LARGE SCALE OPTIMIZATION

Alexander Martin, Department of Mathematics, Darmstadt University of Technology, Germany

Keywords Large scale optimization; mixed integer program; LP relaxation; Lagrangean Relaxation; Dantzig-Wolfe decomposition; Benders' decomposition; aggregation; projection; modeling issues

Contents

1 Introduction 2

2 LP Relaxations 3

General Cutting Planes 3

Cutting Planes Exploiting Structure 6

3 Lagrangean Relaxation 8

4 Decomposition Methods 9

Dantzig-Wolfe Decomposition 10

Benders' Decomposition 12

5 Reformulations 13

6 Final Remarks 15

Glossary

aggregation	technique to reduce the size of a model
basis of $A \in \mathbb{R}^{m\times n}$	a set $B \subseteq \{1,\dots,n\}$, $\lvert B\rvert = m$, such that A_B is regular
Benders' decomposition	relaxation of a problem by deleting part of the variables and reintroduce them via cutting planes
column generation	technique to solve large problems by generating columns on demand
$\mathrm{cone}(V)$	cone generated by the vectors $v \in V \subseteq \mathbb{R}^n$
$\mathrm{conv}(V)$	convex hull of the vectors $v \in V \subseteq \mathbb{R}^n$
cover	a set $S \subseteq N$ with $\sum_{i\in S} a_i > b$, where N is a finite set and $b, a_i \geq 0, i \in N$
Dantzig-Wolfe decomposition	relaxation of a problem by deleting part of the constraints and reintroduce them via column generation
$f(\alpha)$	fractional part of $\alpha \in \mathbb{R}$, i. e., $f(\alpha) = \alpha - \lfloor\alpha\rfloor$
flow cover	special form of a cover for mixed integer sets
Gomory cut	general cutting plane for (mixed) integer programs
graph G	is a pair $G = (V, E)$, where V is a finite set and E is a family of unordered/ordered pairs of elements from V
halfspace	$\{x \in \mathbb{R}^n : a^T x \leq \alpha\}$ for some $a \in \mathbb{R}^n \setminus \{0\}$, $\alpha \in \mathbb{R}$
$L(\lambda)$	Lagrangean function
Lagrangean function	objective function of a Lagrangean relaxation
Lagrangean relaxation	relaxation of a problem by deleting part of the constraints and reintroduce them in the objective function
lift-and-project cuts	general cutting planes for $0/1$ mixed integer programs
Linear Program	a problem of the form $\min\{c^T x : Ax \leq b, x \in \mathbb{R}^n\}$
MIR inequality	general cutting plane for mixed integer programs
Mixed Integer Program	a problem of the form $\min\{c^T x : Ax \leq b, x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p\}$
polyhedron	intersection of finitely many halfspaces that is $= \{x \in \mathbb{R}^n : Ax \leq b\}$ for some $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$
polytope	bounded polyhedron
$\mathbb{R}$	real numbers
$\mathbb{R}_+$	non-negative real numbers
$\mathbb{R}^{m\times n}$	set of $m \times n$ matrices
Steiner tree	a set of edges in a graph that spans a given subset of nodes T
subgradient	a vector g satisfying $f(\lambda) - f(\lambda^0) \leq g^T(\lambda - \lambda^0)$ for some given function f and vector λ^0

$\mathbb{Z}$	integer numbers
$\mathbb{Z}_+$	non-negative integer numbers
$0/1$ polytope	polytope with $0/1$ vertices

Summary In this article we present classical methods for the solution of large scale integer optimization problems. Among them are LP relaxations and cutting planes, Lagrangean relaxation, Dantzig-Wolfe and Benders' decomposition as well as ideas from lifting and projection. We also discuss some modeling issues and discuss their influence on the solvability of some large scale problems.

1 Introduction

Optimization deals with the problem of minimizing/maximizing a certain objective subject to some set of side constraints. Such problems appear in everyone's daily life, for instance, when one tries to go from A to B by plain, train, or car as fast as possible or when one tries to buy some special goods available at different stores as cheap as possible. Optimization problems are mathematically modeled by introducing variables reflecting the options/quantities to be determined and by expressing the objective and the side constraints by functions defined on the domains of the variables. Depending on the characteristics of these function one speaks of linear or non-linear optimization problems. If some or all variables are required to be integer the prefix 'integer' is added. In this article we restrict our discussion to linear and integer linear optimization problems, i. e., to optimization problems where the objective function and the set of side constraints are linear functions and where some or all variables must be integer. Such problems are expressed in the following form

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p, \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{m \times n}, c \in \mathbb{R}^n, b \in \mathbb{R}^m$ and $p = \{0, \ldots, n\}$. If p equals zero, (1) is called a *linear program*, or LP for short, if $p = n$ we speak of a (pure) *integer linear program (IP)*, and in all other cases of a *mixed integer program (MIP)*. MIPs are a powerful tool to model many real-world optimization problems. For instance, with $0/1$ variables, i. e., integer variables that are restricted to take values zero or one, one can model decisions like do we produce product i (yes or no), do we go from A to B by train (yes or no), do we open a facility at location i (yes or no)? For such kind of questions we introduce a variable which we set to one, if we say 'yes' and which we set to zero, if we say 'no'. The main source of $0/1$ linear programs comes from combinatorial optimization problems, **see Combinatorial Optimization and Integer Programming**.

The application areas in which MIPs occur are huge, ranging from telecommunication, VLSI-design, production and energy planning to problems in traffic and transport or scheduling. And the number of applications is still increasing.

In this article we concentrate on large scale MIPs. The term 'large scale' is relative. Its meaning changes with time. What has been considered 'large scale' a couple of years ago is now no longer large scale. Consider, for instance, the traveling salesman problem (TSP), the problem of determining a minimal tour through a given number of cities. In the fifties a TSP through 49 cities in the US (which corresponds to 1176 variables in the standard IP formulation) has been considered large scale, today the world record of solving a TSP is 13 509 cities (or 91 239 786 variables). Interesting to note is that the method that has been used to solve the problem to optimality at that time is basically the same as the one used today, although various new insights and theoretical results improved this method substantially.

In addition, 'large scale' does not only depend on the number of variables or rows. Very often, problems are considered 'large scale' even if these numbers are moderate, but contain certain structures that are considered difficult for current methods. Therefore we focus in this article on methods that have been used and are still used to solve problems considered large rather than on the presentation of some specific large scale models and their solution techniques.

The basic idea of all methods is to get rid of the part of the problem that makes it difficult. The methods differ in which parts to delete and in the way to reintroduce the deleted parts. In Section 2 we consider linear programming relaxations. Here the integrality constraints on the variables are deleted and the resulting linear program is strengthened by cutting planes. In Section 3 part of the constraint matrix is deleted and put into the objective function attached with some penalties. In Section 4 we discuss decomposition methods, in particular Dantzig-Wolfe and Benders' decomposition. These methods also delete part of the constraint matrix, reformulate this part and reintroduce the reformulated part into the constraint matrix. So far we assumed that the problem formulation of (1) is given. However, the one and the same problem can often be modeled in different ways and the methods discussed in Sections 2 through 4 solve sometimes one formulation better than others. In Section 5 we discuss some reformulation techniques, among others aggregation and projection, and show their influence on the solution quality for some examples.

2 LP Relaxations

If we relax the integrality constraints in (1) we obtain the so-called linear programming relaxation of (1):

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{R}^n. \end{aligned} \tag{2}$$

For the solution of linear programs polynomial and efficient methods are known, **see Linear Programming**. In case the optimal LP solution x^* is integral, we solved (1). Otherwise there must be some inequality (called cutting plane) that separates x^* from $P_I = \mathrm{conv}(\{x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p \mid Ax \leq b\})$. The problem of finding such inequalities is called the separation problem. If we find such an inequality we strengthen the LP relaxation by adding this inequality to the LP and continue. Either we find an optimal solution this way or, if we do not find further inequalities and the optimal LP solution is still fractional, we embed the whole procedure in an enumeration scheme. Details of this so-called cutting plane or branch-and-cut method can be found in **Combinatorial Optimization and Integer Programming**.
The key for the success of this method is to find/know good cutting planes for the polyhedron under consideration. In the article **Combinatorial Optimization and Integer Programming** such inequalities are presented, mainly for $0/1$ polytopes resulting from applications in combinatorial optimization. We supplement this approach by representing some of the inequalities that are helpful for mixed integer problems, i. e., inequalities that combine integer with continuous variables. We first discuss ways of generating cutting planes independent of any problem structure. We then look at MIPs with some local structure.

2.1 General Cutting Planes

In the article **Combinatorial Optimization and Integer Programming** one particular class of inequalities which can be applied independent of any problem structure has already been discussed for pure integer programs, namely Gomory cuts. Consider again the situation discussed there, where we are given an integer program $\max\{c^T x \mid Ax = b,\ x \in \mathbb{Z}^n_+\}$ and an optimal LP solution $x^*_N = 0$ and $x^*_B = A_B^{-1} b - A_B^{-1} A_N x^*_N$, where $B \subseteq \{1, \dots, n\}, |B| = m$, and $N = \{1, \dots, n\} \setminus B$. Consider an index $i \in B$ with $x^*_i \notin \mathbb{Z}$. We use the following abbreviations $\bar{a}_j = A_{i\cdot}^{-1} A_{\cdot j}$, $\bar{b} = A_{i\cdot}^{-1} b, f_j = f(\bar{a}_j), f_0 = f(\bar{b})$, where $f(\alpha) = \alpha - \lfloor \alpha \rfloor$ and $\lfloor \alpha \rfloor$ is the largest integer less than or equal to $\alpha \in \mathbb{R}$. Here $A_{i\cdot}$ denotes the i-th row of matrix A and $A_{\cdot j}$ the j-th column. From the fact

$$A_B^{-1} b - A_B^{-1} A_N x_N \in \mathbb{Z} \tag{3}$$

we derive the Gomory cut, **see Combinatorial Optimization and Integer Programming**

$$\sum_{j \in N} f_j x_j \geq f_0. \tag{4}$$

It is valid for $P_I = \mathrm{conv}\{x \in \mathbb{Z}^n_+ \mid Ax = b\}$ and cuts off x^*. This inequality is no longer valid if continuous variables are involved, because adding integer multiples to continuous variables is no longer possible. For instance, $\frac{1}{3} + \frac{1}{3}x_1 - 2x_2 \in \mathbb{Z}$ with $x_1 \in \mathbb{Z}_+, x_2 \in \mathbb{R}_+$ has a larger solution set than

$\frac{1}{3} + \frac{1}{3}x_1 \in \mathbb{Z}$. Nevertheless, it is possible to derive valid inequalities using the following *disjunctive argument*.

Observation 1 *Let $(a^k)^T x \leq \alpha^k$ be a valid inequality for a polyhedron P^k for $k = 1, 2$. Then,*

$$\sum_{i=1}^{n} \min(a_i^1, a_i^2) x_i \leq \max(\alpha^1, \alpha^2)$$

is valid for both $P^1 \cup P^2$ and $\mathrm{conv}(P^1 \cup P^2)$.

This observation, applied in different ways, yields valid inequalities for the mixed integer case. We present three methods that are all more or less based on Observation 1.

GOMORY'S MIXED INTEGER CUTS.

Consider again the situation in (3). Expression (3) is equivalent to $\sum_{j\in N} \bar{a}_j x_j = f_0 + k$ for some $k \in \mathbb{Z}$. We distinguish two cases, $\sum_{j\in N} \bar{a}_j x_j \geq 0$ and $\sum_{j\in N} \bar{a}_j x_j \leq 0$. In the first case,

$$\sum_{j\in N^+} \bar{a}_j x_j \geq f_0$$

must hold, where $N^+ = \{j \in N : \bar{a}_j \geq 0\}$ and $N^- = N \setminus N^+$. In the second case, we have $\sum_{j\in N^-} \bar{a}_j x_j \leq f_0 - 1$, which is equivalent to

$$-\frac{f_0}{1-f_0} \sum_{j\in N^-} \bar{a}_j x_j \geq f_0.$$

Now we apply Observation 1 to the disjunction $P^1 = P_I \cap \{x : \sum_{j\in N} \bar{a}_j x_j \geq 0\}$ and $P^2 = P_I \cap \{x : \sum_{j\in N} \bar{a}_j x_j \leq 0\}$ with $P_I = \mathrm{conv}\{x \in \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p} : Ax = b\}$. We obtain the valid inequality

$$\sum_{j\in N^+} \bar{a}_j x_j - \frac{f_0}{1-f_0} \sum_{j\in N^-} \bar{a}_j x_j \geq f_0. \tag{5}$$

This inequality may be strengthened in the following way. Observe that the derivation of (5) remains unaffected when adding integer multiples to integer variables. By doing this we may put each integer variable either in the set N^+ or N^-. If a variable is in N^+, the final coefficient in (5) is $\bar{a}_j$ and thus the best possible coefficient after adding integer multiples is $f_j = f(\bar{a}_j)$. In N^- the final coefficient in (5) is $-\frac{f_0}{1-f_0}\bar{a}_j$ and thus $\frac{f_0(1-f_j)}{1-f_0}$ is the best choice. Overall, we obtain the best possible coefficient by using $\min(f_j, \frac{f_0(1-f_j)}{1-f_0})$. This yields Gomory's mixed integer cut

$$\begin{array}{llllll}
\displaystyle\sum_{\substack{j:\, f_j \leq f_0 \\ j \text{ integer}}} & f_j x_j & + & \displaystyle\sum_{\substack{j:\, f_j > f_0 \\ j \text{ integer}}} & \frac{f_0(1-f_j)}{1-f_0} x_j & + \\
\displaystyle\sum_{\substack{j\in N^+ \\ j \text{ non-integer}}} & \bar{a}_j x_j & - & \displaystyle\sum_{\substack{j\in N^- \\ j \text{ non-integer}}} & \frac{f_0}{1-f_0} \bar{a}_j x_j & \geq \quad f_0.
\end{array} \tag{6}$$

It can be shown that an algorithm based on iteratively adding these inequalities solves $\min\{c^T x : x \in P_I\}$ in a finite number of steps provided $c^T x \in \mathbb{Z}$ for all $x \in X$. Note also that (6) is at least as strong as (4) in the pure integer case.

MIXED-INTEGER-ROUNDING CUTS.

Consider the following basic mixed integer set $X = \{(x, y) \in \mathbb{Z} \times \mathbb{R}_+ : x - y \leq b\}$ with $b \in \mathbb{R}$ and the inequality

$$x - \frac{1}{1 - f(b)} y \leq \lfloor b \rfloor. \tag{7}$$

Inequality (7) is valid for $P_I = \mathrm{conv}(X)$. The validity of this inequality is illustrated in Figure 1. A formal proof can be obtained by applying Observation 1 to the disjunction $P^1 = P_I \cap \{(x, y) : x \leq \lfloor b \rfloor\}$ and $P^2 = P_I \cap \{(x, y) : x \geq \lfloor b \rfloor + 1\}$.

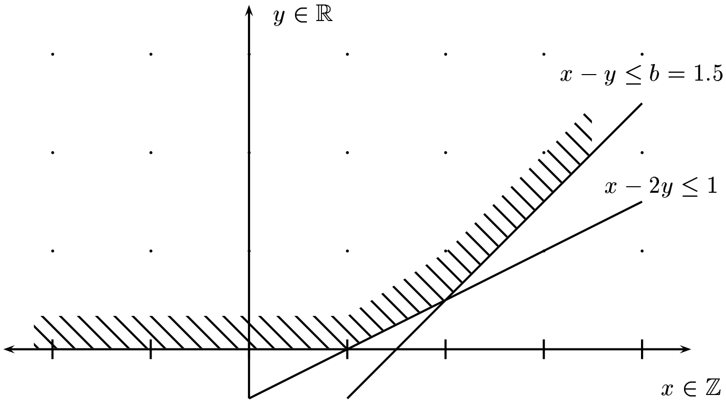

Figure 1: Illustration of basic MIR inequality (7)

The two-dimensional case can be generalized to higher dimensions. Consider the following mixed integer set

$$X = \{(x, y) \in \mathbb{Z}^n_+ \times \mathbb{R}_+ : a^T x - y \leq b\},$$

with $a \in \mathbb{Z}^n, b \in \mathbb{R}$. We take $f_i = f(a_i)$ and $f_0 = f(b)$ in the sequel. The inequality

$$\sum_{i=1}^{n} \left(\lfloor a_i \rfloor + \frac{(f_i - f_0)^+}{1 - f_0} \right) x_i - \frac{1}{1 - f_0} y \leq \lfloor b \rfloor \tag{8}$$

is called a *mixed integer rounding (MIR) inequality*, where $v^+ = \max(0, v)$ for $v \in \mathbb{R}$. It is valid for $P_I = \mathrm{conv}(X)$. To see this apply the two-dimensional inequality (7) to the relaxation $w - z \leq b$ of $a^T x - y \leq b$, where $w = \sum_{\{i \in N : f_i \leq f_0\}} \lfloor a_i \rfloor x_i + \sum_{\{i \in N : f_i > f_0\}} \lceil a_i \rceil x_i \in \mathbb{Z}$ and $z = y + \sum_{\{i \in N : f_i > f_0\}} (1 - f_i) x_i \geq 0$.
MIR inequalities imply Gomory's mixed integer cuts (6) when applied to the mixed integer set $X = \{(x, y^-, y^+) \in \mathbb{Z}^n_+ \times \mathbb{R}^2_+ : a^T x + y^+ - y^- = b\}$. To see this consider the relaxation $a^T x - y^- \leq b$ of X. Applying (8) yields

$$\sum_{i=1}^{n} (\lfloor a_i \rfloor + \frac{(f_i - f_0)^+}{1 - f_0}) x_i - \frac{1}{1 - f_0} y^- \leq \lfloor b \rfloor.$$

Subtracting the original inequality $a^T x + y^+ - y^- = b$ gives Gomory's mixed integer cut (6). MIR inequalities provide a complete description for any mixed $0/1$ polyhedron.

LIFT-AND-PROJECT CUTS.

The idea of "lift and project" is to consider the integer programming problem, not in the original space, but in some space of higher dimension (lifting). Then inequalities found in this higher dimensional space are projected back to the original space resulting in tighter integer programming formulations. Versions of this approach differ in how the lifting and the projection are performed. All approaches only apply to $0/1$ mixed integer programming problems. The validity of the procedure is based on an easy observation.

Observation 2 *If $c_0 + c^T x \geq 0$ and $d_0 + d^T x \geq 0$ are valid inequalities for X, then $(c_0 + c^T x)^T(d_0 + d^T x) \geq 0$ is valid for X.*

Consider a $0/1$ integer program $\min\{c^T x : x \in X\}$ with $X = \{x \in \{0,1\}^p \times \mathbb{R}^{n-p} : Ax \leq b\}$, in which the system $Ax \leq b$ already contains the trivial inequalities $0 \leq x_i \leq 1$ for $i = 1, \ldots, p$. Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ and $P_I = \mathrm{conv}(X)$. Consider the following procedure.

Algorithm 3 *(Lift-and-Project)*

1. *Select an index $j \in \{1, \ldots, p\}$.*
2. *Multiply $Ax \leq b$ by x_j and $1 - x_j$ giving*

$$\begin{aligned} (Ax)x_j &\leq bx_j \\ (Ax)(1 - x_j) &\leq b(1 - x_j) \end{aligned} \tag{9}$$

   and substitute $y_i := x_i x_j$ for $i = 1, \ldots, n,\ i \neq j$ and $x_j := x_j^2$ (lifting).
   Call the resulting polyhedron $L_j(P)$.

3. *Project $L_j(P)$ back to the original space by eliminating variables y_i. Call the resulting polyhedron P_j.*

It can be shown that the j-th component of each vertex of P_j is either zero or one. Now apply Algorithm 3 to P_j by selecting some other index. After repeating this procedure n times, P_I is obtained.
The problem that remains in order to implement Algorithm 3 is to carry out Step 3. Let $L_j(P) = \{(x, y) : Dx + By \leq d\}$. Then the projection of $L_j(P)$ onto the x-space can be described by

$$P_j = \{x : (u^T D)x \leq u^T d \text{ for all } u \in C\},$$

where $C = \{u : u^T B = 0,\ u \geq 0\}$. Thus, the problem of finding a valid inequality in Step 3 of Algorithm 3 that cuts off a current (fractional) solution x^* can be solved by the linear program

$$\begin{array}{ll} \max & u^T(Dx^* - d) \\ & u \in C. \end{array} \tag{10}$$

C is a polyhedral cone and thus the linear program (10) is unbounded, if there is a violated inequality. For algorithmic convenience C is often truncated by some "normalizing set". If an integer variable x_j that attains a fractional value in a basic feasible solution is used to determine the index j in Algorithm 3, then an optimal solution to (10) indeed cuts off x^*.
Observation 2 can be applied to a more general setting by multiplying $Ax \leq b$ not only with x_j and $1 - x_j$, but with products of higher order of the form $\left(\prod_{j \in J_1} x_j\right)\left(\prod_{j \in J_2}(1 - x_j)\right)$ such that $J_1, J_2 \subseteq \{1, \ldots, n\}$ are disjoint and $|J_1 \cup J_2| = d$ for some fixed value $d \geq 1$.
We want to emphasize here that in contrast to the pure integer case none of the cutting plane procedures presented yields a finite algorithm for general mixed integer programs. It is still an open question whether such a procedure exists.

2.2 Cutting Planes Exploiting Structure

The inequalities discussed so far can be applied to any (mixed) integer program. They do not exploit any problem structure. Now we present some inequalities that are derived by concentrating on certain local structures. We illustrate this concept of obtaining inequalities from local structures on one example, the knapsack structure, i. e., we restrict our attention to a single constraint.

The concept of a cover plays a prominent role in this context. We first show how to derive cover inequalities for the $0/1$ knapsack set. We then discuss how to extend these inequalities to more complex mixed integer sets. Consider the $0/1$ knapsack set

$$P_K = \{x \in \{0,1\}^N : \sum_{j \in N} a_j x_j \leq b\}$$

with non-negative coefficients, i. e., $a_j \geq 0$ for $j \in N$ and $b \geq 0$. The set $C \subseteq N$ is a *cover* if

$$\lambda = \sum_{j \in C} a_j - b > 0. \tag{11}$$

In addition, the cover C is said to be *minimal* if $a_j \geq \lambda$ for all $j \in C$. To each cover C, we can associate the *cover inequality*

$$\sum_{j \in C} x_j \leq |C| - 1, \tag{12}$$

stating that in a feasible solution not all variables x_j for $j \in C$ can have value one. If C is minimal, then the inequality (12) defines a facet of $\mathrm{conv}(P_K \cap \{x : x_j = 0,\ j \in N \backslash C\})$.
If a cover C is not minimal, then it is easily seen that the corresponding cover inequality is redundant, i. e., it is the sum of a minimal cover inequality and some upper bound constraints. Generalizations of cover inequalities have been derived for the $0/1$ knapsack set with generalized upper bounds constraints, the $0/1$ knapsack with precedence constraints and the multiple $0/1$ knapsack set. Cover inequalities have been used successfully in general purpose branch-and-cut algorithms to tighten the formulation of $0/1$ integer programs.
Consider now a mixed $0/1$ knapsack set

$$X = \{(x, s) \in \{0,1\}^N \times \mathbb{R}_+ : \sum_{j \in N} a_j x_j \leq b + s\} \tag{13}$$

with non-negative coefficients, i. e., $a_j \geq 0$ for $j \in N$ and $b \geq 0$. Let $C \subseteq N$ be a cover, i. e., C is a subset of N satisfying (11). The inequality

$$\sum_{j \in C} \min{(a_j, \lambda)}\, x_j \leq \sum_{j \in C} \min{(a_j, \lambda)} - \lambda + s \tag{14}$$

is valid for X. Moreover, the inequality (14) defines a facet of $\mathrm{conv}(X \cap \{x :\ x_j = 0,\ j \in N \backslash C\})$.
Observe that each cover C gives rise to a facet-defining inequality, in contrast to the pure integer case where only minimal covers induce facets.
Cover inequalities of the form (14) can be used to derive valid inequalities for more complex mixed integer sets. We illustrate this observation by showing how to derive valid inequalities for an elementary flow model, compare model (FLP) in **Combinatorial and Integer Programming**. Consider the set

$$X = \{(x, y) \in \{0,1\}^N \times \mathbb{R}^N_+ : \sum_{j \in N} y_j \leq b,\ y_j \leq u_j x_j,\ j \in N\},$$

and let $C \subseteq N$ be a (flow) cover, i. e., C is a subset of N satisfying (11). In $\sum_{j \in N} y_j \leq b$, ignore y_j for $j \in N \backslash C$ and replace y_j by $u_j x_j - s_j$ for $j \in C$ where $s_j \geq 0$ is a slack variable. We obtain

$$\sum_{j \in C} u_j x_j \leq b + \sum_{j \in C} s_j.$$

With (11) the following inequality is valid for X

$$\sum_{j \in C} \min{(u_j, \lambda)}\, x_j \leq \sum_{j \in C} \min{(u_j, \lambda)} - \lambda + \sum_{j \in C} s_j,$$

or equivalently, substituting $u_j x_j - y_j$ for s_j,

$$\sum_{j \in C} \left[y_j + (u_j - \lambda)^+ (1 - x_j) \right] \leq b. \tag{15}$$

Inequality (15) is called *flow cover inequality*. It is facet-defining for X.
Flow models have been extensively studied in the literature. Various generalizations of the flow cover inequality have been derived for more complex flow models, in particular for lot-sizing and capacitated facility location problems. Flow cover inequalities have been used successfully in general purpose branch-and-cut algorithms to tighten formulations of mixed integer sets.

3 Lagrangean Relaxations

In the preceding section we have simplified the mixed integer program by relaxing the integrality constraints and by trying to force the integrality of the solution by adding cutting planes. In the method we are going to discuss now we keep the integrality constraints, but relax part of the constraint matrix that causes difficulties. The deleted part is reintroduced into the problem by putting it into the objective function attached with some penalties. Consider again (1). Split A and b into two parts $A = \binom{A_1}{A_2}$ and $b = \binom{b_1}{b_2}$, where $A_1 \in \mathbb{R}^{m_1 \times n}, A_2 \in \mathbb{R}^{m_2 \times n}, b_1 \in \mathbb{R}^{m_1}, b_2 \in \mathbb{R}^{m_2}$ with $m_1 + m_2 = m$. Then, (1) reads

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & A_1 x \leq b_1 \\ & A_2 x \leq b_2 \\ & x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p. \end{aligned} \tag{16}$$

Consider for some fixed $\lambda \in \mathbb{R}^{m_1}, \lambda \geq 0$ the following function

$$\begin{aligned} L(\lambda) = \quad \min \quad & c^T x - \lambda^T (b_1 - A_1 x) \\ \text{s.t.} \quad & x \in P^2, \end{aligned} \tag{17}$$

where $P^2 = \{x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p : A_2 x \leq b_2\}$. (17) is called the *Lagrangean function*. Obviously, (17) is a lower bound for (16), since for any feasible solution $\bar{x}$ of (16) we have

$$c^T \bar{x} \geq c^T \bar{x} - \lambda^T (b_1 - A_1 \bar{x}) \geq \min_{x \in P^2} c^T x - \lambda^T (b_1 - A_1 x) = L(\lambda)$$

Since this holds for each $\lambda \geq 0$ we get that

$$\max_{\lambda \geq 0} L(\lambda) \tag{18}$$

yields a lower bound for (1). (18) is called *Lagrangean relaxation*. Let λ^* be an optimal solution to (18). The questions remain, how good is $L(\lambda^*)$ and how to compute λ^*. An answer to the first question gives the following theorem.

Theorem 4 $L(\lambda^*) = \min\{c^T x : A_1 x \leq b_1, x \in \operatorname{conv}(P^2)\}$.

Since

$$\{x \in \mathbb{R}^n : Ax \leq b\} \supseteq \{x \in \mathbb{R}^n : A_1 x \leq b_1, \; x \in \operatorname{conv}(P^2)\} \supseteq \operatorname{conv}\{x \in \mathbb{Z}^{n-p} \times \mathbb{R}^p : Ax \leq b\}$$

we conclude form Theorem 4

$$z_{LP} \leq L(\lambda^*) \leq z_{IP}, \tag{19}$$

where z_{LP} is the optimal solution value of the LP relaxation and z_{IP} is the optimal integer solution value. Furthermore, $z_{LP} = L(\lambda^*)$ for all objective functions c if and only if $\{x \in \mathbb{R}^n : Ax \leq b\}$ is integral.

It remains to discuss how to compute $L(\lambda^*)$. From a theoretical point of view it can be shown using the equivalence of separation and optimization that $L(\lambda^*)$ can be determined in polynomial time, if $\min\{\tilde{c}^T x \,:\, x \in \mathrm{conv}(P^2)\}$ can be computed in polynomial time for any objective function $\tilde{c}$. In practice, $L(\lambda^*)$ is determined by applying subgradient methods. The function $L(\lambda)$ is piecewise linear, concave and bounded from above. Consider for some fixed $\lambda^0 \in \mathbb{R}^{m_1}_+$ an optimal solution x^0 for (17). Then, $g^0 = A_1 x - b_1$ is a subgradient for L in λ^0, i. e., $L(\lambda) - L(\lambda^0) \leq (g^0)^T(\lambda - \lambda^0)$, since $L(\lambda) - L(\lambda^0) = c^T x^\lambda - \lambda^T(b_1 - A_1 x^\lambda) - (c^T x^0 - (\lambda^0)^T(b_1 - A_1 x^0)) \leq c^T x^0 - \lambda^T(b_1 - A_1 x^0) - (c^T x^0 - (\lambda^0)^T(b_1 - A_1 x^0)) = (g^0)^T(\lambda - \lambda^0)$.
Hence, for λ^* we have $(g^0)^T(\lambda - \lambda^0) \geq L(\lambda^*) - L(\lambda^0) \geq 0$. This suggests in order to find λ^* to start with some λ^0, determine $x^0 = \mathrm{argmin}\{c^T x - (\lambda^0)^T(b_1 - A_1 x) \,:\, x \in P^2\}$ and determine iteratively, $\lambda^0, \lambda^1, \lambda^2, \ldots$ by setting $\lambda^{k+1} = \lambda^k + \mu^k g^k$, where μ^k is some step length to be specified. This iterative method is the essence of the *subgradient method*. Details and refinements of this method can be found in the article on **Nondifferential Optimization**.
Of course, the quality of the Lagrangean relaxation strongly depends on the set of constraints that is relaxed. On one side, we must compute (17) for various values of λ and thus it is necessary to determine one value of $L(\lambda)$ fast. Therefore one may want to relax as many (complicated) constraints as possible. On the other hand, the more constraints are relaxed the worse the bound $L(\lambda^*)$ will get. Therefore, one always must find a compromise between these two conflicting goals. In the following we give some applications where this method has been successfully applied and a good balance between these two opposite objectives can be found.
One application has already been presented in the article **Combinatorial and Integer Programming**. By relaxing the degree constraints in the integer programming formulation for the traveling salesman problem, we are left with a spanning tree problem, which can be solved fast by the greedy algorithm. A main advantage of this TSP relaxation is that for the evaluation of (17) combinatorial algorithms are at hand and no general LP or IP solution techniques must be used. Lagrangean relaxation is also very often used if the underlying linear programs of (1) are just too big to be solved directly and even the relaxed problems in (17) are still large. Often the relaxation can be done in a way that the evaluation of (17) can be solved combinatorially. Other examples are multicommodity flow problems arising for instance in vehicle scheduling or scenario decompositions of stochastic mixed integer programs. In fact, the latter two applications fall into a class of problems where the underlying matrix has bordered block diagonal form, see Figure 2. If we relax the coupling constraints within a Lagrangean relaxation, the remaining matrix decomposes into k independent blocks. Thus, one value $L(\lambda)$ is the sum of k individual terms that can be determined separately. Often each single block A_i models a network flow problem, a knapsack problem or the like and can thus be solved using special purpose combinatorial algorithms.

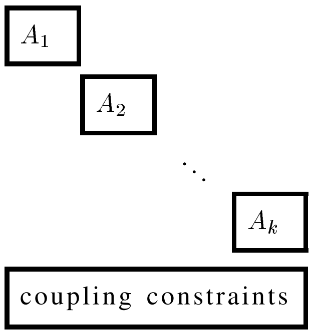

Figure 2: Matrix in bordered block diagonal form

4 Decomposition Methods

The idea of decomposition methods is to decouple a set of constraints (variables) from the problem and treat them at a superordinate level, often called master problem. The resulting residual subordinate problem can often be solved more efficiently. Decomposition methods now work alternately on the master and subordinate problem and iteratively exchange information to solve the original problem to optimality. In this section we discuss two well known examples of this approach, Dantzig-Wolfe decomposition and Benders' decomposition. We will see that as in the case of Lagrangean relaxation these methods also delete part of the constraint matrix. But instead of reintroducing this part in the objective function, it is now reformulated and reintroduced into the constraint system.

4.1 Dantzig-Wolfe Decomposition

Let us start with Dantzig-Wolfe decomposition and consider again (16), where we assume for the moment that $p = n$ and all constraints are equality constraints. Consider the polyhedron $P^2 = \{x \in \mathbb{R}^n : A_2 x \leq b_2\}$. It is a well known fact in the theory of polyhedra that there exist vectors $v_1, \dots, v_k$ and $e_1, \dots, e_l$ such that $P^2 = \operatorname{conv}(\{v_1, \dots, v_k\}) + \operatorname{cone}(\{e_1, \dots, e_l\})$. In other words, $x \in P^2$ can be written in the form

$$x = \sum_{i=1}^{k} \lambda_i v_i + \sum_{j=1}^{l} \mu_j e_j \tag{20}$$

with $\lambda_1, \dots, \lambda_k \geq 0, \sum_{i=1}^{k} \lambda_i = 1$ and $\mu_1, \dots, \mu_l \geq 0$. Substituting for x from (20) we may write (16) as

$$\begin{array}{lll}
\min & c^T(\sum_{i=1}^{k} \lambda_i v_i + \sum_{j=1}^{l} \mu_j e_j) & \\
\text{s.t.} & A_1(\sum_{i=1}^{k} \lambda_i v_i + \sum_{j=1}^{l} \mu_j e_j) \leq b_1 & \\
& \sum_{i=1}^{k} \lambda_i \qquad\qquad\qquad = 1 & \\
& \qquad \lambda \in \mathbb{R}^k_+, \mu \in \mathbb{R}^l_+, &
\end{array}$$

which is equivalent to

$$\begin{array}{ll}
\min & \sum_{i=1}^{k} (c^T v_i)\, \lambda_i + \sum_{j=1}^{l} (c^T e_j)\, \mu_j \\
\text{s.t.} & \sum_{i=1}^{k} (A_1 v_i)\, \lambda_i + \sum_{j=1}^{l} (A_1 e_j)\, \mu_j \leq b_1 \\
& \sum_{i=1}^{k} \lambda_i \qquad\qquad\qquad\qquad = 1 \\
& \qquad \lambda \in \mathbb{R}^k_+, \mu \in \mathbb{R}^l_+.
\end{array} \tag{21}$$

(21) is called the *master problem* of (16). Comparing formulations (16) and (21) we see that we reduced the number of constraints from m to m_1, but obtain $k + l$ variables instead of n. $k + l$ might be large compared to n, in fact even exponential (consider for example the unit cube in $\mathbb{R}^n$ with $2n$ constraints and 2^n vertices) so that there seems to be at first sight no gain in using formulation (21). However, we can use the revised simplex algorithm for the solution of (21). For ease of exposition abbreviate (21) by $\min\{w^T \eta : D\eta = d\}$ with $D \in \mathbb{R}^{(m_1+1)\times(k+l)}$, $d \in \mathbb{R}^{m_1+1}$. Recall from the article **Linear Programming** that the simplex algorithm starts with a (feasible) basis $B \subseteq \{1, \dots, k + l\}$, $|B| = m_1 + 1$, with D_B non-singular and the corresponding (feasible) solution $\eta^*_B = D_B^{-1} d$ and $\eta^*_N = 0$, where $N = \{1, \dots, k + l\} \setminus B$. Observe that $D_B \in \mathbb{R}^{(m_1+1)\times(m_1+1)}$ is (much) smaller than a basis for the original system (16) and that only a fraction of the variables ($m_1 + 1$ out of $k + l$) are possibly non-zero. In addition, on the way to an optimal solution the only operation within the simplex method that involves all columns is the pricing step, where it is checked whether the reduced costs $w_N - \tilde{y}^T D_N$ are non-negative with $\tilde{y}$ being the solution of $y^T D_B = w_B$. The non-negativity of

the reduced costs can be verified via the following linear program:

$$\begin{array}{rl} \max & (c^T - \bar{y}^T A_1)x \\ \text{s.t.} & A_2 x \le b_2 \\ & x \in \mathbb{R}^n, \end{array} \tag{22}$$

where $\bar{y}$ are the first m_1 components of the solution of $\tilde{y}$. The following cases might come up:

(i) (22) has an optimal solution $\tilde{x}$ with $(c^T - \bar{y}^T A_1)\tilde{x} < \tilde{y}_{m_1+1}$.
In this case, $\tilde{x}$ is one of the vectors $v_i, i \in \{1, \dots, k\}$, with corresponding reduced cost $w_i - \tilde{y}^T D_{\cdot i} = (c^T v_i - \tilde{y}^T \binom{A_1 v_i}{1}) = c^T v_i - \bar{y}^T A_1 v_i - \tilde{y}_{m_1+1} < 0$. In other words, $\binom{A_1 v_i}{1}$ may serve as the entering column within the simplex algorithm.

(ii) (22) is unbounded.
Here we obtain a feasible extreme ray e^* with $(c^T - \bar{y}^T A_1)e^* < 0$. e^* is one of the vectors $e_j, j \in \{1, \dots, l\}$. It yields a column $\binom{A_1 e_j}{0}$ with reduced cost $w_{k+j} - D_{\cdot(k+j)} = c^T e_j - \tilde{y}^T \binom{A_1 e_j}{0} = c^T e_j - \bar{y}^T (A_1 e_j) < 0$. That is, $\binom{A_1 e_j}{0}$ is a possible entering column.

(iii) (22) has an optimal solution $\tilde{x}$ with $(c^T - \bar{y}^T A_1)^T \tilde{x} \ge \tilde{y}_{m_1+1}$.
In this case we conclude using the same arguments as in (i) and (ii) that $w_i - \tilde{y}^T D_{\cdot i} \ge 0$ for all $i = 1, \dots, k + l$ proving that x^* is an optimal solution for the master problem (21).

Observe that the whole problem (16) is decomposed into two problems, i. e., (21) and (22), and the approach iteratively works on the master level (21) and the subordinate level (22). The procedure starts with some feasible solution for (21) and generates new promising columns on demand by solving (22). Such procedures are commonly called *column generation* or *delayed column generation algorithms*.

The approach can also be extended to general integer programs with some caution. In this case problem (22) turns from a linear to an integer linear program. In addition, we have to guarantee in (20) that all feasible integer solutions x of (16) can be generated by (integer) linear combinations of the vectors $v_1, \dots, v_k$ and $e_1, \dots, e_l$. Hereto, it is not sufficient to require λ and μ to be integer. Consider as an example the problem $\max\{x_1 + x_2 : A_1 x \le b_1, A_2 x \le b_2, x \in \{0, 1, 2\}^2\}$ with $A_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $b_1 = \binom{1.5}{1.5}$ and $A_2 = (1, 1), b_2 = 2$. In this case, $P^2 = \text{conv}(\{\binom{0}{0}, \binom{2}{0}, \binom{0}{2}\})$, see Figure 3, but the optimal solution $\binom{1}{1}$ of the integer program is not an integer linear combination of the vertices of P^2. However, with the addition that all variables are $0/1$, this difficulty does not occur, since any $0/1$ solution of some $0/1$ polyhedron is always a vertex of that polyhedron. And in fact, column generation algorithm are not only used for the solution of large linear programs, but especially for large $0/1$ integer programs.

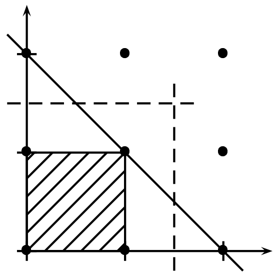

Figure 3: Extending Dantzig-Wolfe decomposition to integer programs

Of course, the presented Dantzig-Wolfe decomposition for linear or $0/1$ integer programs is just one type of column generation algorithms. Others solve the subordinate problem not via general linear of integer programming techniques, but use combinatorial or sort of explicit enumeration algorithms. Furthermore, the problem is often not modeled via (16), but directly as in (21). This is, for instance, the case when the set of feasible solutions have a rather complex description by linear inequalities, but can easily be incorporated into some enumeration scheme.
The application area where Dantzig-Wolfe decomposition or column generation is used is very broad, for instance in airline crew scheduling, vehicle routing, public mass transport, network design, to

name just a few. Of course, also integer programs with bordered block diagonal form, see Figure 2, nicely fit into this context. In contrast to Lagrangean relaxation, where the coupling constraints are relaxed, Dantzig-Wolfe decomposition keeps these constraints in the master problem and relaxes the constraints of the blocks having the advantage that (22) decomposes into independent problems, one for each block.

4.2 Benders' Decomposition

Let us finally turn to *Benders' decomposition*. Benders' decomposition also deletes part of the constraint matrix, but in contrast to Dantzig-Wolfe decomposition, where we delete part of the constraints and reintroduce them via column generation, we now delete part of the variables and reintroduce them via cutting planes. In this respect, Benders' decomposition is dual to Dantzig-Wolfe decomposition. Consider again (1) and write it in the form

$$\begin{aligned} \min \quad & c_1^T x_1 + c_2^T x_2 \\ \text{s.t.} \quad & A_1 x_1 + A_2 x_2 \le b \\ & x_1 \in \mathbb{R}^{n_1}, x_2 \in \mathbb{R}^{n_2}, \end{aligned} \tag{23}$$

where $A = [A_1, A_2] \in \mathbb{R}^{m \times n}$, $A_1 \in \mathbb{R}^{m \times n_1}$, $A_2 \in \mathbb{R}^{m \times n_2}$, $c_1, x_1 \in \mathbb{R}^{n_1}$, $c_2, x_2 \in \mathbb{R}^{n_2}$ with $n_1 + n_2 = n$. Note that we have assumed for ease of exposition the case of a linear program. We will see, however, that what follows is still true if $x_1 \in \mathbb{Z}^{n_1}$. Our intention is to get rid of the variables x_2. These variables prevent (23) from being a pure integer program in case $x_1 \in \mathbb{Z}^{n_1}$. Also in the linear programming case they might be the origin for some difficulties, see the applications below. One well known approach to get rid of variables is projection, see also the lift-and-project cuts in Section 2. In order to apply projection we must slightly reformulate (23) to

$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & -z + c_1^T x_1 + c_2^T x_2 \le 0 \\ & \qquad A_1 x_1 + A_2 x_2 \le b \\ & z \in \mathbb{R},\ x_1 \in \mathbb{R}^{n_1},\ x_2 \in \mathbb{R}^{n_2}, \end{aligned} \tag{24}$$

Now, (24) is equivalent to

$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & -uz + u c_1^T x_1 + v^T A_1 x_1 \le v^T b \\ & z \in \mathbb{R},\ x_1 \in \mathbb{R}^{n_1}, \\ & \tbinom{u}{v} \in C, \end{aligned} \tag{25}$$

where

$$C = \{ \tbinom{u}{v} \in \mathbb{R}^{m+1} : v^T A_2 + u c_2^T = 0,\ u \ge 0, v \ge 0 \}. \tag{26}$$

C is a polyhedral cone, thus there exist vectors $\binom{\bar{u}_1}{\bar{v}_1}, \dots, \binom{\bar{u}_s}{\bar{v}_s}$ such that $C = \operatorname{cone}(\{\binom{\bar{u}_1}{\bar{v}_1}, \dots, \binom{\bar{u}_s}{\bar{v}_s}\})$. These extreme rays can be rescaled such that $\bar{u}_i$ is zero or one. Thus $C = \operatorname{cone}(\{\binom{0}{v_k} : k \in K\}) + \operatorname{cone}(\{\binom{1}{v_j} : j \in J\})$ with $K \cup J = \{1, \dots, s\}$ and $K \cap J = \emptyset$. With this description of C, (25) can be restated as

$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & -z \le c_1^T x_1 + v_k^T (b - A_1 x_1) \quad \text{for all } k \in K, \\ & \ \ 0 \le \qquad\quad v_j^T (b - A_1 x_1) \quad \text{for all } j \in J, \\ & z \in \mathbb{R}, x_1 \in \mathbb{R}^{n_1}. \end{aligned} \tag{27}$$

(27) is called *Benders' master problem*. Benders' master problem has just $n_1 + 1$ variables instead of $n_1 + n_2$ variables in (23), or in case $x_1 \in \mathbb{Z}^{n_1}$ we have reduced the mixed integer program (23) to an almost pure integer program (27) with one additional continuous variable z. However, (27) contains an enormous number of constraints, in general exponentially many in n, see also the discussion on page 10. To get around this problem, we solve Benders' master problem by cutting plane methods, see Section 2 or the article **Combinatorial and Integer Programming**, respectively. We start with a small subset of extreme rays of C (possibly the empty set) and optimize (27) just over this subset. We obtain an optimal solution x^*, z^* of the relaxed problem and we must check whether this solution satisfies all other inequalities in (27). This can be done via the following linear program

$$\begin{aligned} \min \quad & v^T(b - A_1 x_1^*) + u(z^* - c_1^T x_1^*) \\ \text{s.t.} \quad & \tbinom{u}{v} \in C. \end{aligned} \tag{28}$$

(28) is called *Benders' subproblem*. It is feasible, since $\binom{0}{0} \in C$, and (28) has an optimal solution value of zero or it is unbounded. In the first case, x_1^*, z^* satisfies all inequalities in (27) and we have solved (27) and thus (23). In the latter case we obtain an extreme ray $\binom{u^*}{v^*}$ from (28) with $(v^*)^T(b - A_1 x_1^*) + u^*(z^* - c_1^T x_1^*) < 0$ which after rescaling yields a cut for (27) violated by x_1^*, z^*. We add this cut to Benders' master problem (27) and iterate.

Benders' decomposition is very often implicitly used within cutting plane algorithms, see for instance the derivation of lift-and-project cuts in Section 2. Other application areas are problems whose constraint matrix has bordered block diagonal form, where we have coupling variables instead of coupling constraints, see Figure 4, i. e., the structure of the constraints is the transposed of the structure of the matrix in Figure 2. Such problems appear, for instance, in stochastic integer programming. Benders' decomposition is attractive in this case, because Benders' subproblem decomposes into k independent problems.

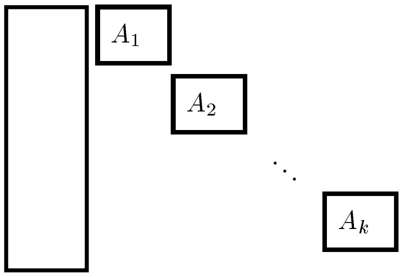

Figure 4: Matrix in bordered block diagonal form with coupling variables

5 Reformulations

In Sections 2 through 4 we have discussed methods for the solution of large scale mixed integer programming problems, where we assumed that the problem formulation is given as in (1). But, there are in general various alternatives to model some practical problem as a mixed integer program or to reformulate mixed integer programs in various ways. We have already discussed some of these reformulation ideas in the previous sections, see, for instance, lift-and-project, Dantzig-Wolfe- or Benders' decomposition.
In this section we put our emphasis on some ideas on how to model practical problems. There are no general rules like the fewer the number of variables and/or constraints the better the formulation that can be stated and applied to any given problem. The model strongly depends on the availability of data, on the solution method that is intended to be used, the quality of possible relaxations that are used within the solution process, and of course also the availability of computer power. In the following we exemplarily discuss models for two different problems. The first is a classical problem in combinatorial optimization, the Steiner tree problem. The second is an example for a planning problem with a certain time horizon. From the Steiner tree problem we will see that fewer variables is not always better, the planning problem shows that some kind of so-called *aggregation* is very often

necessary, since not all data is available or the models just get too big to be solvable by state-of-the-art software.

Let us start with the Steiner tree problem. Given an undirected graph $G = (V, E)$ and a node set $T \subseteq V$, a *Steiner tree for T in G* is a subset $S \subseteq E$ of the edges such that $(V(S), S)$ contains a path from s to t for all $s, t \in T$, where $V(S)$ denotes the set of nodes incident to an edge in S. In other words, a Steiner tree is an edge set S that spans T. The *Steiner tree problem* is to find a minimal Steiner tree with respect to some given edge costs $c_e, e \in E$.
A canonical way to formulate the Steiner tree problem as an integer program is to introduce, for each edge $e \in E$, a variable x_e indicating whether e is in the Steiner tree ($x_e = 1$) or not ($x_e = 0$). Consider the integer program

$$\begin{array}{lll} \min & c^T x & \\ & x(\delta(W)) \geq 1, & \text{for all } W \subset V, W \cap T \neq \emptyset, \\ & & (V \setminus W) \cap T \neq \emptyset, \\ & 0 \leq x_e \leq 1, & \text{for all } e \in E, \\ & x \text{ integer}, & \end{array} \tag{29}$$

where $\delta(X)$ denotes the cut induced by $X \subseteq V$, i. e., the set of edges with one end node in X and one in its complement, and $x(F) := \sum_{e \in F} x_e$, for $F \subseteq E$. The first inequalities are called *(undirected) Steiner cut inequalities* and the inequalities $0 \leq x_e \leq 1$ *trivial inequalities*. It is easy to see that there is a one-to-one correspondence between Steiner trees in G and $0/1$ vectors satisfying the undirected Steiner cut inequalities. Hence, the Steiner tree problem can be solved via (29). Another way to model the Steiner tree problem is to consider the problem in a directed graph. We replace each edge $[u, v] \in E$ by two anti-parallel arcs (u, v) and (v, u). Let A denote this set of arcs and $D = (V, A)$ the resulting digraph. We choose some terminal $r \in T$, which will be called the *root*. A *Steiner arborescence (rooted at r)* is a set of arcs $S \subseteq A$ such that $(V(S), S)$ contains a directed path from r to t for all $t \in T \setminus \{r\}$. Obviously, there is a one-to-one correspondence between (undirected) Steiner trees in G and Steiner arborescences in D which contain at most one of two anti-parallel arcs. Thus, if we choose arc costs $\vec{c}_{(u,v)} := \vec{c}_{(v,u)} := c_{[u,v]}$, for $[u, v] \in E$, the Steiner tree problem can be solved by finding a minimal Steiner arborescence with respect to $\vec{c}$. Note that there is always an optimal Steiner arborescence which does not contain an arc and its anti-parallel counterpart, since $\vec{c} \geq 0$. Introducing variables y_a for $a \in A$ with the interpretation $y_a := 1$, if arc a is in the Steiner arborescence, and $y_a := 0$, otherwise, we obtain the integer program

$$\begin{array}{lll} \min & \vec{c}^T y & \\ & y(\delta^+(W)) \geq 1, & \text{for all } W \subset V, r \in W, \\ & & (V \setminus W) \cap T \neq \emptyset, \\ & 0 \leq y_a \leq 1, & \text{for all } a \in A, \\ & y \text{ integer}, & \end{array} \tag{30}$$

where $\delta^+(X) := \{(u, v) \in A \mid u \in X, v \in V \setminus X\}$ for $X \subset V$, i. e., the set of arcs with tail in X and head in its complement. The first inequalities are called *(directed) Steiner cut inequalities* and $0 \leq y_a \leq 1$ are the *trivial inequalities*. Again, it is easy to see that each $0/1$ vector satisfying the directed Steiner cut inequalities corresponds to a Steiner arborescence, and conversely, the incidence vector of each Steiner arborescence satisfies (30). Which of the two models (29) and (30) should be used to solve the Steiner tree problem in graphs?
At first glance, (29) is preferable to (30), since it contains only half the number of variables and basically the same number of constraints. However, it turns out that the optimal value of the LP relaxation of the directed model $z_d := \min\{\vec{c}^T y \mid y$ satisfies the trivial and directed Steiner cut inequalities$\}$ is greater than or equal to the corresponding value of the undirected formulation $z_u := \min\{c^T x \mid x$ satisfies the trivial and undirected Steiner cut inequalities$\}$. Even, if the undirected formulation is tightened by the so-called Steiner partition inequalities, this relation holds. This is even more astonishing, since the separation problem of the Steiner partition inequalities is difficult ($\mathcal{NP}$-hard), whereas the directed Steiner cut inequalities can be separated in polynomial time by max flow computation. Finally, the disadvantage of the directed model that the number of variables is doubled is not really a bottleneck. Since we are minimizing a non-negative objective function, the

variable of one of two anti-parallel arcs will usually be at its lower bound and rarely touched by the solution algorithm. Thus, the directed model is much better than the undirected, though it contains more variables. And in fact, most state-of-the-art solvers for the Steiner tree problem in graphs use formulation (30) or one that is equivalent to (30).

We finally present a model for a typical planning problem. We will see that such models get huge immediately and that some sort of reduction is necessary to make such models tractable. The kind of reduction that is very often used in this context is called *aggregation*, which we will explain on the following concrete example.
Suppose some consulting company has a certain number of projects P that must be undertaken within a certain time period T. The company has a set of employees E available for this projects. In addition, there are a set of predefined skills S, each employee e has a subset e_S of these skills and each project p has a certain demand $d^t_{p,s}$ of skill s in each period $t \in T$. In addition, there is some cost c_{espt} associated with the decision that employee e with skill s is working for project p in time period t.
To model this problem we introduce $0/1$ variables x_{espt} with the interpretation that $x_{espt} = 1$ if and only if employee e with skill s works for project p in time period t. We get the following model:

$$\begin{aligned}
\min \quad & \sum_{e\in E}\sum_{s\in e_S}\sum_{p\in P}\sum_{t\in T} c_{espt}x_{espt} \\
& \sum_{s\in e_S}\sum_{p\in P} x_{espt} \leq 1 \quad && \text{for all } e \in E, t \in T, \\
& \sum_{e\in E} x_{espt} = d^t_{p,s}, && \text{for all } p \in P, s \in S, t \in T, \\
& x_{espt} \in \{0,1\}, && \text{for all } e \in E, p \in P, s \in S, t \in T.
\end{aligned} \tag{31}$$

The first type of inequality expresses that each employee e can work at most for one project in time period t. The second set of constraints guarantees that the demands for skills in each project are satisfied in time period t. Problem (31) in this form is a classical assignment problem. However, in practice there are many more side constraints that make the problem difficult to solve. For instance, there should be a balance in working time for all employees, employees should continuously work in some project and do not switch from one project to another frequently, each employee should be assigned to his favorite skill, and so forth. If we look on the size of (31) and consider some realistic numbers such as $|T| = 220$, i. e., a time horizon of one year with around 220 working days, $|E| = 50$, i. e., around 50 employees, which is not much, $|P| = 10$, i. e., there are ten projects that are simultaneously undertaken, and $|S| = 5$, i. e., there are around five skills available and necessary to run the projects, we obtain for the number of variables $|T|\cdot|E|\cdot|P|\cdot|S| = 220\cdot 50\cdot 10\cdot 5 = 550000$. That is we obtain around half a million of $0/1$ variables, which makes the problem basically unsolvable for todays software and computer power. Therefore some reductions (in this context also called *aggregations*) are on demand.
One type of aggregation that is frequently used is to model the time horizon coarser. For instance, one possibility is to model the first month of the year on a day basis, to use weekly time intervals later and maybe at the end even one month per time period. This does not only reduce the size of the model, but sometimes is also forced by the data given, since nobody typically plans on a day basis one year ahead. Another aggregation possibility is to group employees into teams that always work together in projects. This does not only reduce the number of variables, but might also be wanted by the company (and the employees), since well functioning teams are more profitable. There are of course many more aggregations imaginable that on one side reduce the size of the problem and on the other hand might even be wanted by the project partner. The two selected examples of aggregation should just show that one has to find the right balance between models that reflect the reality as detailed as possible and models that are tractable by todays software.

6 Final Remarks

We have seen that many skills are necessary to solve large scale (discrete) optimization problems. First and this was the topic of Section 5 one has to find the right model, a model that reflects the reality well enough, that is accepted by the project partner and that is tractable by the solution methods at hand. Hereto and this is the second skill, one must have in mind the algorithms and their behavior on various models that one wants to use to solve the problem. In Sections 2 through 4 we have seen some algorithmic principles to attack large scale problems. In particular, we have discussed LP relaxation, Lagrangean relaxation, Dantzig-Wolfe and Benders' decomposition methods. However, we have just discussed the principle ideas behind these methods. To turn these ideas into an efficient, robust and fast computer program is a long way. To be able to go this way is the third skill one needs. Questions from software engineering, which data structures are to be used, how to exploit the underlying structure of the problem have not been discussed here, but are essential and might decide on the final question, whether a large scale problem is solvable or not.

In this article we have presented classical methods to solve large scale mixed integer programs. In the newer literature further approaches are discussed that have their success for some specific problems, but are not yet in the state to solve large scale problems. Semidefinite programming, combinatorial relaxations, basis reduction, Gomory's group approach, test sets and optimal primal algorithms are some of these promising methods. The future will show whether they can contribute to the challenge of solving large scale problems.

References

[1] Applegate, D., Bixby, R. E., Chvátal, V., and Cook, W. (1998). On the solution of traveling salesman problems. In *Documenta Mathematica Journal der Deutschen Mathematiker-Vereinigung, International Congress of Mathematicians*, pages 645 – 656. [Gives a nice impression on the skills that are necessary to solve large scale TSPs, cf. Section 6.]

[2] Balas, E., Ceria, S., and Cornuéjols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295 – 324. [Contains all the details about lift-and-project cuts discussed in Section 2.1.]

[3] Dell'Amico, M., Maffioli, F., and Martello, S. (1997). *Annotated Bibliographies in Combinatorial Optimization*. Wiley, Chichester. [A good reference book including literature on decomposition and column generation.]

[4] Goemans, M. X. and Myung, Y. (1993). A catalog of Steiner tree formulations. *Networks*, 23:19 – 28. [Compares various formulations for the Steiner tree problem discussed in Section 5.]

[5] Williams, H. P. (1990). *Model building in mathematical programming*. Wiley, Chichester. [The book discusses modeling issues in detail (cf. Section 5) as well as pros and cons of various formulations.]

[6] Martin, R. (1999). *Large scale linear and integer programming: A unified approach*. Kluwer. [This is the book form of this article. The methods described here and many more are extensively discussed.]

[7] Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. Wiley, Chichester. [Complements Schrijver's book on integer programming. It also focuses on practical issues with many illustrating examples.]

[8] Schrijver, A. (1986). *Theory of Linear and Integer Programming*. Wiley, Chichester. [A comprehensive book with excellent historical notes on the theory of linear and integer programming.]