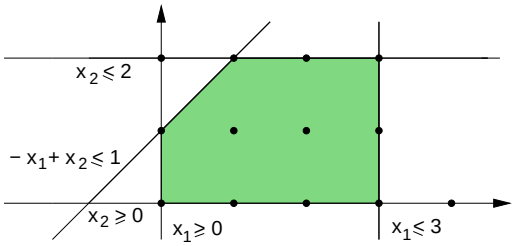

Fig. 15 The graphical interpretation of the constraints and the polyhedron $P(A,\mathbf{b})$ (the shaded region) containing the feasible solutions $\mathbf{x} \in \mathbf{Z}^2$ of the integer linear program given in Example 6.

$$\max \mathbf{c}^T\mathbf{x}, A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \mathbf{Z}^n$$

we still have a linear objective function, and the side constraints $\mathbf{A}^T\mathbf{x} \leq b$ are linear and describe a polyhedron $P(A,b)$, but the feasible points are just the *lattice points* $\mathbf{x} \in P(A,b) \cap \mathbf{Z}^n$.

In particular and in contrast to the case of Linear Programming, an optimal solution of an integer linear program is not necessarily attained on the boundary of $P(A,b)$, but may be situated in its interior. Figure 16(a) illustrats the case where none of the extreme points of $P(A,b)$ is integral.

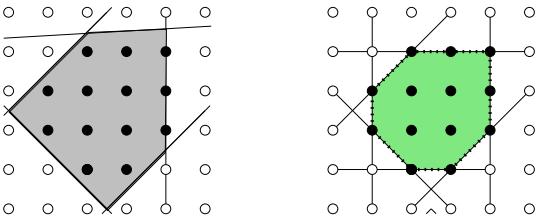

Fig. 16 The feasible points of an integer linear program (a) and a linear constraint system for the convex hull of all its integral solutions (b).

This makes the problem hard in general (see, for instance, [22] for a proof and [3, 20, 25, 26, 27] for further information):

Theorem 14. *It is NP-hard to decide whether an integer linear program has a solution above/below a certain threshold.*

In contrary, the corresponding linear program obtained by dropping the integrality requirement, can be solved in polynomial time. How are linear and integer linear programs related to each other?

Definition 8. For an integer linear program

$$\max \mathbf{c}^T\mathbf{x}, A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \mathbf{Z}^n$$

the linear program

$$\max \mathbf{c}^T\mathbf{x}, A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \mathbf{R}^n$$

obtained by dropping the integrality requirements is called a *linear relaxation* as its feasible region $P(A,\mathbf{b})$ contains all integral feasible points $\mathbf{x} \in P(A,b) \cap \mathbf{Z}^n$ of the corresponding integer linear program.

The linear relaxation can be solved in polynomial time, but its optimal solution might be fractional and, thus, no solution of the corresponding integer linear program.

However, the convex hull of all integral solutions of an integer linear program is a polyhedron and, thus, can be described by means of linear inequalities, see Figure 16(b). Thus, in principle there exists a constraint system for each integer linear program, called ideal formulation, such that the feasible region has integral extreme points only:

Definition 9. For an integer linear program

$$\max \mathbf{c}^T\mathbf{x}, A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \mathbf{Z}^n$$

a linear program

$$\max \mathbf{c}^T\mathbf{x}, \overline{A}\mathbf{x} \leq \overline{\mathbf{b}}, \mathbf{x} \in \mathbf{R}^n$$

is an *ideal formulation* if

$$P(\overline{A},\overline{\mathbf{b}}) = \{\mathbf{x} \in \mathbf{R}^n_+ : \overline{A}\mathbf{x} \leq \overline{\mathbf{b}}\} \;=\; \text{conv}\{\mathbf{x} \in \mathbf{Z}^n_+ : A\mathbf{x} \leq \mathbf{b}\}.$$

As the optimum of a linear program is always attained at an extreme point of $P(A,\mathbf{b})$, linear programming techniques can be applied to solve integer linear programs given as ideal formulations! This leads to *polynomial time solvability*, provided that the required inequalities can be separated in polynomial time (i.e., that it can be checked efficiently whether a given point satisfies all inequalities or violates some of them; for instance, this is the case if the ideal formulation contains only a polynomial number of constraints).

In general, finding an ideal formulation for an integer linear program is as hard as solving the problem itself. In some special cases, however, certain properties related to the underlying combinatorial problem can lead to the desired situation, e.g. if the constraint matrix A of the integer linear program has a special structure. We will next define such a type of matrices.

Definition 10.

- A matrix $A \in \mathbf{Z}^{m\times n}$ of full row rank is *unimodular* if the determinant of each basis of A is in $\{-1,1\}$.
- A matrix $A \in \mathbf{Z}^{m\times n}$ is *totally unimodular* if the determinant of each square submatrix of A is in $\{-1,0,1\}$.

Remark. Unimodular and totally unimodular matrices must have entries in $\{-1,0,1\}$ only. A matrix $A \in \mathbf{Z}^{m\times n}$ is totally unimodular if and only if

- (A,I) is totally unimodular;
- A^T is totally unimodular.

The following matrices A, A^T, (A^T,I) are examples of totally unimodular matrices:

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Recall that a polyhedron is integral if all its extreme points are integral. The relation of unimodularity and the integrality of polyhedra coming from integer programming formulations is as follows (see [21] for the proof and [3, 20, 25, 26, 27] for further information):

Theorem 15. *Consider a matrix $A \in \mathbf{Z}^{m\times n}$.*

- *For A with full row rank, $P^{=}(A,\mathbf{b}) = \left\{\mathbf{x} \in \mathbf{R}^n_+ : A\mathbf{x} = \mathbf{b}\right\}$ is integral for all right hand side vectors $\mathbf{b} \in \mathbf{Z}^m$ with $P^{=}(A,\mathbf{b}) \neq \emptyset$ if and only if A has full row rank and is unimodular.*
- *$P(A,\mathbf{b}) = \left\{\mathbf{x} \in \mathbf{R}^n_+ : A\mathbf{x} \leq \mathbf{b}\right\}$ is integral for all right hand side vectors $\mathbf{b} \in \mathbf{Z}^m$ with $P(A,\mathbf{b}) \neq \emptyset$ if and only if A is totally unimodular.*

Remark. The proof of the latter theorem is based on *Cramer's rule*: For a non-singular matrix $A \in \mathbf{R}^{n\times n}$ and $\mathbf{b} \in \mathbf{Z}^n$, we have

$$A\mathbf{x} = \mathbf{b} \iff \mathbf{x} = A^{-1}\mathbf{b} \iff x_i = \frac{\det(A^i)}{\det(A)}$$

where A^i is obtained from A by replacing the i-th column by $\mathbf{b}$. From $\det(A) \in \{-1,1\}$ for totally unimodular matrices, it follows $x_i \in \mathbf{Z}$.

Thus, all integer linear programs with (totally) unimodular constraint matrices have an integral polyhedron as the convex hull of its feasible solutions and can be solved with the help of linear programming techniques.

However, as the hardness of solving integer linear programs implies, we do not always have totally unimodular constraint matrices. A more general setting involves Linear Programming Duality:

Definition 11. A system $A\mathbf{x} \leq \mathbf{b}$ of linear inequalities is *totally dual integral (TDI)* if the linear program

$$\min \mathbf{b}^T\mathbf{y} \text{ s.t. } A^T\mathbf{y} = \mathbf{c},\ \mathbf{y} \geq \mathbf{0}$$

has an integral optimal solution for every integral vector $\mathbf{c}$ such that $\max \mathbf{c}^T\mathbf{x}, A\mathbf{x} \leq \mathbf{b}$ is bounded.

Note that A is totally unimodular if and only if the system $A\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$ is totally dual integral for *all* integral vectors $\mathbf{b}$.

Also the concept of totally dual integrality, introduced by Edmonds & Giles [12], is related to the integrality of polyhedra:

Theorem 16. *If the system $A\mathbf{x} \leq \mathbf{b}$ is totally dual integral, $A \in \mathbf{R}^{m\times n}$ and $\mathbf{b} \in \mathbf{R}^m$, then we have:*

- *The primal problem*
$$\max \mathbf{c}^T\mathbf{x} \text{ s.t. } A\mathbf{x} \leq \mathbf{b}$$
has an integral optimal solution for all $\mathbf{c} \in \mathbf{Z}^n$;
- *The polytope*
$$P(A,\mathbf{b}) = \{\mathbf{x} \in \mathbf{R}^n : A\mathbf{x} \leq \mathbf{b}\}$$
is integral.

To summarize: Ideal formulations follow the idea to go back to linear programs in order to solve integer linear programs.

In fact, we can apply linear programming techniques and solve an integer linear program in polynomial time in one of the following situations: the canonical integer linear programming formulation $A\,\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$ is

- *ideal* as $P(A,\mathbf{b})$ is *integral*;
- not ideal, but the constraint system $A\,\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$ with $P(A,\mathbf{b}) = \text{conv}\{\mathbf{x} \in \mathbf{Z}^n_+ : A\,\mathbf{x} \leq \mathbf{b}\}$ is easy to find by adding (polynomially many) further constraints;
- not ideal in general, but is ideal for some *special cases* where additional combinatorial properties are satisfied.

Note that ideal formulations for integer linear programs typically involve a much larger number of constraints than compact formulations using integrality requirements (which makes the separation problem harder).

For most integer optimization problems, no ideal formulation is known at all. In this general situation, one might start from a canonical integer linear program $A\,\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$ and try to find hyperplanes approximating $\text{conv}\{\mathbf{x} \in \mathbf{Z}^n_+ : A\,\mathbf{x} \leq \mathbf{b}\}$ at the "right place" (i.e. near the optimal solution as illustrated in Figure 17).

Such approaches to enhance the original formulation are called *cutting plane methods* and work as follows:

Generic Cutting Plane Method
Input: an integer linear program (ILP) $\max \mathbf{c}^T\mathbf{x}, A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \mathbf{Z}^n$
Output: an optimal integer solution $\mathbf{x}^*$

1. Solve the linear relaxation (LP) $\max \mathbf{c}^T\mathbf{x}, A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \mathbf{R}^n$.
   If $P(A,b)$ is *empty*, then the ILP is also infeasible, and STOP.
   Else, let $\mathbf{x}^*$ be an optimal (extreme point) solution of the LP.
2. If $\mathbf{x}^*$ is *integral*, then STOP because $\mathbf{x}^*$ is also optimal for the ILP.
3. If $\mathbf{x}^*$ is *not integral*, then find an inequality that is satisfied by all feasible solutions of the ILP, but is violated by $\mathbf{x}^*$.
4. Append this inequality (the *cutting plane*) to the LP, and proceed with Step 1.

A classical way to generate new valid inequalities from the known constraints in $A\mathbf{x} \leq \mathbf{b}$ are so-called Chvátal-Gomory cuts, introduced by Chvátal [5] and implicitly by Gomory [19].

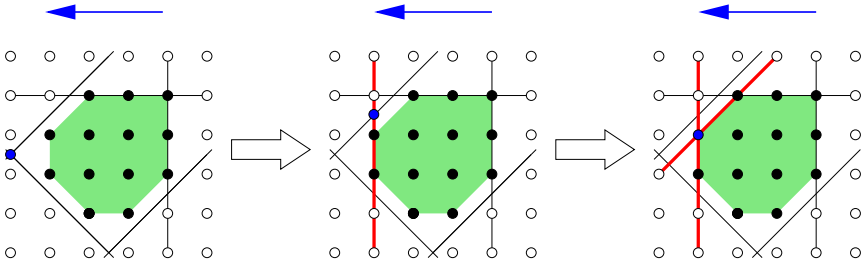

Fig. 17 The feasible points of an integer linear program and additional linear constraints (cutting planes) approximating the convex hull of all its integral solutions.

For any polyhedron $P(A,\mathbf{b})$, let $P_I(A,\mathbf{b})$ denote the convex hull of all integer points in $P(A,\mathbf{b})$. If $\sum a_i x_i \leq b$ is a valid inequality for $P(A,\mathbf{b})$ and has integer coefficients only, then $\sum a_i x_i \leq \lfloor b \rfloor$ is a Chvátal-Gomory cut for $P(A,\mathbf{b})$ and valid for $P_I(A,\mathbf{b})$. In fact, every valid inequality for the convex hull of all integral solutions can be generated by applying the Chvátal-Gomory procedure (i.e., adding all Chvátal-Gomory cuts) to $P(A,b)$ a finite number of times. This guarantees that cutting plane methods indeed terminate.

It is a currently active field of research to find more efficient cutting planes than the classical ones, as e.g. split cuts, intersection cuts and others [1, 2, 6, 11].

For more information on Integer Programming, see e.g. [3, 20, 25, 26, 27].

4.2 Computing integer network flows

We finally discuss the Integer Network Flow Problem. Given a network $N = (D;s,t;\mathbf{c})$ with $D = (V,A)$, the problem of finding an integral (s,t)-flow $f : A \to \mathbf{Z}$ maximizing the value $val(f)$. In order to formulate this problem as integer linear program, we again need

- the variables x_a to express the flow $f(a)$ on each arc $a \in A$,
- the linear objective function max $\sum_{a\in\delta^+(s)} x_a$ of maximizing the flow leaving the source s,
- the linear flow conservation constraints $\sum_{a\in\delta^-(v)} x_a = \sum_{a\in\delta^+(v)} x_a \;\; \forall v \in V \setminus \{s,t\}$,
- the linear capacity constraints $x_a \leq c_a \;\; \forall a \in A$,

and in addition, *integrality* is required for all variables:

$$x_a \in \mathbf{Z} \; \forall a \in A.$$

Thus, the problem of finding an integral (s,t)-flow $f : A \to \mathbf{Z}$ of maximal value $val(f)$ leads to:

Problem 4 (Integer Maximum Network Flow Problem). Given a network $N = (D;s,t;c)$ with digraph $D = (V,A)$. Finding an (s,t)-flow $f : A \to \mathbf{Z}$ maximizing the

value $val(f)$ by solving the following integer linear program:

$$\begin{array}{rll} \max & \sum_{a\in\delta^+(s)} x_a & \\ s.t. & \sum_{a\in\delta^-(v)} x_a = \sum_{a\in\delta^+(v)} x_a & \forall v \in V\setminus\{s,t\} \\ & x_a \leq c_a & \forall a \in A \\ & x_a \geq 0 & \forall a \in A \\ & x_a \in \mathbf{Z} & \forall a \in A \end{array}$$

With $V' = V\setminus\{s,t\}$ denoting the set of internal nodes of the digraph, let again

- $F \in \mathbf{Z}^{A\times V'}$ be the matrix of the flow conservation constraints,
- $\mathbf{d} \in \mathbf{Z}^A$ with $d_a = 1$ if $a \in \delta^+(s)$, $d_a = 0$ otherwise be the objective vector.

Then the program encoding the Integer Maximum Network Flow Problem reads in matrix notation:

$$\begin{array}{rll} \max & \mathbf{d}^T\mathbf{x} & \\ s.t. & F\ \mathbf{x} = \mathbf{0} & \\ & I\ \mathbf{x} \leq \mathbf{c} & \\ & \mathbf{x} \in \mathbf{Z}^A_+ & \end{array}$$

Indeed, every vector $\mathbf{x} \in \mathbf{Z}^A$ satisfying all the above constraints corresponds to an integral (s,t)-flow f, an optimal solution to a maximum flow. How hard or easy is it to compute a maximum flow as optimal solution of the above integer linear program?

As integer linear programs are hard to solve in general, this leads to the question whether we can find an ideal formulation by taking advantage of special combinatorial properties of the underlying Network Flow Problem.

Recalling that all integer linear programs with (totally) unimodular constraint matrices have an integral polyhedron as the convex hull of its feasible solutions, we wonder whether the constraint matrices for the Network Flow Problem satisfy this property.

Indeed, one example for unimodularity are node/arc incidence matrices of digraphs, the underlying discrete structure for the networks of our flow problem:

Theorem 17. *The node/arc incidence matrix of any digraph $D = (V,A)$ is totally unimodular.*

The proof of the latter theorem is based on the following characterization of totally unimodular matrices:

Theorem 18. *A matrix $M \in \mathbf{Z}^{m\times n}$ is totally unimodular if and only if each subset $I \subseteq \{1,\ldots,n\}$ of columns has a bipartition $I = I_A \cup I_B$ s.t. for all rows $j \in \{1,\ldots,m\}$, we have $\sum_{i\in I_A} m_{ji} - \sum_{i\in I_B} m_{ji} \in \{-1,0,1\}$.*

Thus, we shall study how our constraint matrix is related to this property.

In fact, for a network $N = (D;s,t;\mathbf{c})$ with digraph $D = (V,A)$, the flow conservation matrix $F \in \mathbf{Z}^{V'\times A}$ has one row for each of the constraints

$$\sum_{a\in\delta^-(v)} x_a - \sum_{a\in\delta^+(v)} x_a = 0 \;\forall v \in V' = V \setminus \{s,t\}.$$

The column of F for arc $a = (u,v) \in A$ reads as follows:

$$F = \begin{array}{c} \\ \\ u \\ \\ v \\ \\ \end{array} \begin{pmatrix} & a & \\ & \vdots & \\ \cdots & 1 & \cdots \\ & \vdots & \\ \cdots & -1 & \cdots \\ & \vdots & \end{pmatrix}$$

Hence, F is the node/arc incidence matrix of a digraph and, thus indeed *totally unimodular*. As adding the identity matrix I to a totally unimodular matrix yields again a totally unimodular matrix, this implies:

Corollary 1. *The Maximum Network Flow Problem*

$$\begin{array}{rl} \max & \mathbf{d}^T\mathbf{x} \\ s.t. & F\ \mathbf{x} = \mathbf{0} \\ & I\ \mathbf{x} \leq \mathbf{c},\ \mathbf{x} \geq \mathbf{0} \end{array}$$

has for all integral capacities $\mathbf{c} \in \mathbf{Z}_+^A$ *an integral optimum.*

As a matrix is totally unimodular if and only if its transposed matrix is totally unimodular, it follows for the dual linear program:

Corollary 2. *The Minimum Cut Problem*

$$\begin{array}{rl} \min & \mathbf{c}^T\mathbf{y} \\ s.t. & F^T\mathbf{z} + I^T\mathbf{y} \geq \mathbf{d} \\ & \mathbf{y} \geq \mathbf{0} \end{array}$$

has for all integral vectors $\mathbf{d} \in \mathbf{Z}_+^A$ *an integral optimum.*

To conclude: the latter results form an example par excellence in the field of so-called Polyhedral Combinatorics, a powerful, coherent and unifying tool for Combinatorial Optimization, involving algorithms, the geometry of solution sets and min-max relations with dual problems. The studied Network Flow Problem demonstrates that these aspects are closely related in general:

"Often a polynomial-time algorithm yields, as a by-product, a description (in terms of inequalities) of an associated polyhedron. Conversely, an appropriate description of the polyhedron often implies the polynomial-time solvability of the associated optimization problem, by applying linear programming techniques. With the duality theorem of linear programming, polyhedral characterizations yield min-max relations, and vice versa." Alexander Schrijver

References

1. Balas, E., Saxena, A.: Optimizing Over the Split Closure. Mathematical Programming **113**, 219–240 (2008)
2. Basu, A., Cornuéjols, G., Margot, M.: Intersection Cuts with Infinite Split Rank. Mathematics of Operations Research **37**, 21-40 (2012)
3. Bertsimas, D., Weismantel, R. : Optimization over Integers. Dynamic Ideas, Belmont, MA, (2005)
4. Bland, R.G.: New finite pivoting rules for the simplex method. Mathematics of Operations Research **2**, 103–107 (1977)
5. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Math. **4**, 305–337 (1973)
6. Conforti, M., Cornuéjols, G., Zambelli, G.: Corner Polyhedron and Intersection Cuts. Surveys in Operations Research and Management Science **16**, 105-120 (2011)
7. Dantzig, G.B.: Maximization of a linear function of variables subject to linear inequalities. In T.C. Koopmans (ed.) Activity Analysis of Production and Allocation, John Wiley & Sons, New York, (1951) pp. 339–347.
8. Dantzig, G.B.: Notes on linear programming. RAND Corporation,(1953)
9. Dantzig, G.B.: The Diet Problem. Interfaces, The Practice of Mathematical Programming **20**, 43–47 (1990)
10. Dantzig, G.B., Thapa, M.N.: Linear Programming 2/ Theory and Extensions, Springer-Verlag (2003)
11. Del Pia A., Wagner C., Weismantel R.: A probabilistic comparison of the strength of split, triangle, and quadrilateral cuts. Operations Research Letters **39**, 234–240 (2011)
12. Edmonds, J., Giles, R.: A min-max relation for submodular functions on graphs. In: Studies in Integer Programming (Proceedings of the Workshop on Integer Programming, Bonn, 1975; P.L. Hammer, E.L. Johnson, B.H. Korte, G.L. Nemhauser, eds.), 185–204 (1977)
13. Edmonds, J., Karp R.M.: Theoretical improvements in algorithmic efficiency for network flow problems, Journal Assoc. Comput. Mach. **19**, 248–264 (1972)
14. Farkas, G.: A Fourier-féle mechanikai elv alkamazásai. Mathematikai ś Termśzettudományi Értesítö **12**, 457–472 (1894)
15. Farkas, G.: Über die Theorie der Einfachen Ungleichungen. Journal für die Reine und Angewandte Mathematik **124**, 1–27 (1902)
16. Ford, L.R., Fulkerson, D.R.: Maximum flow through a network, Canad. Journal of Mathematics **8**, 399–404 (1956)
17. Ford, L.R., Fulkerson, D.R.: Network Flow Theory, Princeton Press, Princeton (1962)
18. Gale, Kuhn, H., and Tucker:, "Linear Programming and the Theory of Games - Chapter XII", in Koopmans, Activity Analysis of Production and Allocation, Wiley, http://cowles.econ.yale.edu/P/cm/m13/m13-19.pdf . (1951) See Lemma 1 on page 318.
19. Gomory, R.: Outline of an algorithm for integer solutions to linear programs. Bul. of the American Math. Soc. **64**, 275–278 (1958)
20. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Berlin, 1988
21. Hoffman, A.J., Kruskal, J.B.: Integral Boundary Points of Convex Polyhedra. In Kuhn, H.W.; Tucker, A.W., Linear Inequalities and Related Systems, Annals of Mathematics Studies **38**, Princeton University Press, Princeton, (1956) pp. 223–246
22. Kannan, R., Monma, C.L.: On the computational complexity of integer programming problems. Lecture Notes in Economws and Mathematical Systems **157**, 161–172 (1978)
23. Klee, V., Minty, G.J.: How good is the simplex algorithm? In Shisha, Oved. Inequalities III, Academic Press, New York-London (1972), pp. 159–175
24. Minkowski, H.: Allgemeine Lehrsätze über konvexe Polyeder. Ges. Wiss. Göttingen, 198–219 (1897)
25. Nemhauser G.L., Wolsey L.A.: Integer Programming and Combinatorial Optimization. Wiley-Interscience, New-York (1998)

26. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, Chichester (1986)
27. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin-Heidelberg (2003)
28. Weyl, H.: Elementare Theorie der konvexen Polyeder. Comment. Math. Helvetici, 7 (1935)