- Master thesis -

Submitted in fulfilment of the requirements for the degree of Master of Science

Separation of Generic Cutting Planes in Branch-and-Price

Jonas T. Witt

First reviewer: Prof. Dr. Marco E. Lübbecke
Second reviewer: Prof. Dr. Arie M.C.A. Koster

September 30, 2013

Department of Mathematics
RWTH Aachen

Contents

Chapter 1

Introduction

Reformulating a given mixed integer program by the use of Dantzig-Wolfe decomposition leads to a potentially stronger linear programming (LP) relaxation than the inital problem formulation and it may reduce the problem's symmetry [1]. The obtained formulation is called *extended formulation*. Since the number of variables can be exponential in the size of the original problem, the LP relaxation of the extended formulation is often solved by applying *column generation* – the dynamic generation of new columns. Solving the LP relaxtion in every node of the branch-and-bound tree by column generation is called *branch-and-price*. If we additionally separate cutting planes during the solution process, the procedure is known as *branch-price-and-cut*.

For several problem classes combinatorial cutting planes were successfully adapted in the context of a branch-price-and-cut algorithm, but most often these cutting planes are problem specific and cannot be applied in general. Separation of cutting planes helps to improve the dual bound in the root node and reduces the number of used branch-and-bound nodes. Additionally, this can accelerate the solution process. In this thesis we discuss the separation of generic cutting planes in branch-and-price. Thereby, we restrict ourselves to the case of solving a reformulated problem via branch-and-price since we want to exploit the original problem formulation. Cutting planes which are formulated in the original problem's solution space can easily be considered in the branch-and-price procedure.

Suppose we want to separate a given solution of the extended formulation's LP relaxation. We can transfer the given solution into the original problem's solution space and separate this solution by applying separation procedures in the original problem.

This was already implemented [2], however, it is not successful at many problem classes. Note that we can only use separators for mixed integer programs which do not use basis information since the transferred solution is in general not basic feasible to the original LP relaxation. This is why we calculate an auxiliary basic feasible solution for the original LP relaxation. We generate cutting planes which cut off this solution and hope that these cutting planes also cut off the given solution, which we wanted to separate initially. This separation procedure was introduced but not tested by Range [3]. We will discuss this separator as weall as some extensions and present experimental results of our implementation, which is done in the generic branch-price-and-cut solver `GCG` [2]. To our knowledge, this is the first implementation of a separator which is used in the context of a branch-price-and-cut algorithm and generates generic cutting planes in the original problem's solution space by the use of basis information.

In the following chapter we give a short introduction to column generation, Dantzig-Wolfe reformulation, and branch-and-price. After an introduction to cutting planes including their potential use in branch-and-price in chapter 3, we present the main ideas of this thesis in chapter 4. We introduce ways to calculate basic feasible solutions for the original LP relaxation, which can be used to generate cutting planes by the use of basis information, and formally define the main separation procedure, which we call *basis separator*. Furthermore, we present valid inequalities which tighten the original LP relaxation and can be used in combination with the basis separator. At the end, we evaluate the performance of the basis separator including its variations and extensions in chapter 5, before we summarize the results of this thesis and draw a conclusion in chapter 6.

Chapter 2

Column Generation and Branch-and-Price

In the following chapter we give a short introduction to column generation and branch-and-price, where we follow the presentations of Desrosiers and Lübbecke [4, 5]. First of all, we explain the column generation technique which is applied when solving linear programs with a huge but finite number of variables. Then, we demonstrate how mixed integer programs can be reformulated so that the linear programming (LP) relaxation of the reformulated problem is potentially stronger than the original one. The LP relaxation of the reformulation can be solved by applying column generation. In the last section, we present a branching rule which is used when the LP relaxation of some mixed integer program is solved with column generation. This completes the branch-and-price algorithm.

2.1 Column Generation

Let $n \in \mathbb{N}$ be some natural number and let $J \subset \mathbb{N}$ be some finite set of indices where the cardinality number $m := |J|$ is huge in comparison to n. Additionally, let $x^j \in \mathbb{Q}^n$ be some vector and let $c_j \in \mathbb{Q}$ be some rational number for all $j \in J$ together with some vector $b \in \mathbb{Q}^n$. Suppose we want to solve the following *master problem*:

$$
\begin{aligned}
(MP) \quad \min \quad & \sum_{j \in J} c_j \lambda_j & & \\
\text{s.t.} \quad & \sum_{j \in J} x^j \lambda_j & \geq \quad & b \\
& \lambda_j & \geq \quad & 0 \quad \forall j \in J.
\end{aligned}
$$

Because this problem has got a huge number of variables, it is not efficient or even practicable to solve it by applying a traditional linear programming algorithm like the simplex algorithm [6] or an interior point method [7]. This is why we solve the master problem with the help of column generation as follows. We initialize the problem with some subset $J' \subseteq J$ of variables and call it the *restricted master problem*. We first solve the restricted master problem to optimality with primal solution $\lambda^* \in \mathbb{Q}^{|J'|}$ and dual solution $\mu^* \in \mathbb{Q}^n$ by applying a linear programming algorithm. Then we generate new variables by solving the following *pricing problem* to optimality

$$\begin{aligned} (PP) \quad \bar{c}^* := \quad & \min \quad c(x) - (\mu^*)^T x \\ & \text{s.t.} \quad\quad x \in X, \end{aligned}$$

where $X := \{x^j \in \mathbb{Q}^n : j \in J\}$ is the set of column coefficient vectors of the master problem and the objective function $c(x^j) := c_j$ is defined as the corresponding objective coefficient used in the master problem for all $j \in J$. We call the objective value $\bar{c}(x) := c(x) - \mu^* x$ of some column $x \in X$ its current *reduced cost*.

Let x^{j^*} with $j^* \in J$ be the obtained optimal solution for the pricing problem. If the strict inequality $\bar{c}(x^*) < 0$ holds, we add the improving variable λ_{j^*} with coefficient column x^{j^*} and objective coefficient c_{j^*} to the restricted master problem and repeat the just described proceeding, starting with solving the modified restricted master problem. Otherwise the inequality $\bar{c}(x^*) \geq 0$ holds. This proves that there is no improving variable $x \in X$ with negative reduced cost anymore. Therefore, the current solution λ^* is not only optimal for the restricted master problem but also optimal for the master problem.

In most applications the set X of columns is a set of combinatorial objects and we have more information about each column than just its column coefficients. This information is used when solving the pricing problem. The pricing problems are often solved with a suitable combinatorial algorithm or with branch-and-cut, which is introduced at the beginning of chapter 3. Hence, we do not have to explicitly calculate the reduced cost of all columns in each step.

2.2 Dantzig-Wolfe Reformulation

Let n be some natural number and let $x \in \mathbb{Q}^n$ have the form $x = (x^1, x^2, \ldots, x^K)$ where $x^k \in \mathbb{Q}^{n_k}$ for all $k \in \mathcal{K} := \{1, \ldots, K\}$. Similarly, let $c = (c^1, c^2, \ldots, c^K) \in \mathbb{Q}^n$ be

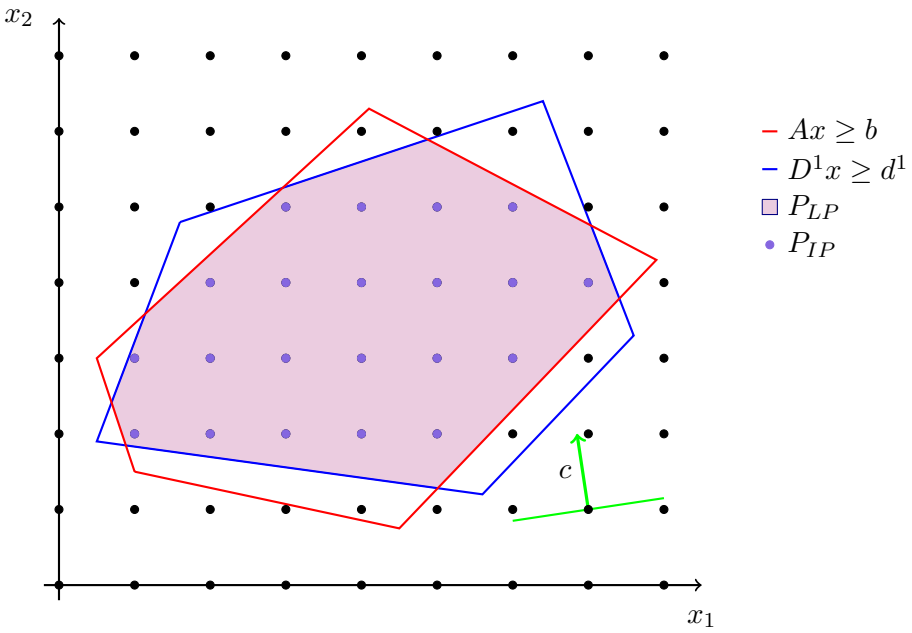


Figure 2.1: The polyhedra defined by the constraints of a 2-dimensional binary original problem with $K = 1$ and the set of feasible solutions together with the objective function vector c.

some rational vector. Furthermore, let m be some natural number. Consider matrices $A \in \mathbb{Q}^{m \times n}$ and $D^k \in \mathbb{Q}^{m_k \times n_k}$ for all $k \in \mathcal{K}$ along with vectors $b \in \mathbb{Q}^m$ and $d^k \in \mathbb{Q}^{m_k}$ for all $k \in \mathcal{K}$. Suppose the following problem is given

$$\begin{array}{lllll} (OP) & \min & c^T x & & \\ & \text{s.t.} & D^k x^k & \geq & d^k \quad \forall k \in \mathcal{K} \\ & & Ax & \geq & b \\ & & x_i^k & \in & \mathbb{Z} \quad \forall i \in \mathbb{I}_k \; \forall k \in \mathcal{K}, \end{array}$$

where $\mathbb{I}_k \subseteq \{1, \ldots, n_k\}$ for all $k \in \mathcal{K}$. We will refer to this problem as the *original problem* and call its LP relaxation the *original LP relaxation*. In figure 2.1 an example for the set of feasible solutions to an original problem and a corresponding original LP relaxation is depicted.

To ease the notation we define the set

$$X_k := \{x^k \in \mathbb{Q}^{n_k} : D^k x^k \geq d_k, \; x_i^k \in \mathbb{Z} \; \forall i \in \mathbb{I}_k\}$$

for all $k \in \mathcal{K}$, the set P_{IP} as the set of all feasible solutions to the original problem, and the polyhedron P_{LP} as the set of all feasible solutions to the LP relaxation of the original problem.

We introduce two ways of reformulating the original problem. First, we present the classical *convexification* approach which is based on the representation theorems of Minkowski and Weyl [8]. The convex hull $conv(Y)$ of some set Y is defined as follows:

$$conv(Y) := \left\{ \sum_{y \in Y} \alpha_y y : \alpha_y \geq 0 \ \forall y \in Y, \sum_{y \in Y} \alpha_y = 1 \right\}. \tag{2.1}$$

Each vector $x^k \in X^k$ with $k \in \mathcal{K}$ can be described as a convex combination of all extreme points $\{\mathbf{x}_p^k : p \in \mathcal{P}(k)\} \subseteq X^k$ together with a non-negative linear combination of all extreme rays $\{\mathbf{x}_r^k : r \in \mathcal{P}(k)\}$ of the convex hull $conv(X^k)$:

$$x^k = \sum_{p \in \mathcal{P}(k)} \mathbf{x}_p^k \lambda_p^k + \sum_{r \in \mathcal{R}(k)} \mathbf{x}_r^k \delta_r^k, \tag{2.2}$$

$$\sum_{p \in \mathcal{P}(k)} \lambda_p^k = 1, \tag{2.3}$$

$$\lambda_p^k \geq 0 \quad \forall p \in \mathcal{P}(k), \tag{2.4}$$

$$\delta_r^k \geq 0 \quad \forall r \in \mathcal{R}(k). \tag{2.5}$$

Note that the set of extreme points as well as the set of extreme rays of $conv(X^k)$ are finite sets for all $k \in \mathcal{K}$.
Due to the above reformulation, a vector $x \in P_{IP}$ can be written as

$$x = \sum_{k \in \mathcal{K}} \left(\sum_{p \in \mathcal{P}(k)} \mathbf{x}_p^k \lambda_p^k + \sum_{r \in \mathcal{R}(k)} \mathbf{x}_r^k \delta_r^k \right), \tag{2.6}$$

such that the following constraints are satisfied:

$$\sum_{p \in \mathcal{P}(k)} \lambda_p = 1 \quad \forall k \in \mathcal{K},$$

$$\lambda_p^k \geq 0 \quad \forall p \in \mathcal{P}(k) \ \forall k \in \mathcal{K},$$

$$\delta_r^k \geq 0 \quad \forall r \in \mathcal{R}(k) \ \forall k \in \mathcal{K}.$$

Using identity (2.2) in combination with the corresponding constraints (2.3), (2.4), and (2.5) we can reformulate the original problem and describe it by the following *extended*

formulation:

$$
\begin{aligned}
(IMP_1) \quad \min \quad & \sum_{k\in\mathcal{K}} \left(\sum_{p\in\mathcal{P}(k)} c_{pk}\lambda_p^k + \sum_{r\in\mathcal{R}(k)} c_{rk}\delta_r^k \right) \\
\text{s.t.} \quad & \sum_{k\in\mathcal{K}} \left(\sum_{p\in\mathcal{P}(k)} \mathbf{x}_p^k\lambda_p^k + \sum_{r\in\mathcal{R}(k)} \mathbf{x}_r^k\delta_r^k \right) \geq b \\
& \sum_{p\in\mathcal{P}(k)} \lambda_p^k = 1 \qquad \forall k \in \mathcal{K} \\
& \lambda_p^k \geq 0 \qquad \forall p \in \mathcal{P}(k)\ \forall k \in \mathcal{K} \\
& \delta_r^k \geq 0 \qquad \forall r \in \mathcal{R}(k)\ \forall k \in \mathcal{K} \\
& \sum_{p\in\mathcal{P}(k)} \mathbf{x}_p^k\lambda_p^k + \sum_{r\in\mathcal{R}(k)} \mathbf{x}_r^k\delta_r^k = x^k \qquad \forall k \in \mathcal{K} \\
& x^k \in X^k \qquad \forall k \in \mathcal{K},
\end{aligned}
$$

where the objective function is defined as $c_{jk} := (c^k)^T x_j^k$ for all $j \in \mathcal{P}(k) \cup \mathcal{R}(k)$ and $k \in \mathcal{K}$.
Note that the LP relaxation of problem (IMP_1) is a master problem like problem (MP) that we have seen in the column generation context. Due to the potentially huge number of columns of problem (IMP_1), we solve the LP relaxation of the extended formulation by applying column generation. We also call problem (IMP_1) the *integer master problem* and its LP relaxation the *linear master problem*.

Another way of reformulating the problem is called *discretization* approach. In contrast to the convexification approach we dot not reformulate the convex hull $conv(X^k)$, but we actually reformulate the set X^k for all $k \in \mathcal{K}$. For the remainder of this section we define $\mathbb{I}_k := \{1, \ldots, n_k\}$ for all $k \in \mathcal{K}$. Thus, the set X^k contains only integer vectors for all $k \in \mathcal{K}$.
It was shown [6] that every integer vector $x^k \in X^k$ can be described as an integral combination

$$
\begin{aligned}
x^k &= \sum_{p\in\mathcal{P}(k)} \mathbf{x}_p^k\lambda_p^k + \sum_{r\in\mathcal{R}(k)} \mathbf{x}_r^k\delta_r^k, \\
\sum_{p\in\mathcal{P}(k)} \lambda_p &= 1, \\
\lambda_p^k &\in \{0,1\} \quad \forall p \in \mathcal{P}(k), \\
\delta_r^k &\in \mathbb{Z}_{\geq 0} \quad \forall r \in \mathcal{R}(k),
\end{aligned}
$$

where $\{\mathbf{x}_p^k : p \in \mathcal{P}\} \subseteq X^k$ is some finite set of integer vectors and $\{\mathbf{x}_r^k : r \in \mathcal{R}\}$ is some set of integer rays of X^k. Notice that we vary the use of the sets $\mathcal{P}$ and $\mathcal{R}$ in the different contexts of convexification and discretization. We can now analogously reformulate the original problem in the following way:

$$
\begin{aligned}
(IMP_2) \quad \min \quad & \sum_{k\in\mathcal{K}} \left(\sum_{p\in\mathcal{P}(k)} c_{pk}\lambda_p^k + \sum_{r\in\mathcal{R}(k)} c_{rk}\delta_r^k \right) & & & \\
\text{s.t.} \quad & \sum_{k\in\mathcal{K}} \left(\sum_{p\in\mathcal{P}(k)} \mathbf{x}_p^k\lambda_p^k + \sum_{r\in\mathcal{R}(k)} \mathbf{x}_r^k\delta_r^k \right) & \geq & b & \\
& \sum_{p\in\mathcal{P}(k)} \lambda_p^k & = & 1 & \forall k \in \mathcal{K} \\
& \lambda_p^k & \in & \{0,1\} & \forall p \in \mathcal{P}(k)\ \forall k \in \mathcal{K} \\
& \delta_r^k & \in & \mathbb{Z}_{\geq 0} & \forall r \in \mathcal{R}(k)\ \forall k \in \mathcal{K}.
\end{aligned}
$$

Note that the LP relaxation of problem (IMP_2) equals the LP relaxation of the extended formulation that was introduced in the context of the convexification approach. Therefore, we use the same notations for this problem and its LP relaxation as in the context of the convexification approach.

When solving the linear master problem with column generation, we need the constraints containing just λ variables excluding the integrality constraints, i.e. we need the convexity constraint and the reformulated constraints of the original problem. Let $\mu_0^* \in \mathbb{Q}$ be an optimal dual solution corresponding to the convexity constraint and let $\mu^* \in \mathbb{Q}^m$ be an optimal dual solution corresponding to the other constraints. Furthermore, let the matrix A have the form $A = (A^1, A^2, \ldots, A^{\mathcal{K}})$, where $A^k \in \mathbb{Q}^{m\times n_k}$. The kth pricing problem of the extended formulation is an integer program for each $k \in \mathcal{K}$:

$$
\begin{aligned}
(PP_k) \quad z_k^* := \quad & \min \quad (c^k)^T x^k - (\mu^*)^T A^k x^k - \mu_0^* \\
& \text{s.t.} \quad x^k \in X^k.
\end{aligned}
$$

The minimum $\min_{k\in\mathcal{K}} z_k^*$ represents the optimal reduced cost. In many applications the pricing problems can be solved with a combinatorial algorithm. If no combinatorial algorithm is at hand, the pricing problems are solved by applying branch-and-cut. We give a short introduction to branch-and-cut at the beginning of chapter 3.

Vanderbeck and Savelsbergh [1] presented a generalization of the discretization approach to sets containing not only integer but also continuous variables. We can reformulate

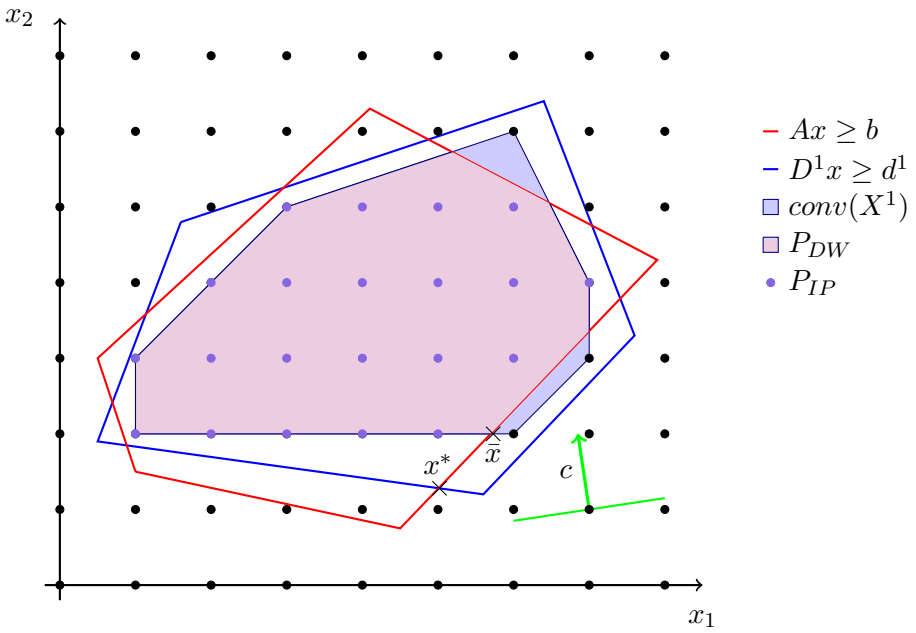


Figure 2.2: The polyhedra defined by the constraints of a 2-dimensional binary original problem with $K = 1$ and the convex hull of all integer points X^1 together with the objective function vector c, an optimal solution x^* for the original LP relaxation, and an optimal solution $\bar{x}$ for the linear master problem transferred to the original problem's solution space.

these mixed integer sets by applying discretization to all integer variables and convexification to all continuous variables.

At the end we want to analyze the relation of the original LP relaxation to the linear master problem. The following identity

$$z_{DW} = \min\{c^T x : x = (x^1, x^2, \ldots, x^K) \in \mathbb{Q}^n, Ax \geq b, x^k \in conv(X^k)\ \forall k \in \mathcal{K}\},$$

where z_{DW} is the optimal solution value to the linear master problem, holds due to Geoffrion [9]. We define the set of feasible solutions to the linear master problem transferred to the original problem's solution space as

$$P_{DW} := \{x = (x^1, x^2, \ldots, x^K) \in \mathbb{Q}^n : Ax \geq b, x^k \in conv(X^k)\ \forall k \in \mathcal{K}\}.$$

It is obvious that the relation $P_{DW} \subseteq P_{LP}$ holds. Hence, the linear master problem

leads to a potentially better lower bound than the original LP relaxation:

$$z_{DW} = \min\{c^T x : x \in P_{DW}\} \geq \min\{c^T x : x \in P_{LP}\} =: z_{LP}.$$

If $P_{DW} \subset P_{LP}$, we call the relaxation obtained by applying the Dantzig-Wolfe reformulation *stronger* than the original LP relaxation. In figure 2.2 we can see an example which illustrates the above described relations.

Besides the fact that the reformulated problem consists of variables which contain more information than the original variables, the potentially better lower bound is the main reason for using one of the previously described reformulation techniques.

2.3 Branch-and-Price

In the last sections we have seen how a given mixed integer program can be reformulated and how the LP relaxation of the obtained extended formulation is solved via column generation. In this section we present branching rules and name some futher features which help us to solve the mixed integer program to optimality. The branching rules are used similarly to the branching rules in the traditional branch-and-bound algorithm plus some modifications. See [8] for an introduction to branch-and-bound.

Consider the extended formulation which is obtained by reformulating a given mixed integer program with the use of the convexification approach and let $(\bar{\lambda}, \bar{\delta})$ be an optimal solution for the linear master problem but not feasible to the integer master problem. It is easy to see that branching candidates are original variables x_i^k with

$$\bar{x}_i^k = \sum_{p \in \mathcal{P}(k)} \mathbf{x}_{pi}^k \bar{\lambda}_p^k + \sum_{r \in \mathcal{R}(k)} \mathbf{x}_{ri}^k \bar{\delta}_r^k \notin \mathbb{Z}$$

and $i \in \mathbb{I}_k$, where $\mathbf{x}_{ji}^k$ is the ith component of $\mathbf{x}_j^k$ for $j \in \mathcal{P}(k) \cup \mathcal{R}(k)$ and $k \in \mathcal{K}$. We create two new problems by adding the constraint $x_j^k \geq \lceil \bar{x}_j^k \rceil$ and $x_j^k \leq \lfloor \bar{x}_j^k \rfloor$, respectively, for some branching candidate x_j^k, where $\lceil \cdot \rceil$ is the ceiling function and $\lfloor \cdot \rfloor$ is the floor function.
We can either add the constraints to the master problem or we can add the constraints to the pricing problem. The first option leads to a new dual variable for each constraint. The optimal solution values of these dual variables enter the objective function of the corresponding pricing problem. The other option results in new bounds for the

branching variable x_j^k in the pricing problem. We refer to Desrosiers and Lübbecke [4] for further analysis of this branching rule and its extension to the discretization approach.

This branching strategy is not limited to the initial linear master problem, which is the root node's LP relaxation of the underlying branch-and-bound tree. Since the branching decisions lead to new master constraints or new pricing constraints, we have to keep track of all previous branching decisions. Then we are able to apply the branching rules analogously.

All in all, this results in an algorithm which solves arbitrary mixed integer programs to optimality and takes advantage of the problem's structure. Solving problems by the use of column generation and branching is called *branch-and-price*. As we have seen, an original problem can be reformulated with one of the described techniques and branch-and-price can be applied to the obtained extended formulation.

Additional features can accelerate the solution process. A summary of these features can be found in [4] and [5]. We only want to mention *aggregation* here. It is a method to aggregate identical pricing problems to one representative pricing problem but it can only be applied when using the discretization apporach. Aggregation helps to remove symmetry from the problem and can be used in many applications. See [4] for further explanations.

Chapter 3

Cutting Planes

In this chapter we give a short introduction to branch-and-cut and discuss some techniques to generate valid inequalities for general integer programs. In the end we will explain how the concept of cutting planes can be transferred to the context of branch-and-price.

Consider some matrix $A \in \mathbb{Q}^{m \times n}$ along with vectors $b \in \mathbb{Q}^m$ and $c \in \mathbb{Q}^n$. Suppose the following integer program is given:

$$\begin{array}{lllll} (IP) & \min & c^T x & & \\ & \text{s.t.} & Ax & \geq & b \\ & & x_i & \in & \mathbb{Z}_{\geq 0} \quad \forall i \in \{1, \ldots, n\}. \end{array}$$

To ease the notation we define the set $X := \{x \in \mathbb{Z}^n_{\geq 0} : Ax \geq b\}$ of feasible solutions to problem (IP).
We call $\pi^T x \geq \pi_0$ with $(\pi, \pi_0) \in \mathbb{Q}^{n+1}$ a *valid inequality* for $conv(X)$ if $\pi^T x \geq \pi_0$ holds for all $x \in conv(X)$. Note that all rows of $Ax \geq b$ are valid inequalities for $conv(X)$.

Let x^* be an optimal solution for the LP relaxation of problem (IP). We want to find a valid inequality $\pi^T x \geq \pi_0$ with $(\pi, \pi_0) \in \mathbb{Q}^{n+1}$ such that $\pi^T x^* < \pi_0$ holds. These types of inequalities are called *cutting planes* and their generation is called *separation*. If we find a cutting plane which cuts off the solution x^*, we can add it to the problem formulation. Afterwards, we reoptimize the LP relaxation of problem (IP) and try to generate another cutting plane. If we do not find another cutting plane, we branch on a fractional variable and continue to solve the integer program with branch-and-bound. This strategy is called *branch-and-cut* and it is the standard technique to solve (mixed)

integer programs.

There are several ways of separating cutting planes. On the one hand, there are strategies which are only applicable to specific problems. On the other hand, there are separation techniques which can be applied in general. In the following section we discuss some strategies to obtain cutting planes for integer programs.

3.1 Separating Arbitrary LP-Feasible Solutions

To begin with, we analyse the complexity of separating an arbitrary feasible solution x^* for the LP relaxation of problem (IP). We call such a solution *LP-feasible*. We present the polynomial equivalence of separation and optimization [8] and refer to Garey and Johnson [10] for an introduction to computational complexity theory.

We define a *class of polyhedra* $\mathcal{P}$ as a set $\mathcal{P} = \{P_t : t \in \mathcal{T}\}$ where $\mathcal{T}$ is some set of objects and P_t is a bounded rational polyhedron for all $t \in \mathcal{T}$. The class of polyhedra $\mathcal{P}$ is called *proper* if there exist natural numbers n_t and s_t for each object $t \in \mathcal{T}$ which can be computed in polynomial time with respect to the size of t such that $P_t \in \mathcal{R}^{n_t}$ and the polyhedron P_t can be described by linear inequalities of size at most s_t.
The polynomial equivalence of separation and optimization [8] states that for a class of implicitly defined polyhedron $\mathcal{P}$ the following holds: Optimizing some linear objective function over a polyhedron $P \in \mathcal{P}$ or proving that P is empty can be done in polynomial time if and only if separating an arbitrary vector v exactly from the polyhedron P can be done in polynomial time. *Exactly separating* means finding a cutting plane which cuts off the vector v and is valid for P or proving that $v \in P$.
Note that a class of polyhedra $\mathcal{P}$ where each polyhedron $P \in \mathcal{P}$ describes the convex hull of all integer points of an explicitly defined, bounded rational polyhedron is proper. Hence, if problem (IP) is $\mathcal{NP}$-hard, separating an arbitrary LP-feasible solution x^* exactly is $\mathcal{NP}$-hard as well. Notice that in many applications problem (IP) is indeed $\mathcal{NP}$-hard. Thus, separating x^* exactly may take much time, but there are some techniques to separate an arbitrary feasible solution x^* heuristically in polynomial time. This means, if we do not find a cutting plane which cuts off the solution x^*, this does not prove that there does not exist any cutting plane which does. We only want to present the approach of optimizing over the first Chvátal closure and refer to Achterberg [11] for a summary of other methods which separate an arbitrary LP-feasible solution x^*.

Chvátal as well as Gomory introduced separately from each other a strategy to generate valid inequalities and we show how these inequalities can be used to separate a solution x^*. We follow the presentations of Schrijver [8].
Let $u \in \mathbb{Q}^m$ be some rational vector and reconsider problem (IP), where $a_j \in \mathbb{Q}^m$ with $j \in \{1, 2, \ldots, n\}$ is the jth column of matrix A. The inequality

$$\sum_{j=1}^{n} \lceil u^T a_j \rceil x_j \geq \lceil u^T b \rceil \tag{3.1}$$

is valid for the convex hull $conv(X)$ and is called a *Chvátal-Gomory inequality*. We define the set P^1 of feasible solutions to the LP relaxation of problem (IP) which satisfy all Chvátal-gomory inequalities

$$P^1 := \{x \in P : \sum_{j=1}^{n} \lceil u^T a_j \rceil x_j \geq \lceil u^T b \rceil \ \forall u \in \mathbb{Q}^m\}, \tag{3.2}$$

where P is the polyhedron of feasible solutions to the LP relaxation of problem (IP). It was shown [12] that P^1 is a polyhedron as well, which we call *the first Chvátal closure* of P. Therefore, we can iteratively add all Chvátal-gomory inequalities for P^i and obtain the polyhedron P^{i+1} for all $i \in \mathbb{N}_0$ where $P^0 := P$. Since we only added valid inequalities for the convex hull $conv(X)$, the relation $P^i \supseteq conv(X)$ holds for all $i \in \mathbb{N}_0$. Chvátal in fact proved [12] that $P^k = conv(X)$ for some $k \in \mathbb{N}$. This leads to a sequence of polyhedra $(P^i)_{i \in \mathbb{N}_0}$ with

$$P = P^0 \supseteq P^1 \supseteq P^2 \ldots \supseteq P^k = conv(X). \tag{3.3}$$

The *Chvátal rank* of a Chvátal-gomory inequality is defined as the smallest natural number $i \in \mathbb{N}_0$ such that the inequality is valid for P^i. Hence, P^i consists of all Chvátal-gomory inequalities with Chvátal rank smaller or equal to i.

Reconsider the problem of separating an arbitrary feasible solution x^* to the LP relaxation of problem (IP). Instead of separating x^* exactly from the convex hull $conv(X)$, we separate x^* exactly from the polyhedron P^1. Unfortunately, Eisenbrand [13] proved that this problem is in general $\mathcal{NP}$-hard, too. We still want to present a strategy to separate x^* exactly from the polyhedron P^1.
The following mixed integer program, which was presented by Fischetti and Lodi [14],

represents the desired separation problem

$$
\begin{array}{lllllll}
(CG) & \min & \sum_{j=1}^{n} \alpha_j x_j^* - \alpha_0 & & & & \\
& \text{s.t.} & & f_j & = & \alpha_j - u^T A_j & \forall j \in \{1, 2, \dots n\} \\
& & & f_0 & = & \alpha_0 - u^T b & \\
& & 0 \leq & f_j & \leq & 1 - \delta & \forall j \in \{0, 1, \dots, n\} \\
& & 0 \leq & u_j & \leq & 1 - \delta & \forall j \in \{1, 2, \dots, n\} \\
& & & \alpha_j & \in & \mathbb{Z} & \forall j \in \{0, 1, \dots, n\},
\end{array}
$$

where A_j denotes the jth column of A and $\delta \in \mathbb{Q}_{>0}$ is some small tolerance parameter. Note that the variables α_j and α_0 represent the values $\lceil u^T A_j \rceil$ for $j \in \{1, 2, \dots, n\}$ and $\lceil u^T b \rceil$, respectively.
Fischetti and Lodi also added some modifications to accelerate the solution process and analyzed the performance of the obtained separation algorithm. They concluded that optimizing over the first Chvátal closure often gives a very good approximation of the optimal solution value, though it may require large computing time.

There are a lot of other cutting plane methods which cut off a LP-feasible solution heuristically. Some of them are applicable to general mixed integer programs such as clique cuts [11] or $\{0, \frac{1}{2}\}$-cuts [15]. But there are also cuts which are only applicable to specific types of constraints like knapsack cover cuts or flow cover cuts [11].

3.2 Separating Basic LP-Feasible Solutions

In most applications linear programs which appear in branch-and-cut are solved with an algorithm which calculates basic feasible optimal solutions. Note that this can be done by applying an interior point method plus some crossover [16, 17] or by applying the simplex algorithm. Thus, an obtained optimal solution x^* to the LP relaxation is basic feasible and a basis of x^* is known. There are several separation strategies which exploit the basis information to cut off the solution x^*. In this section we introduce some fundamental concepts of linear programming and present the Gomory cutting plane method to show exemplarily how basis information can be used to generate cuts for integer programs. Note that the Gomory cutting plane method can be extended to mixed integer programs. The extended cuts are called Gomory mixed integer cuts. We refer to Achterberg [11] for a summary of cutting planes for mixed integer programs

which are generated with the help of basis information.
Consider the following integer program

$$\begin{array}{lllll} (IP) & \min & c^T x & & \\ & \text{s.t.} & Ax & = & b \\ & & x_i & \in & \mathbb{Z}_{\geq 0} \quad \forall i \in \{1, \ldots, n\}, \end{array}$$

where $A \in \mathbb{Q}^{m \times n}$ is a matrix of rank $rk(A) = m$ and $c \in \mathbb{Q}^n$ as well as $b \in \mathbb{Q}^m$ are rational vectors. Furthermore, let A consist of columns a_j with $j \in \{1, 2, \ldots, m\}$.
Let $B \subseteq (1, 2, \ldots, m)$ with $|B| = n$ and $N = (1, 2, \ldots, m) \backslash B$ be some ordered subsets of the column indices. We call B *basis* of A and the matrix $A_B := (a_j : j \in B)$ with columns a_j for all $j \in B$ *basis matrix* of A if A_B is nonsingular and therefore invertible. The set N is called *non-basis* of A and the matrix $A_N := (a_j : j \in N)$ is called *non-basis matrix* of A. The feasible solution x^* to the LP relaxation of problem (IP) with $x^*_N := 0$ and $x^*_B := A_B^{-1} b$ is called a corresponding *basic solution*, where $x_N := (x_j : j \in N)$ and $x_B := (x_j : j \in B)$ are the component vectors induced by N and B, respectively. If $x^*_B \geq 0$ holds, the solution x^* is a *basic feasible solution* of A and we also call x^* *basic LP-feasible*.
It was shown [18] that a solution x^* is basic feasible to some linear program if and only if it is a vertex of the polyhedron of feasible solutions. Recall that a point $x \in P$ is a vertex of the polyhedron P if it cannot be written as a strict convex combination of other points of the polyhedron P. This conclusion helps us to transfer the concept of basic feasible solutions to problems which are not written in the form of problem (IP). If we consider linear programs with inequality constraints $A'x \geq b'$ and free variables, a vertex x^* of the polyhedron of feasible solutions satisfies exactly $\bar{n} := rk(A')$ linearly independent inequalities with equality. We will refer to this result in the following sections.

Let B be a basis of A and let x^* be the corresponding basic feasible solution. We can reformulate the constraints $Ax = b$ as follows

$$x_B + A_B^{-1} A_N x_N = A_B^{-1} b.$$

Thus, we obtain the equations

$$x_k + \sum_{j \in N} \bar{a}_{kj} x_j = \bar{b}_k \ \forall k \in B,$$

where $\bar{a}$ denotes the entry in the kth row and the jth column of the matrix $A_B^{-1}A_N$ and $\bar{b} := A^{-1}b$.
Suppose $x^* \notin \mathbb{Z}^n$. Since $x_j^* = 0$ for all $j \in N$, there exists an index $\bar{k} \in B$ with $\bar{b}_{\bar{k}} \notin \mathbb{Z}$. The Chvátal-gomory inequality

$$x_{\bar{k}} + \sum_{j \in N} \lceil \bar{a}_{\bar{k}j} \rceil x_j \geq \lceil \bar{b}_{\bar{k}} \rceil \tag{3.4}$$

cuts off the solution x^* because the following strict inequality holds

$$x_{\bar{k}}^* + \sum_{j \in N} \lceil \bar{a}_{\bar{k}j} \rceil x_j^* = \bar{b}_{\bar{k}} < \lceil \bar{b}_{\bar{k}} \rceil.$$

These types of inequalities can be separated efficiently when applying the simplex algorithm to solve the LP relaxation of problem (IP).

3.3 Cutting Planes in Branch-and-Price

In the following section we explain how the concept of cutting planes can be transferred to the context of branch-and-price. Our presentation is based on [19].

There are two types of cutting planes in the context of branch-and-price with a reformulated original problem. First, we consider cutting planes which are defined on the variables of the extended formulation. Suppose we reformulated problem (OP), which was defined in section 2.2, by applying the discretization approach and we obtained the extended formulation (IMP_2). We only consider the case $K = 1$ and $\mathbb{I}_1 = \{1, 2, \ldots, n\}$. For the remainder of this section let $\mathcal{P} := \mathcal{P}(1)$ and $\mathcal{R} := \mathcal{R}(1)$ as well as $\lambda_p := \lambda_p^1$ for all $p \in \mathcal{P}$ and $\delta_r := \delta_r^1$ for all $r \in \mathcal{R}$. Additionally, let $X := X^1 = \{x \in \mathbb{Z}^n : D^1 x \geq d^1\}$. Thus, we consider an original problem of the form

$$\begin{array}{lllll} (OP') & \min & c^T x & & \\ \text{s.t.} & & Ax & \geq & b \\ & & x & \in & X, \end{array}$$

and the extended formulation

$$
\begin{array}{lll}
(IMP_2') \quad \min & \sum_{p\in\mathcal{P}} c_p\lambda_p + \sum_{r\in\mathcal{R}} c_r\delta_r & \\
\text{s.t.} & \sum_{p\in\mathcal{P}} x_p\lambda_p + \sum_{r\in\mathcal{R}} x_r\delta_r \geq b & \\
& \sum_{p\in\mathcal{P}} \lambda_p = 1 & \\
& \lambda_p \in \{0,1\} & \forall p \in \mathcal{P} \\
& \delta_r \in \mathbb{Z}_{\geq 0} & \forall r \in \mathcal{R}.
\end{array}
$$

Of course, we can use the integrality constraints of the λ and δ variables to generate cutting planes defined on these variables, but it is not clear how these inequalities affect the pricing problems. Note that the pricing problems are formulated with original variables and we cannot just translate an inequality formulated with master variables. The structure of the pricing problems is not changed in specific cases. In general though, these inequalities change the structure of the pricing problems and lead to more difficult pricing problems.

A way to handle these inequalities in the pricing problems was introduced by Desaulniers et al. [19]. We have to calculate the potential new column's coefficient of a cutting plane explicitly in the pricing problem. If the generated cut is a Chvátal-Gomory inequality, the calculation of the coefficient is determined by the constraints of the master problem and its multipliers used to generate the Chvátal-Gomory inequality. Note that we also have to model the ceiling function for this calculation. This can be done by adding a new constraint and two new variables, one continuous and one integer, to the formulation of the pricing problem as we have already seen in problem (CG) in section 3.1. This does not seem to make the solution of the pricing problem more difficult, but if we generate several cuts, it might influence the solution time.
The situation completely changes if the generated cut is a clique cut which forbids to use more than one column of a set of conflicting columns $\mathcal{C}$:

$$\sum_{p\in\mathcal{C}} \lambda_p \leq 1.$$

In this case we have to check if the potential new column $\mathbf{x}_0$ is in conflict with all columns in the set $\mathcal{C}$. This can be done by adding a boolean variable y_p which is true if and only if the column $\mathbf{x}_0$ is in conflict with the column $\mathbf{x}_p$ for all $p \in \mathcal{C}$. Since the size of the set $\mathcal{C}$ can be as big as the number of all potential columns, it can have a huge influence on

the solution time of the pricing problem. Spoorendonk and Desaulniers [20] presented a technique to generate the coefficients of a clique cut heuristically.

As we have seen, the change of the pricing problem and therefore the use of the cutting planes depends on the cutting plane type. Furthermore, it is not clear how the change of the pricing problem influences the solution process if we use a combinatorial algorithm. So far, there are only a few cases where cutting planes on the master variables were integrated successfully into a combinatorial algorithm [19].

Now, we consider cutting planes which are formulated in the original problem's solution space in case we used the convexification approach. We restrict our attention to an integer original problem with just one set of additional constraints besides $Ax \geq b$ which was used to reformulated the problem. We use the same notations as in the beginning of this section, which leads to the extended formulation

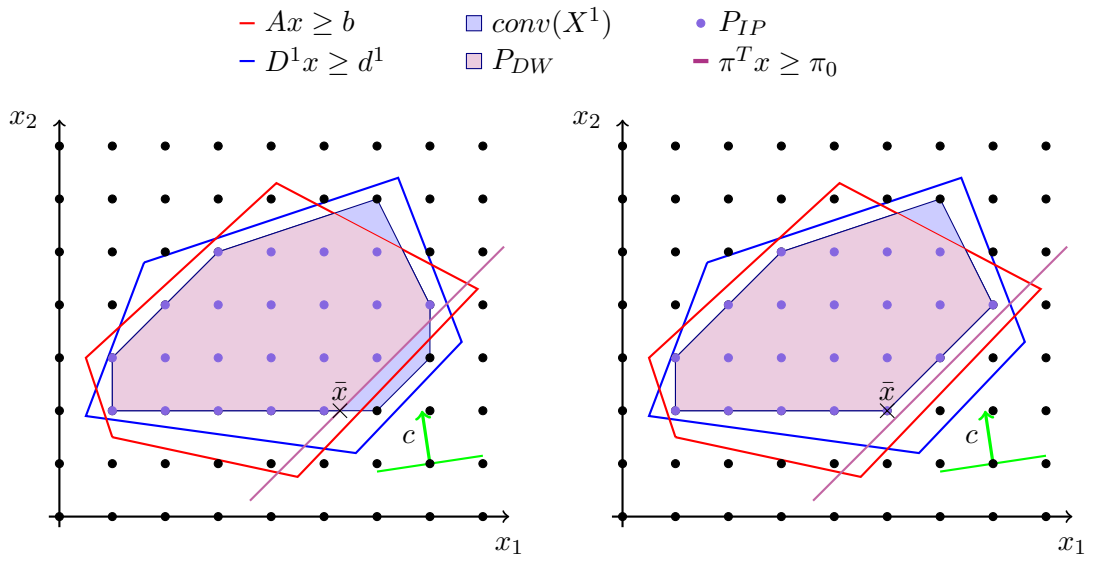
$$
\begin{aligned}
(IMP_1') \quad \min \quad & \sum_{p\in\mathcal{P}} c_p\lambda_p + \sum_{r\in\mathcal{R}} c_r\delta_r & & & \\
\text{s.t.} \quad & \sum_{p\in\mathcal{P}} x_p\lambda_p + \sum_{r\in\mathcal{R}} x_r\delta_r & \geq & b & \\
& \sum_{p\in\mathcal{P}} \lambda_p & = & 1 & \\
& \lambda_p & \geq & 0 & \forall p \in \mathcal{P} \\
& \delta_r & \geq & 0 & \forall r \in \mathcal{R} \\
& \sum_{p\in\mathcal{P}} x_p\lambda_p^k + \sum_{r\in\mathcal{R}} x_r\delta_r & = & x & \\
& x & \in & X. &
\end{aligned}
$$

A cutting plane of the form $\pi^T x \geq \pi_0$ with $(\pi, \pi_0) \in \mathbb{Q}^{n+1}$ can be handled like a branching decision, which we have seen in section 2.3. Hence, we have to add the cutting plane either to the master problem or the pricing problem. Adding the cutting plane to the master problem leads to new dual variables $\nu_0 \in \mathbb{Q}$. Thus, we add the term

$$-\nu_0^* \pi^T x^k,$$

where $\nu_0^* \in \mathbb{Q}$ is an optimal dual solution corresponding to the cutting plane $\pi^T x \geq \pi_0$, to the objective funtion of the pricing problem. Adding the cutting plane to the pricing problem does not lead to any further changes.

In figure 3.1 we can see the polyhedra of feasible solutions to the linear master problem



(a) Adding $\pi^T x \geq \pi_0$ to the master problem.

(b) Adding $\pi^T x \geq \pi_0$ to the pricing problem.

Figure 3.1: The polyhedra defined by a 2-dimensional binary original problem with $K = 1$ and an valid inequality $\pi^T x \geq \pi_0$ for P_{IP} which is not valid for $\bar{x}$.

after adding a cutting plane to either the master problem or the pricing problem. As we can see, adding the cutting plane to the pricing problem leads to a potentially stronger linear master problem in comparison to adding the cutting plane to the master problem. Note that in case we add the cutting plane to the master problem, the solution space of the pricing problem does not change at all. This might be useful when using a combinatorial algorithm to solve the pricing problem.

In the end, we can handle both types of cutting planes, but cutting planes on master variables may lead to more difficult pricing problems. For the remainder of this thesis we will only concentrate on cutting planes formulated on the original variables. In the next chapter we present some techniques to separate these types of cutting planes.

Chapter 4

Separation of Cutting Planes in the Original Problem

Suppose we reformulated the original problem (OP) by the use of the convexification approach and obtained the extended formulation (IMP_1). Both problems were introduced in section 2.2. Remember that we called the extended formulation the integer master problem and its LP relaxation the linear master problem. Similarly to section 2.2, we name the set of feasible solutions to the original problem P_{IP}, the polyhedron of feasible solutions to the original LP relaxation P_{LP}, and the polyhedron of feasible solutions to the linear master problem P_{LMP}.
Let $(\bar{\lambda}, \bar{\delta})$ be an optimal solution for the linear master problem and let

$$\bar{x} := \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}(k)} \mathbf{x}_p^k \bar{\lambda}_p^k + \sum_{r \in \mathcal{R}(k)} \mathbf{x}_r^k \bar{\delta}_r^k$$

be the corresponding feasible solution for the original LP relaxation. Our goal is to find a cutting plane of the form $\pi^T x \geq \pi_0$ which is valid for all $x \in P_{IP}$ but violated by $\bar{x}$. Thus the following conditions must hold:

$$\begin{aligned} &\pi^T x \geq \pi_0 \quad \forall x \in P_{IP}, \\ &\pi^T \bar{x} < \pi_0. \end{aligned}$$

An example for this cutting plane is shown in 4.1.
There are several ways of finding cutting planes, but in a lot of applications we would make use of a known basis for $\bar{x}$ as seen in section 3.2. Unfortunately, we usually do not have a basis of $\bar{x}$ at hand although we translated the basic feasible solution $(\bar{\lambda}, \bar{\delta})$

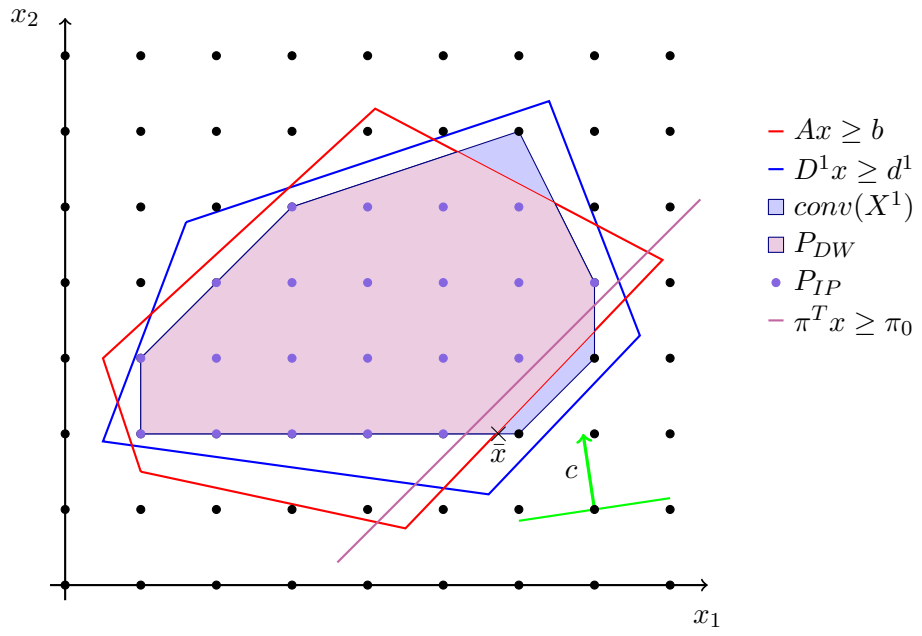


Figure 4.1: The polyhedra defined by a 2-dimensional binary original problem with $K = 1$ and an valid inequality $\pi^T x \geq \pi_0$ for P_{IP} which is not valid for $\bar{x}$.

into the original solution space. The reason for this is that in general the vertices of the polyhedron P_{LMP} do not correspond to the vertices of the polyhedron P_{LP} as already seen in section 2.2.

In the following section we discuss some ideas of how to find a basic feasible solution x^* and use it to separate the solution $\bar{x}$. Then we present the corresponding separation algorithm which makes use of these basic feasible solutions. In the last section of this chapter we present valid inequalities which help us to strengthen the original LP-relaxation. These inequalities might support the success of the separation procedure.

4.1 Search for Basic LP-Feasible Solutions

In order to exploit separation strategies in the original solution space which make use of a basis and cut off the current master solution $\bar{x}$, we have to find a basic LP-feasible solution for the original LP relaxation. We want to find a basic feasible solution x^* of the original LP relaxation which is close to the solution $\bar{x}$ in the following way: When we separate the solution x^* by using a basis of x^*, at least one generated cutting plane does

not only cut off the solution x^* but also the solution $\bar{x}$. We cannot guarantee to find such a basic feasible solution, but in the following we want to present some strategies to obtain basic feasible solutions to the original LP relaxation, which might be useful.

4.1.1 Solving the Original LP Relaxation

An obvious strategy to find a vertex of the polyhedron P_{LP} would be solving the original LP relaxation to optimality using the simplex algorithm and obtaining an optimal basic feasible solution x^*. Of course, the idea behind this approach is that $c^T x^*$ as well as $c^T \bar{x}$ are both lower bounds on the optimal solution value of the original problem. In section 2.2 we have seen that the inequality $c^T x^* \leq c^T \bar{x}$ is valid. And since the identity

$$c^T \bar{x} = \min\{c^T x : x = (x^1, x^2, \ldots, x^K) \in \mathbb{Q}^n, Ax \geq b, x^k \in conv(X^k)\ \forall k \in \mathcal{K}\} \quad (4.1)$$

holds, we can conclude that the condition $conv(X^k) = \{x^k \in \mathbb{Q}^{n_k} : D^k x^k \geq d_k\}$ for all $k \in \mathcal{K}$ implies $c^T \bar{x} = c^T x^*$. If this is the case, we expect the approximation x^* to be useful. Otherwise, it is doubtful that the solution x^* is close to the solution $\bar{x}$ since we just use the original objective function, which does not contain any information about the specific solution $\bar{x}$.

4.1.2 Search on the Same Faces of the Polyhedron

The strategy to obtain a basic feasible solution, which is presented in this section, was presented by Range [3] in the same context as here and by Dash and Goycoolea [21] in the context of separating rank-1 Gomory mixed-integer cuts heuristically.

We already mentioned that the solution $\bar{x}$ does not need to be a vertex of the polyhedron P_{LP}, but it still might be lying on some of its faces. A *face* F of a polyhedron P is the non-empty intersection $F := P \cap \{x \in \mathbb{Q}^n : \alpha^T x = \alpha_0\}$ of P and the set of solutions which satisfy a given valid inequality $\alpha^T x \geq \alpha_0$ for P. We say that the solution x lies on a face F if the condition $x \in F$ holds. We refer to Cook [12] for an introduction to faces of polyhedra.

Let $\mathcal{F} := \{F : F \text{ face of } P_{LP},\ \bar{x} \in F\}$ be the set of faces of the polyhedron P_{LP} that the solution $\bar{x}$ lies on and let $\mathcal{X}_{\mathcal{F}} := \bigcap_{F \in \mathcal{F}} \{x \in F : x \text{ basic feasible to } P_{LP}\}$ be the set of basic LP-feasible solutions which lie on all these faces $F \in \mathcal{F}$. The set $\mathcal{F}$ of an exemplary problem is depicted in figure 4.2.

We can write $\bar{x}$ as a convex combination $\bar{\alpha} \in [0,1]^{|\mathcal{X}_\mathcal{F}|}$ of these basic LP-feasible solutions

$$\bar{x} = \sum_{x \in \mathcal{X}_\mathcal{F}} \bar{\alpha}_x x$$

with $\sum_{x \in \mathcal{X}_\mathcal{F}} \bar{\alpha}_x = 1$. Since the set $\mathcal{X}_\mathcal{F}$ containts only basic LP-feasible solutions, we can easily generate cutting planes which cut off the solution x by the use of a corresponding basis for $x \in \mathcal{X}_\mathcal{F}$.

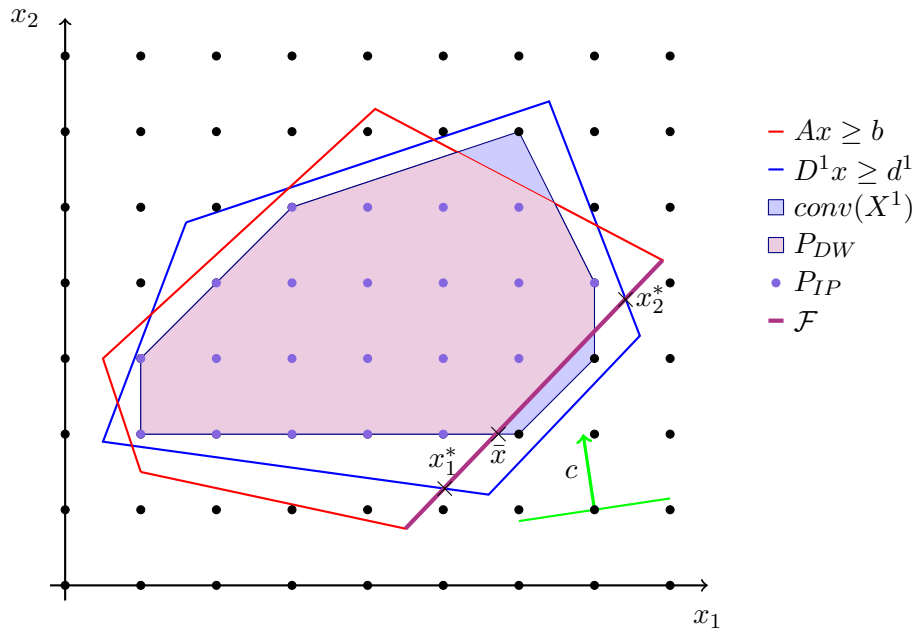


Figure 4.2: The polyhedra defined by the constraints of a 2-dimensional binary original problem with $K = 1$ and the corresponding optimal solutions $x_1^* \in \mathcal{X}_\mathcal{F}$ and $x_2^* \in \mathcal{X}_\mathcal{F}$ to problem (4.2) together with the set $\mathcal{F}$.

The following theorem was presented by Ralphs and Galati [22] and it demonstrates the potential use of these cutting planes.

Theorem 4.1. *Let $\hat{x} \in \mathbb{Q}^n$ be some rational vector and let $\mathcal{X} \subset \mathbb{Q}^n$ be some subset of the underlying vector space. Furthermore, let $\hat{\alpha} \in \mathbb{Q}_{\geq 0}^{|\mathcal{X}|}$ be the coefficient vector of some convex combination of $\mathcal{X}$ with $\sum_{x \in \mathcal{X}} \hat{\alpha}_x x = \hat{x}$ and $\sum_{x \in \mathcal{X}} \hat{\alpha}_x = 1$.*
If $\hat{x}$ violates an inequality $\pi^T x \geq \pi_0$ with $(\pi, \pi_0) \in \mathbb{Q}^{n+1}$, then there must exist some $x' \in \mathcal{X}$ with $\hat{\alpha}_{x'} > 0$ such that x' also violates the inequality $\pi^T x \geq \pi_0$.

Proof. Assume that $\pi^T x \geq \pi_0$ holds for all $x \in \mathcal{X}$ with $\hat{\alpha}_x > 0$. Using this assumption together with the identities $\sum_{x\in\mathcal{X}} \hat{\alpha}_x x = \hat{x}$ and $\sum_{x\in\mathcal{X}} \hat{\alpha}_x = 1$ we can conclude

$$\pi^T \hat{x} = \pi^T \left(\sum_{x\in\mathcal{X}} \hat{\alpha}_x x \right) = \sum_{x\in\mathcal{X}} \hat{\alpha}_x \pi^T x \geq \sum_{x\in\mathcal{X}} \hat{\alpha}_x \pi_0 = \pi_0 \sum_{x\in\mathcal{X}} \hat{\alpha}_x = \pi_0,$$

which is a contradiction to $\pi^T \hat{x} < \pi_0$. □

Hence, in order to cut off the solution $\bar{x}$ it is necessary to cut off a solution $x \in \mathcal{X}_\mathcal{F}$. We want to use this condition although it is not sufficient. This is why we search for some $x^* \in \mathcal{X}_\mathcal{F}$. Note that an optimal solution for the original LP relaxation is in general not in $\mathcal{X}_\mathcal{F}$. Therefore, we use the strategy proposed by Range [3] as well as by Dash and Goycoolea [21], which we will present in the following.

We want to find a solution $x^* \in \mathcal{X}_\mathcal{F}$, e.g. a basic feasible solution x^* to the orginal LP relaxation which lies on the same faces as the solution $\bar{x}$. If the solution $\bar{x}$ is not basic LP-feasible, it is obvious that the basic LP-feasible solution x^* has to lie on some additional faces of the polyhedron P_{LP}, the solution $\bar{x}$ does not lie on.
Since the constraints of the original LP relaxation are inequalities, a constraint represents a potential face of the polyhedron of all feasible solutions to the original LP relaxation. Thus, a solution x^* lies on a potential face if and only if the corresponding constraint is satisfied with equality by x^*.
Let $\mathbb{J}_0 := \{j \in \{1,\ldots,m\} : A_j \bar{x} = b_j\}$ and $\mathbb{J}_k := \{j \in \{1,\ldots,m_k\} : D_j^k \bar{x} = d_j^k\}$ for all $k \in \mathcal{K}$ be the index sets of all inequalities which are satisfied with equality by $\bar{x}$. We formulate the following optimization problem

$$\begin{aligned} z^* = \min \quad & \sum_{k\in\mathcal{K}_0} \sum_{j\in\mathbb{J}_k} s_j^k & & & \\ \text{s.t.} \quad & D^k x^k - s^k &=& d^k & \forall k \in \mathcal{K} \\ & Ax - s^0 &=& b & \\ & x_i^k &\in& \mathbb{Q} & \forall i \in \mathbb{I}_k \; \forall k \in \mathcal{K} \\ & s_j^k &\geq& 0 & \forall j \in \{1,\ldots,m\} \; \forall k \in \mathcal{K}_0, \end{aligned} \tag{4.2}$$

where $\mathcal{K}_0 := \mathcal{K} \cup \{0\}$. We call the objective function used in problem 4.2 the *face objective function*. Notice that we only added slack variables to the LP relaxation of the original problem and changed the objective function. Thus, the polyhedron of all feasible solutions of problem (4.2) equals the polyhedron P_{LP} disregarding the slack variables. The following proposition shows that an optimal solution for problem (4.2) exists.

Proposition 4.2 ([3]). *A feasible solution* $(x^*, s_*^0, \ldots, s_*^K)$ *with* $(s_*^k)_j = 0$ *for all* $j \in \{1, \ldots, m\},\ k \in \mathcal{K}$ *and objective value* $z^* = 0$ *to problem* (4.2) *exists and is optimal.*

Proof. First of all, it is obvious that the optimal objective value is non-negative because the slack variables are non-negative. Furthermore, the solution $(\bar{x}, \bar{s}^0, \ldots, \bar{s}^K)$ with $\bar{s}^0 := A\bar{x} - b$ and $\bar{s}^k := D^k\bar{x}^k - d^k$ for all $k \in \mathcal{K}$ is feasible to problem (4.2) and the identity $\bar{s}_j^k = 0$ holds for all $j \in \mathbb{J}_k, k \in \mathcal{K}$. Thus, we obtain the optimal solution value $z^* = 0$ due to the solution $(x^*, s_*^0, \ldots, s_*^K) := (\bar{x}, \bar{s}^0, \ldots, \bar{s}^K)$. □

As we mentioned, the solution $\bar{x}$ does not need to be a vertex of P_{LP}. This is transferred to the solution $(\bar{x}, \bar{s}^0, \ldots, \bar{s}^K)$ of problem (4.2), which was defined in the previous proof, because its formulation differs only in the use of slack variables. Therefore, we may obtain another solution $(x^*, s_*^0, \ldots, s_*^K) \neq (\bar{x}, \bar{s}^0, \ldots, \bar{s}^K)$ when solving problem (4.2) with the simplex algorithm. The vector x^* is apparently a vertex of the polyhedron P_{LP} and so we are able to separate it by making use of a corresponding basis.

4.1.3 Combination of Both Strategies

We have seen two strategies of how to find a basic feasible solution for the original LP relaxation by solving a linear program. Since both of them differ only in the objective function and the use of slack variables, we are able to combine both approaches. This can be done by using a convex combination of the used objective functions.
For the remainder of this section let $c \neq 0$ and $\mathbb{J}_k \neq \emptyset$ for some $k \in \mathcal{K}_0$. In problem (4.2) we replace the objective function $S(x, s^0, \ldots, s^K) := \sum_{k=0}^{K} \sum_{j \in \mathbb{J}_k} s_j^k$ by the convex combination

$$f_\alpha(x, s^0, \ldots, s^K) := \alpha S(x, s^0, \ldots, s^k) + (1 - \alpha) \frac{\sqrt{\sum_{k \in \mathcal{K}_0} |\mathbb{J}_k|}}{||c||} c^T x, \tag{4.3}$$

where $\alpha \in [0, 1]$ and $|| \cdot ||$ is the Euclidean norm of a vector. Note that $\sqrt{\sum_{k \in \mathcal{K}_0} |\mathbb{J}_k|}$ is the Euclidian norm of the objective function vector $S(x, s^0, \ldots, s^K)$.
We can either fix α or calculate some value for α before solving problem (4.2) with objective function f_α. In the following we want to present a strategy to calculate a suitable value for α.

We define

$$\bar{D}_j^k := (\underbrace{0, \ldots, 0}_{\sum_{k'=1}^{k-1} n_{k'}}, D_j^k, \underbrace{0, \ldots, 0}_{\sum_{k'=k+1}^{n} n_{k'}}) \tag{4.4}$$

as the n-dimensional row corresponding to the jth row of the matrix D_j^k for all $j \in m_k$ and $k \in \mathcal{K}$. Let

$$\mathcal{J} := \{(A_j, b_j) : j \in \mathbb{J}_0\} \cup \bigcup_{k \in \mathcal{K}} \{(\bar{D}_j^k, d_j^K) : j \in \mathbb{J}_k\} \tag{4.5}$$

be the set of all inequalities of the original formulation which are satisfied with equality by $\bar{x}$. We are now interested in the dimension d of the vector space

$$span(\mathcal{J}) := \left\{ \sum_{v \in \mathcal{J}} \mu_v v : \mu \in \mathbb{Q}^{|\mathcal{J}|} \right\} \tag{4.6}$$

which is spanned by $\mathcal{J}$. The value d equals the maximum number of linearly independent vectors in the set $\mathcal{J}$. Since a basic feasible solution satisfies $n' := rk(\mathcal{A})$ linearly independent inequalities with equality as seen in section 3.2, where

$$\mathcal{A} := \begin{pmatrix} A^1 & A^2 & \dots & A^K \\ D^1 & 0 & \dots & 0 \\ 0 & D^2 & \dots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \dots & 0 & D^K \end{pmatrix} \tag{4.7}$$

and $A = (A^1, A^2, \dots, A^K)$ with $A^k \in \mathbb{Q}^{m \times n_k}$ for all $k \in \mathcal{K}$, the quotient $\alpha_{\bar{x}} := \frac{d}{n'} \in [0, 1]$ shows how close the solution $\bar{x}$ is to being basic feasible. The larger the quotient $\alpha_{\bar{x}}$, the closer is $\bar{x}$ to being basic feasible to the original LP relaxation. This is why $\alpha_{\bar{x}}$ might be a suitable value for the convex combination coefficient α. If $\alpha_{\bar{x}}$ is close to one, we will base our search on the basis information we already have due to $\bar{x}$, which means weighting the objective function S higher than the original objective function. Otherwise, if $\alpha_{\bar{x}}$ is close to zero, we will depend more on the original objective function than on the poor basis information of $\bar{x}$. We call the obtained objective function $f_{\bar{\alpha}}$ *generic convex objective function.*

The dimension d of the vector space $span(\mathcal{J})$ can be obtained by using Gaussian elimination in polynomial time.

All in all, we are able to combine both previous approaches, which results in a more reasonable algorithm by the use of a generically calculated convex combination coefficient.

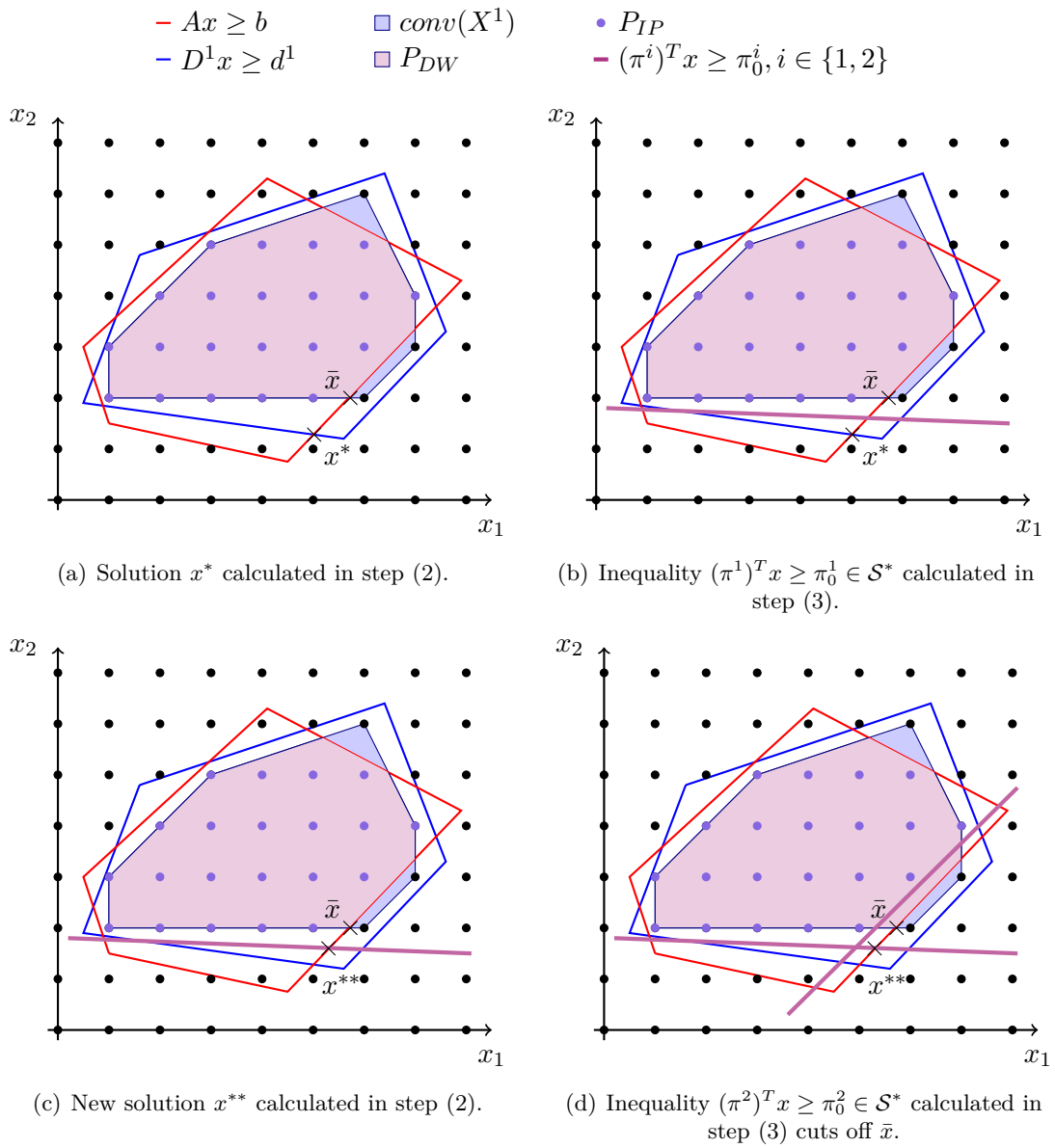
4.2 The Basis Separator

So far, we described approaches to calculate basic LP-feasible solutions. Generating cutting planes which cut off one of these solutions might help us to cut off the solution $\bar{x}$. In this section we describe the obtained basic separation procedure and present an extension which allows us to apply the basic separation procedure iteratively. The obtained extended algorithm was introduced by Range [3].

Let $c_B : P_{LP} \to \mathbb{Q}^n$ be a function which maps a feasible solution x to the original LP relaxation to an objective function vector $c_B(x)$. Note that c_B might be one of the objective functions used in the previously described strategies to calculate a basic LP-feasible solution.
Separating a basic LP-feasible solution obtained by minimizing the objective $c_B(\bar{x})$ over the polyhedron P_{LP} and checking if one of the generated cutting planes also cuts off the solution $\bar{x}$ results in a separation procedure.
Since this procedure is likely to fail, especially when the polyhedron P_{DW} is much stronger than the polyhedron P_{LP}, we present an extension which deals with the failure to cut off $\bar{x}$.

Let $\mathcal{S}^* \subseteq \mathbb{Q}^{n+1}$ be the set of valid inequalities for $conv(P_{IP})$ of the form $\pi^T x \geq \pi_0$ which were generated by separating some basic feasible solution x^* to the original LP relaxation. Suppose $\pi^T \bar{x} \geq \pi_0$ holds for all $(\pi, \pi_0) \in \mathcal{S}^*$. In this case we are not able to cut off the solution $\bar{x}$, but we can make use of the generated inequalities.
Since all inequalities $\pi^T x \geq \pi_0$ with $(\pi, \pi_0) \in \mathcal{S}^*$ are not valid for our approximation x^*, we add them to the original LP relaxation and get a new approximation x^{**} by applying the basic separation procedure again. Adding these inequalities strengthens the original LP relaxation, but we do not cut off any feasible solutions to the original problem. Repeatedly separating these approximations of $\bar{x}$ until a predefined number of iterations is achieved or no more cuts are found results in separation algorithm 4.1, which is exemplary applied in figure 4.3. We also call algorithm 4.1 *basis separator*.

This separation algorithm including an extension, which we will present in section 4.3.1, was presented by Range [3]. He uses the objective function S, which was proposed in section 4.1.2, as the objective function vector $c_B(\bar{x})$.



(a) Solution x^* calculated in step (2).

(b) Inequality $(\pi^1)^T x \geq \pi_0^1 \in \mathcal{S}^*$ calculated in step (3).

(c) New solution x^{**} calculated in step (2).

(d) Inequality $(\pi^2)^T x \geq \pi_0^2 \in \mathcal{S}^*$ calculated in step (3) cuts off $\bar{x}$.

Figure 4.3: The steps of the basis separator 4.1 illustrated by applying the algorithm to a 2-dimensional example.

Algorithm 4.1 Basis separator

Input: Optimal solution $\bar{x}$ to the linear master problem (transferred to original solution space), the original LP relaxation P_{LP}, function $f : P_{LP} \to \mathbb{Q}^n$ and maximal number of iterations $i_{\max}$.
Output: Cutting plane $\pi^T x \geq \pi_0$ which is valid for all $x \in P_{IP}$, but not for $\bar{x}$, or nothing.

(1) Initialize $\mathcal{S} := \emptyset$ and $i := 0$.

(2) Solve $\min\{f(\bar{x})^T x : x \in P_{LP}, \pi^T x \geq \pi_0 \; \forall(\pi, \pi_0) \in \mathcal{S}\}$ using the simplex algorithm and let x^* be an optimal basic feasible solution.

(3) Separate x^* and let $\mathcal{S}^*$ be the set of generated inequalities.

(4) Return all inequalities $\pi^T x \geq \pi_0$ with $(\pi, \pi_0) \in \mathcal{S}^*$ which satisfy the strict inequality $\pi^T \bar{x} < \pi_0$. If there does not exist such an inequalitiy, set $i := i + 1$.

(5) If $i \geq i_{\max}$ or $S^* = \emptyset$ holds, return nothing, otherwise set $\mathcal{S} := \mathcal{S} \cup \mathcal{S}^*$ and go to step (2).

4.3 Strengthening the Original LP Relaxation

The success of the previous strategies highly depends on the LP relaxations of the original and the master problem. If the LP relaxation of the master problem is much stronger than the original LP relaxation, it will be hard to approximate the solution $\bar{x}$ by some basic feasible solution x^* of the original LP relaxation no matter what objective function we use. This is why we want to strengthen the LP relaxation of the original problem by adding valid inequalties for the polyhedron $conv(P_{IP})$ which is the convex hull of all feasible solutions to the original problem.
When solving the linear master problem we optimize over the polyhedron

$$P_{DW} := \{x = (x_1, \ldots, x^K) \in \mathbb{Q}^n : Ax \geq b, \; x^k \in conv(X^k) \; \forall k \in \mathcal{K}\}. \tag{4.8}$$

Notice that the polyhedron P_{DW} can be obtained by adding all valid inequalities for the convex hull $conv(X^k)$ for all $k \in \mathcal{K}$ to the formulation of the polyhedron P_{LP} if the polyhedron P_{DW} is a strict subset of the polyhedron P_{LP}. Thus, we are especially interested in finding some of these inequalities to get a better approximation of P_{DW}.
Suppose we only add all inequalities which are valid for $conv(X^k)$ for all $k \in \mathcal{K}$ and are satisfied with equality by $\bar{x}$ to the formulation of P_{LP} and obtain the polyhedron P'_{LP}.

Note that the vector $\bar{x}$ is a vertex of P'_{LP} and therefore basic feasible to the corresponding LP relaxation, where the set of feasible solutions is given by the polyhedron P'_{LP}.

Let $\mathcal{C}^k \subseteq \mathbb{Q}^{n_k+1}$ for $k \in \mathcal{K}$ be some set of valid inequalities for $conv(X^k)$ with the form $(\pi^k)^T x^k \geq \pi_0^k$. Consider the following linear program, which employs these sets of valid inequalities:

$$\begin{array}{lrcll} \min & c^T x & & & \\ \text{s.t.} & D^k x^k & \geq & d^k & \forall k \in \mathcal{K} \\ & (\pi^k)^T x^k & \geq & \pi_0^k & \forall (\pi^k, \pi_0^k) \in \mathcal{C}^k \ \forall k \in \mathcal{K} \\ & Ax & \geq & b & \\ & x_i^k & \in & \mathbb{Q} & \forall i \in \mathbb{I}_k \ \forall k \in \mathcal{K} \end{array} \tag{4.9}$$

We will will refer to problem (4.9) as the *stronger original LP relaxation* and define the set of its feasible solutions as P_S. Note that we can replace the original LP relaxation by the stronger original LP relaxation in every context, in which we have been using it. Notice that the statement

$$P_{DW} \subseteq P_S \subseteq P_{LP} \tag{4.10}$$

holds because we just added valid inequalities for the convex hull $conv(X^k)$ for some $k \in \mathcal{K}$ which are also valid for the polyhedron P_{DW}. Since the relation (4.10) holds, this results in a potentially better approximation of the polyhedron P_{DW}, and therefore, we might get a better approximation x^* of $\bar{x}$ when applying the basis separator 4.1.

Now, consider an inequality $\pi^T x \geq \pi_0$ which is valid for the convex hull $conv(P_{IP})$. In general this inequality is not valid for the polyhedron P_{DW} because P_{DW} is the set of all feasible solutions to the linear master problem transferred to the original solution space. Remember that the linear master problem is the LP relaxation of the integer master problem which is some formulation with solution set P_{IP}. Thus, we do not only add the inequality $\pi^T x \geq \pi_0$ to the (stronger) original LP relaxation, but we also add it to the linear master problem. Thus, relation (4.10) maintains valid.

In the following sections we present strategies to calculate valid inequalities, which help us to strengthen the original LP relaxation. We differentiate between valid inequalities for $conv(P_{IP})$ and valid inequalities for $conv(X^k)$ for some $k \in \mathcal{K}$ not only because of the previously described difference but also because there are different strategies to obtain valid inequalities for each inequality type.

4.3.1 Valid Inequalities Obtained During the Solution Process

In this section we will present some types of valid inequalities for $conv(P_{IP})$ and $conv(X^k)$ for $k \in \mathcal{K}$ which are directly obtained during the solution process using branch-and-price. Since we do not need the additional valid inequalities until we have solved the linear master problem to optimality, we can use all information we gain while applying column generation in the root node of our branch-and-price tree.

First of all, we use the original objective function. Let $z_{DW} := c^T \bar{x}$ be the lower bound to the optimal objective value of the original problem, which is due to the linear master problem. Obviously, the inequality $c^T x \geq z_{DW}$ is valid for all $x \in conv(P_{IP})$ and it is satisfied with equality by $\bar{x}$. Thus, it might be useful to add this inequality to the (stronger) original LP relaxation as well as to the linear master problem.
Furthermore, we might know that the objective value of the original problem is always integer because the objective function only consists of integer coefficients and the solution set P_{IP} only contains integer vectors. If we do know that this is the case, we can even add the inequality $c^T x \geq \lceil z_{DW} \rceil$, which we call the *original objective cut*. Range [3] introduced the original objective cut in this context.

When solving the kth pricing problem for some $k \in \mathcal{K}$, we optimize over the polyhedron $conv(X^k)$. If we solve the pricing problem with a branch-and-cut algorithm, valid inequalities for the convex hull $conv(X^k)$ might be separated in the root node. As we have just seen, these *pricing cuts* may help us to strengthen the original LP relaxation. We can add these inequalities to the set $\mathcal{C}^k$ for all $k \in \mathcal{K}$ and obtain a potentially stronger LP relaxation. Notice that we usually solve the pricing problems with branch-and-cut if we do generic column generation. This way we can add these inequalities without any further effort.

We can obtain another type of valid inequalities when the objective function $(c^k)^T x^k - (\mu^*)^T A^k x^k - \mu_0^*$ of the kth pricing problem for some $k \in \mathcal{K}$ is considered, which was used in the last pricing round of the root node. If we solve the linear master problem to optimality, the reduced costs of any potential new variable is non-negative. Therefore the inequality $(c^k)^T x^k - (\mu^*)^T A^k x^k \geq \mu_0^*$ is valid for the convex hull $conv(X^k)$ and we can add it to it to the (stronger) original LP relaxation. We call such an inequality a

reduced cost cut.
Suppose $\bar{\delta} = 0$. Then these inequalities are also satisfied with equality by $\bar{x}$ because $\bar{x}^k$ can by definition be written as a convex combination of columns of the linear master problem for all $k \in \mathcal{K}$

$$\bar{x}^k = \sum_{p \in \mathcal{P}(k)} \mathbf{x}_p^k \bar{\lambda}_p^k,$$

where $\sum_{p \in \mathcal{P}(k)} \bar{\lambda}_p^k = 1$. Since $(\bar{\lambda}, 0)$ is optimal to the linear master problem, every column corresponding to a variable λ_p^k with $\bar{\lambda}_p^k > 0$ has reduced costs equal to zero. Hence, the following holds:

$$\begin{aligned}
\left((c^k)^T - (\mu^*)^T A^k\right) \bar{x}^k &= \left((c^k)^T - (\mu^*)^T A^k\right) \left(\sum_{p \in \mathcal{P}(k)} \mathbf{x}_p^k \bar{\lambda}_p^k \right) \\
&= \sum_{\substack{p \in \mathcal{P}(k): \\ \bar{\lambda}_p^k > 0}} \bar{\lambda}_p^k \underbrace{\left((c^k)^T - (\mu^*)^T A^k\right) \mathbf{x}_p^k}_{=\mu_0^*} \\
&= \sum_{\substack{p \in \mathcal{P}(k): \\ \bar{\lambda}_p^k > 0}} \bar{\lambda}_p^k \mu_0^* \\
&= \mu_0^* \underbrace{\sum_{\substack{p \in \mathcal{P}(k): \\ \bar{\lambda}_p^k > 0}} \bar{\lambda}_p^k}_{=1} \\
&= \mu_0^*.
\end{aligned}$$

So we can see that the reduced cost cuts are satisfied with equality by the optimal solution $\bar{x}$ to the linear master problem transferred to the original solution space if the pricing problems are bounded.

4.3.2 The Chief-and-Slave Algorithm

In this section we present a strategy to generate valid inequalities for the polyhedron $conv(X^k)$ for all $k \in \mathcal{K}$ by the use of mixed integer programs. Our strategy is based on the algorithm proposed by Louveaux et al. [23]. They use the algorithm to analyze the strength of multi-row cuts in the context of branch-and-cut.

For the remainder of this section let $K = 1$ and $\mathbb{I}_1 = \{1, 2, \ldots, n\}$ as in section 3.3. We also make use of the previous notations. Due to these restrictions, the considered problem consists of just one integer pricing problem.

To simplify the notation we denote $P^X := \{x^k \in \mathbb{Q}^n : D^1 x \geq d^1\}$ as the set of all feasible solutions to the LP relaxation of the pricing problem.
As we have seen, we are especially interested in valid inequalities for $conv(X)$ which are satisfied with equality by the solution $\bar{x}$ because they might help us to find a basic feasible solution which is close to $\bar{x}$. These inequalities should be linearly independent to the inequalities we already use in the formulation of the pricing problem. Otherwise, we would not gain any information.
If a new inequality is valid for all solutions in X but not for all solutions in P^X, it is linearly independent to the already used inequalities. Otherwise, we could describe the new inequality by a linear combination of the already used inequalities and thus, it would be valid for all solutions in P^X. We will use this sufficient condition to avoid generating unnecessary inequalities. Notice that it is easy to test if some vectors are linearly independent by using a linear program, but it is not clear how to formulate the linear independence of a variable vector to given vectors as a constraint in an integer linear program. For the remainder of this section let X be finite. Obviously, the set Q of valid inequalities for X of the form $\pi^T x \geq \pi_0$ can be written as

$$Q = \left\{(\pi, \pi_0) \in \mathbb{Q}^{n+1} : \pi^T x \geq \pi_0 \ \forall x \in X\right\}.$$

Let x^* be some feasible solution for the LP relaxation of the pricing problem. We generate valid inequalities which are satisfied with equality by $\bar{x}$ and cut off the solution x^* by solving the following linear program

$$\begin{array}{lrcl} \min & \pi^T x^* - \pi_0 & & \\ \text{s.t.} & \pi^T \bar{x} & = & \pi_0 \\ & (\pi, \pi_0) & \in & Q. \end{array} \tag{4.11}$$

Besides the fact that Q can have exponential size in n, the problem is unbounded if there exists a cutting plane which cuts off the solution x^* and is satisfied with equality by $\bar{x}$. Otherwise, the vector $(0, 0) \in \mathbb{Q}^{n+1}$ is optimal to problem (4.11). This is why we introduce a normalization proposed by Balas and Perregaard [24]:

$$\pi^T(\bar{x} - x^*) = 1. \tag{4.12}$$

In the following we show that if we add this normalization constraint to problem (4.11), the problem gets bounded while no relevant solution is disregarded.

Proposition 4.3 ([23])**.** *Let $x^* \in P^X$ and $\bar{x} \in conv(X)$. The following optimization*

problem is allways bounded:

$$\begin{array}{rrcl} \min & \pi^T x^* - \pi_0 & & \\ s.t. & \pi^T \bar{x} & = & \pi_0 \\ & \pi^T(\bar{x} - x^*) & = & 1 \\ & (\pi, \pi_0) & \in & Q. \end{array} \tag{4.13}$$

Moreover, if an optimal solution exists, the optimal objective value is equal to -1.

Proof. Let $(\bar{\pi}, \bar{\pi}_0)$ be an optimal solution for problem (4.13), then the following holds:

$$\bar{\pi}^T x^* - \underbrace{\bar{\pi}_0}_{\bar{\pi}^T \bar{x}} = \bar{\pi}^T (x^* - \bar{x}) = -1.$$

□

We now proof that adding the normalization constraint does not lead to disregarding any relevant solutions.

Proposition 4.4 ([23])**.** *Let* $x^* \in P^X$ *and* $\bar{x} \in conv(X)$*. Furthermore, let* $(\pi, \pi_0) \in Q$ *be an inequality which cuts off the solution* x^* *and is satisfied with equality by* $\bar{x}$*. Then there exists some rational number* $\alpha > 0$ *such that* $(\alpha\pi, \alpha\pi_0) \in Q$ *and* $(\alpha\pi)^T(\bar{x} - x^*) = 1$.

Proof. We know that $\pi^T \bar{x} = \pi_0$ and $\pi^T x^* < \pi_0$. Hence, the strict inequalitiy $\pi^T \bar{x} > \pi^T x^*$ and therefore $\pi^T(\bar{x} - x^*) > 0$ holds. Thus, the value $\alpha := \dfrac{1}{\pi^T(\bar{x} - x^*)}$ fullfills the desired properties. □

Due to proposition 4.3 and 4.4, we have seen that solving problem (4.13) to optimality produces an inequality which cuts off a given solution x^* and is satisfied with equality by $\bar{x}$ if such an inequality exists.

We already mentioned that the set Q might have exponential size in n. This is why we want to solve problem (4.13) by applying cut generation.
Let $S \subseteq X$ and define the set

$$Q(S) := \left\{ (\pi, \pi_0) \in \mathbb{Q}^{n+1} : \pi^T x \geq \pi_0 \ \forall x \in S \right\}. \tag{4.14}$$

Consider the following *chief problem*:

$$\begin{array}{rlrcl} C(S) := & \min & \pi^T x^* - \pi_0 & & \\ & \text{s.t.} & \pi^T \bar{x} & = & \pi_0 \\ & & \pi^T(\bar{x} - x^*) & = & 1 \\ & & (\pi, \pi_0) & \in & Q(S). \end{array} \tag{4.15}$$

Note that the chief problem with $S = X$ is equal to problem (4.13). We start solving the chief problem with the set $S := \emptyset$. In order to check if an optimal solution $(\bar{\pi}, \bar{\pi}_0)$ of the chief problem for some $S \subseteq X$ is feasible and therefore optimal to problem (4.13), we solve the following *slave problem*

$$\begin{array}{rlrcl} S(\bar{\pi}, \bar{\pi}_0) := & \min & \bar{\pi}^T x & - & \bar{\pi}_0 \\ & \text{s.t.} & x & \in & X. \end{array} \tag{4.16}$$

Notice that the slave problem is an integer program since X is a set containing just integer vectors. If the optimal solution value $S(\bar{\pi}, \bar{\pi}_0)$ is non-negative, the solution $(\bar{\pi}, \bar{\pi}_0)$ is optimal to problem (4.13). Otherwise, let x' be an optimal solution for the slave problem. Then the strict inequality $\bar{\pi}^T x' < \bar{\pi}_0$ holds and thus, $(\bar{\pi}, \bar{\pi}_0)$ is not feasible to problem (4.13). Adding x' to the set S and iteratively applying this procedure leads to algorithm 4.2, which is exemplary performed in figure 4.4.

Algorithm 4.2 Chief-and-slave algorithm

Input: Finite sets $X \subset \mathbb{Q}^n$ and $S_0 \subseteq X$ together with vectors $\bar{x} \in conv(X)$ and $x^* \notin conv(X)$.

Output: A Valid inequality of the form $\bar{\pi}^T x \geq \bar{\pi}_0$ which is valid for all X and are fullfilled with equality by $\bar{x}$, or nothing.

(1) Initialize $S = S_0$.

(2) Solve the chief problem (4.15).

(3) If the chief problem is feasible, let $(\bar{\pi}, \bar{\pi}_0)$ be an optimal solution and go to step (4). Otherwise, return nothing.

(4) Solve the slave problem (4.16). Let x' be an optimal solution.
If $\bar{\pi}^T x' \geq \bar{\pi}_0$, return $\bar{\pi}^T x \geq \pi_0$. Otherwise, define $S := S \cup \{x'\}$ and go to step (2).

We apply algorithm 4.2 with the set of feasible solutions X to the pricing problem as input data. Furthermore, we use some subset $S_0 \subseteq X$, which we will specify in the

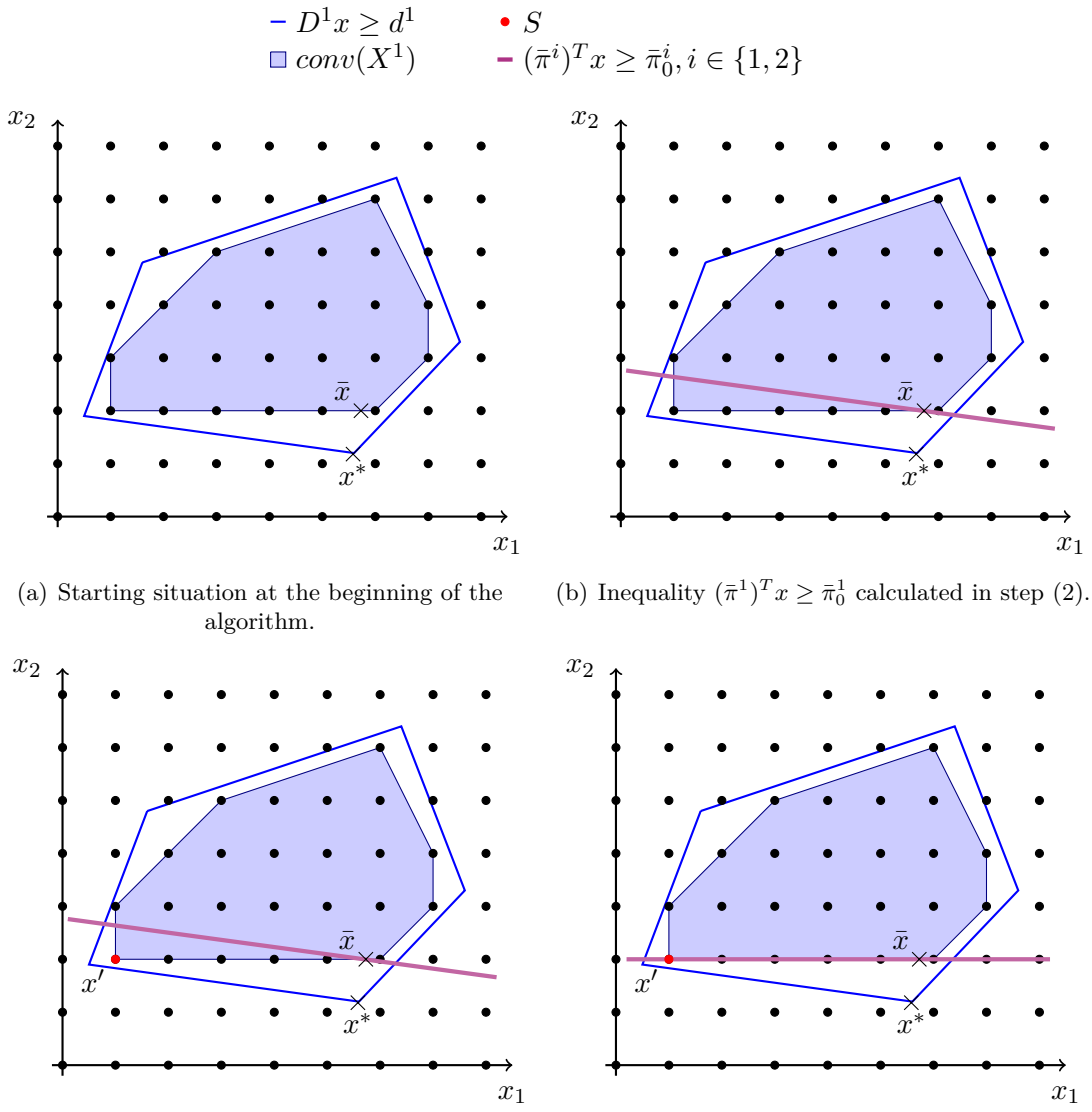
following, and the vector $\bar{x}$ as a current optimal solution to the linear master problem, transferred to the original solution space and an optimal solution x^* to the original LP relaxation. If the algorithm calculates a valid inequality, we add it to the formulation of the pricing problem and the original problem. After that, we calculate a new optimal solution x^* to the original LP relaxation and repeat this procedure.

We now want to deal with the choice of the set S_0. Obviously, we could initialize the algorithm with the set $S_0 = \emptyset$, but starting with a non-empty set S_0 potentially reduces the number of iterations. In section 4.3.1 we have seen that the solution $\bar{x}$ can be written as a convex combination of columns of the linear master problem

$$\bar{x} = \sum_{p \in \mathcal{P}} \mathbf{x}_p \bar{\lambda}_p,$$

where $\sum_{p \in \mathcal{P}} \bar{\lambda}_p = 1$. Note that the columns of the linear master problem do not represent any extreme rays of X because we only consider a finite set X. Analogously to proving that the reduced cost cuts are satisfied with equality by $\bar{x}$, we can show that a valid inequality for the convex hull $conv(X)$ which is satisfied with equality by $\bar{x}$ is also satisfied with equality by all $\mathbf{x}_p$ with $\lambda_p > 0$.
We initialize the set S_0 with $S_0 := \{\mathbf{x}_p : \lambda_p > 0\}$. This might result in less iterations than initializing the set S_0 with the empty set.

In this section we have seen an algorithm which produces valid inequalities heuristically that are satisfied with equality by the solution $\bar{x}$. Although we only considered the case of one integer pricing problem, the procedure can easily be transferred to the general case because the pricing problem was examined independently of the master problem.
Due to the restriction that the generated inequality has to be satisfied with equality by $\bar{x}$, we limited the set of possible valid inequalities, but it still might take some time to apply algorithm 4.2. This is because we have to solve an integer program in step (4) which is equal to a pricing problem with a different objective function and as already mentioned, the pricing problems are $\mathcal{NP}$-hard in most applications.



(a) Starting situation at the beginning of the algorithm.

(b) Inequality $(\bar{\pi}^1)^T x \geq \bar{\pi}_0^1$ calculated in step (2).

(c) Solution x' calculated in step (4) does not satisfy inequality $\bar{\pi}^T x \geq \bar{\pi}_0$.

(d) Inequality $(\bar{\pi}^2)^T x \geq \bar{\pi}_0^2$ calculated in step (2) is valid for $conv(X)$.

Figure 4.4: Applying the chief-and-slave algorithm.

Chapter 5

Experimental Results

We implemented the basis separator 4.1 including its variations and extensions in GCG 1.1.0.1 [2], which is based on SCIP 3.0.1.3 [11] with CPLEX 12.5.0.0 as LP-solver.

We applied the basis separator to instances of various problem classes. Namely these classes are the *bin packing problem (bp)*, the *cutting stock problem (cs)*, the *vertex coloring problem (coloring)*, the *capacitated p-median problem (cpmp)*, the *generalized assignment problem (gap)*, the *resource allocation problem (rap)* and the *multiple knapsack problem (mkp)*. Additionally, we tested the basis separator on instances of the MIPLIB 2003 [25] and the MIPLIB 2010 [26] which were already successfully tested with GCG [27]. We refer to these instances as *gcgjournal* instances. Formal problem definitions and the test sets of the problem classes are stated in appendix A.

All computations but the ones dealing with SCIP exclusively and GCG's computations on resource allocation problem or gcgjournal instances were performed by using the *RWTH Compute Cluster* [28]. The other calculations were performed on Intel Core i7-2600 CPUs with 16GB of RAM on openSUSE 12.1 workstations running Linux kernel 3.1.10. The default time limit is 3600 seconds unless stated otherwise.

When using the basis separator, we previously have to solve the root node's original LP relaxation in order to solve the linear program in step (2) of algorithm 4.1 by using a specific implementational feature of SCIP, the *diving mode*. With the help of the diving mode we can change and solve the current linear program easily. Afterwards, all changes are reversed.
For the purpose of a better comparison of GCG's performance with and without using

the basis separator, we solve the original LP relaxation even if the basis separator is disabled. Thus, the default setting of `GCG` which we use is `GCG`'s initial default setting plus solving the original LP relaxation.

We only tested the basis separator on instances where the root node of the reformulated problem was solved before the time limit was reached and the solution process is not finished after solving the root node when using the previously defined default setting. Depending on the solution time of `GCG`'s default setting, we classified all instances of a given problem class into groups with the exception that we divided the multiple knapsack instances of the same type into groups. We refer to appendix A for further details. The *easy* (`e`) instances were solved to optimality within 60 seconds using the default setting, the *middle* (`m`) instances were solved to optimality between 60 seconds and 3600 seconds, and the *hard* (`h`) instances were not solved to optimality within the time limit of 3600 seconds. Each test set contains at least five instances. If there exist less than five instances of a specific problem class and difficulty level, we grouped the instances of more than one difficulty level into a test set. The test sets are named as follows. Each test set's name consists of the problem class in the previously introduced short form plus the suffix `-nr`, which stands for the instances that were not solved in the root node, and at least one suffix `-e`, `-m`, and `-h`, which represents the contained difficulty levels.

In order to cut off an auxiliary basic LP-feasible solution in step (3) of algorithm 4.1, we use all of `SCIP`'s separators but the *integer objective value separator*, *closecuts meta separator*, *rapidlearning separator*, and the *Chvátal-Gomory cuts computed via a sub-MIP* as introduced in section 3.1. Namely we separate an auxiliary basic LP-feasible solution using the *clique separator*, the *complemented mixed integer rounding cuts separator*, the *flowcover separator*, the *Gomory MIR cuts separator*, the *implied bounds separator*, the *multi-commodity-flow network cut separator*, the *oddcycle separator*, the *strong Chvátal-Gomory cuts separator* and the $\{0, \frac{1}{2}\}$*-cuts separator*. In order to separate cutting planes using these separators, we set `SCIP`'s parameters to the values which are given by `SCIP`'s predefined parameter setting *aggressive*.

In the next section we compare the different objectives which can be used to calculate basic LP-feasible solutions in step (2) of the basis separator. Then, we analyze the basis separator in section 5.2. We examine which one of `SCIP`'s separators finds useful cutting planes and we compare the basis separator to the *master separator* [2], which is by default enabled when using `GCG`.

In section 5.3 we examine the impact of the additional inequalities which were introduced in section 4.3, before we compare the performance of `GCG` including the basis separator to `GCG`'s performance, using the default setting.

5.1 Comparison of the Different Search Strategies

First of all, we compare the performance of the basis separator 4.1 when using the different choices of the objective functions f to calculate the auxiliary basic LP-feasible solutions. We introduced the original objective function (`origobj`) in section 4.1.1, the face objective function (`face`) in section 4.1.2, and the generic convex objective function, which is a combination of both strategies with a generically calculated convex combination (`genconv` or `gen`), in section 4.1.3. In table 5.1 we can see the number of nodes (Nod), the solution time (T), the Dantzig-Wolfe gap that was closed (Gap cl), where the gap is given by the difference of the root dual bound of the Dantzig-Wolfe reformulated problem and the optimal solution value, and the number of found cuts due to the basis separator (Fnd) when using the different objectives. Furthermore, we state the time that was spent in the basis separator (BT), and the percentage of affected instances (Aff), where at least one cut was found. In each line the percentage of affected instances and the shifted geometrical mean of the other values for the different test sets is presented. On the one hand, we use the shifted geometrical mean because it allows us to average out nonnegative values in contrast to the geometrical mean, which demands positive values. On the other hand, it has some advantage compared to the arithmetic mean. We use the geometrical shift 100 for the number of nodes, 10 for the solution time as well as for the time spent in the separator, and 1 for the integrality gap that was closed as well as for the number of generated cuts.

Note that the size of the given Dantzig-Wolfe gap can only be reduced due to a better dual bound. This is an important measure to evaluate the use of a separation algorithm besides the solution time and the number of used branch-and-bound nodes.

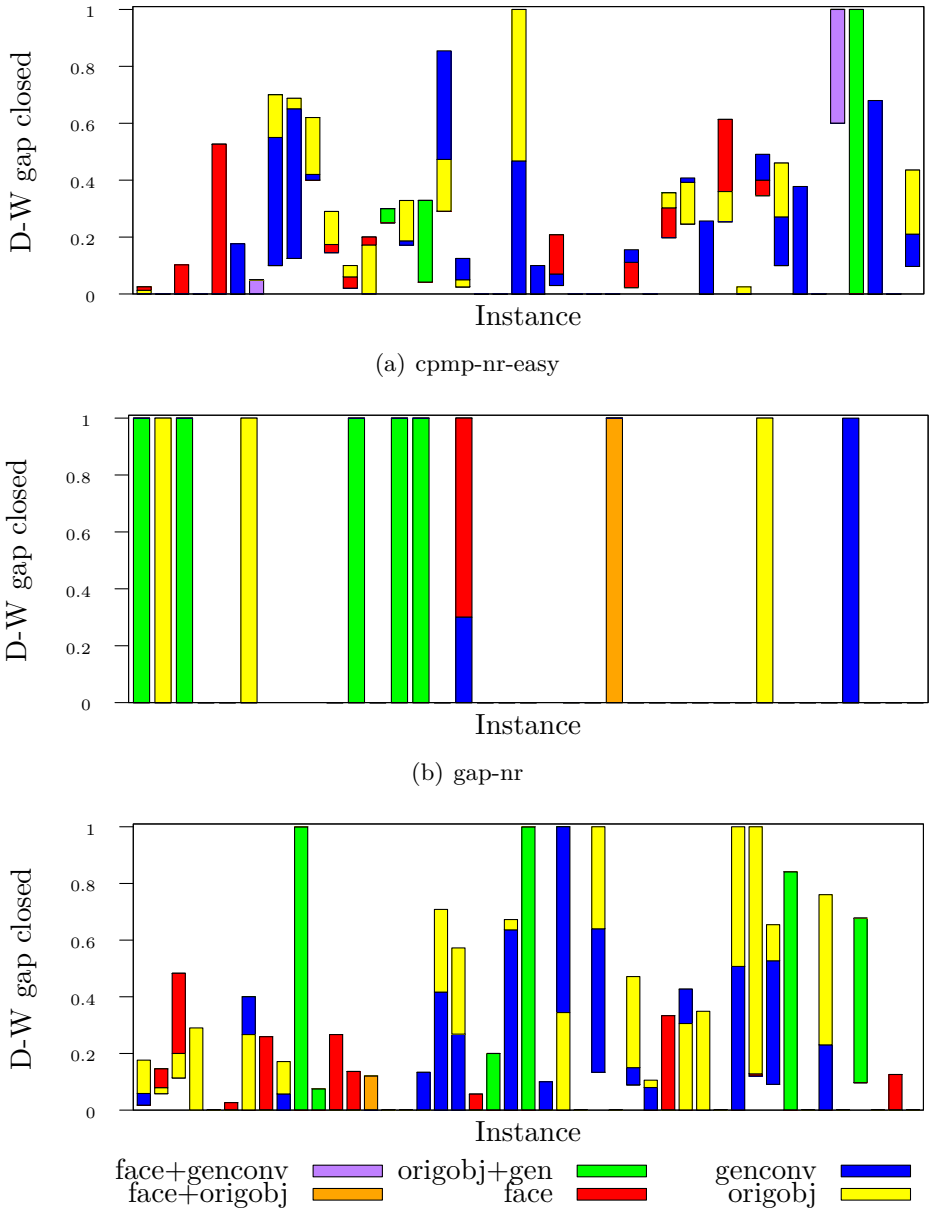
In the first part of the table we calculated the values for all instances of a given test set which were solved to optimality by all observed settings. This way we can explicitly compare how many nodes and how much time was needed to solve these instances to optimality. We consider the easy and middle instances in this part. In the second part of the table we take a look at the other instances and compare the values for complete test sets. The number of examined instances for each test set is stated in brackets after the test set's name.

	genconv						face						origobj					
	Nod	T	Gap cl	Fnd	BT	Aff	Nod	T	Gap cl	Fnd	BT	Aff	Nod	T	Gap cl	Fnd	BT	Aff
bp-nr-e (266)	184.9	27.6	0.000	0.1	20.7	0.07	185.5	19.5	0.000	0.2	9.5	0.11	156.5	75.2	0.000	0.0	59.7	0.00
bp-nr-m (33)	7357.8	598.3	0.000	0.1	86.7	0.04	7460.1	470.2	0.000	0.1	24.0	0.12	3106.2	1054.7	0.000	0.0	223.1	0.00
cs-nr-e (125)	223.8	26.9	0.000	0.4	16.3	0.37	223.0	12.8	0.000	0.4	4.2	0.43	223.8	22.0	0.000	0.0	10.2	0.00
cs-nr-m (149)	1106.1	237.4	0.000	0.8	3.7	0.67	1106.1	223.9	0.000	0.8	0.9	0.63	1106.1	233.7	0.000	0.0	1.8	0.01
coloring-nr-e (5)	65.3	28.4	0.000	2.1	17.1	0.20	65.3	11.2	0.000	2.8	1.9	0.40	65.3	29.8	0.000	0.0	20.8	0.00
coloring-nr-m (5)	1398.3	717.3	0.000	1.5	200.6	0.20	1398.3	359.6	0.000	0.0	49.5	0.00	1398.3	995.9	0.000	0.0	433.6	0.00
cpmp-nr-e (42)	24.4	70.0	0.196	8.9	16.2	0.85	36.7	56.1	0.119	45.3	2.6	0.90	32.5	61.4	0.197	6.6	6.0	0.78
cpmp-nr-m (44)	127.5	520.9	0.106	5.3	83.3	0.68	144.9	545.1	0.076	22.5	10.9	0.86	114.1	496.4	0.112	4.4	23.6	0.68
gap-nr (35)	5.9	11.1	0.172	1.8	1.1	0.54	6.9	11.4	0.020	0.7	0.8	0.20	5.6	10.8	0.149	1.8	0.9	0.57
rap32-nr-e-m	24.2	644.1	0.195	2.4	33.2	0.61	30.9	713.2	0.054	1.1	0.7	0.40	21.7	571.5	0.267	3.7	24.5	0.66
rap64-nr-e-m (29)	9.8	539.2	0.206	1.6	20.5	0.48	11.7	556.3	0.042	0.3	0.5	0.10	10.5	569.4	0.168	1.1	19.0	0.55
gcgjournal-nr-e-m (5)	348.7	70.6	0.290	4.4	0.8	0.80	339.2	66.8	0.000	2.1	0.5	0.80	311.5	66.4	0.245	2.4	0.9	0.80
cpmp-nr-h (80)	640.2	3539.5	0.053	12.9	633.4	0.06	3064.5	3352.1	0.044	101.5	17.3	0.05	1941.1	3357.5	0.057	12.8	34.5	0.05
rap32-nr-h (17)	73.3	3289.8	0.204	10.3	83.8	0.11	78.1	3553.5	0.048	3.2	1.8	0.05	69.7	3327.3	0.142	6.9	48.1	0.17
rap64-nr-h (10)	26.4	3572.8	0.165	2.6	44.6	0.00	28.0	3600.0	0.000	0.4	0.8	0.00	23.7	3585.4	0.142	3.7	33.8	0.10
gcgjournal-nr-h (11)	369.8	3600.0	0.075	140.8	10.7	0.09	382.2	3600.0	0.057	146.3	8.9	0.09	350.7	3600.0	0.062	171.1	8.8	0.09
mkp-u-dis-nr-h (49)	689.5	3118.3	0.049	0.5	10.9	0.08	739.6	3125.6	0.035	0.3	7.7	0.04	551.8	3161.6	0.039	0.3	15.9	0.06
mkp-u-sim-nr-h (49)	559.0	3276.4	0.063	0.4	13.5	0.04	632.1	3276.0	0.058	0.3	9.6	0.02	608.0	3332.2	0.062	0.3	18.7	0.02
mkp-w-dis-nr-h (73)	398.4	3469.0	0.016	0.2	31.8	0.01	439.3	3600.0	0.013	0.2	16.5	0.01	441.2	3466.7	0.016	0.2	18.5	0.01
mkp-w-sim-nr-h (83)	499.7	3316.3	0.048	0.3	26.3	0.02	548.0	3397.3	0.041	0.2	15.2	0.02	578.4	3259.2	0.047	0.3	17.9	0.04

Table 5.1: Test results for the different approaches to obtain an auxiliary basic LP-feasible solution.

In table 5.1 we can see that we are not able to reduce the size of the Dantzig-Wolfe gap for the bin packing, the cutting stock, and the coloring problem instances, although we find cuts for some instances when using the `face` or the `genconv` setting. Since the generated cuts do not affect the dual bound in the root node, the number of nodes stays almost the same for all settings.

The situation changes when we consider the other test sets. The cuts which are found on instances of the other test sets influence the dual bound in the root node. So on shifted geometrical average we are able to close the Dantzig-Wolfe gap at least 5% on all other test sets by the use of one of the considered settings. When considering the remaining test sets where we only examine instances that were solved to optimality by all settings, we close at least 10% and at most 29% of the Dantzig-Wolfe gap measured

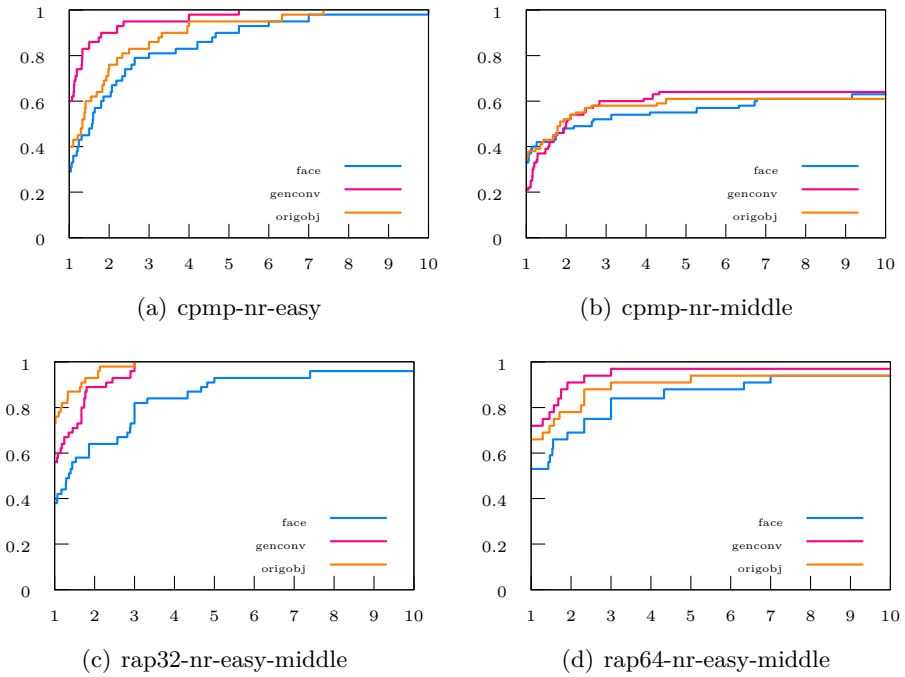


(a) cpmp-nr-easy

(b) gap-nr

(c) rap32-nr-easy-middle

Figure 5.1: Comparison of the Dantzig-Wolfe gap that was closed due to the basis separator with the use of the different objective functions. When we order the settings for a given instance depending on how much of the original gap is closed, the difference between two consecutive settings is marked with the color of the setting that reduces the size of the original gap more on the given instance. If we obtain the same gaps for two settings, we use the color of the combination of both settings.



(a) cpmp-nr-easy (b) cpmp-nr-middle

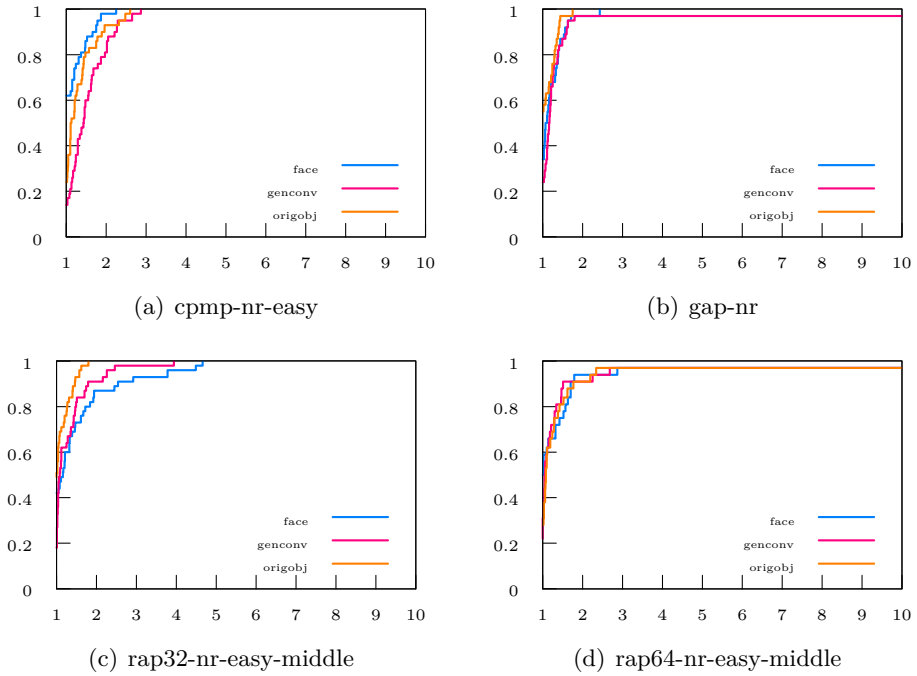
(c) rap32-nr-easy-middle (d) rap64-nr-easy-middle

Figure 5.2: Node performance profiles showing in how many instances (y-axis in percent) a given setting needs at most x times less nodes (factor on the x-axis).

in the shifted geometrical mean over a test set. How much of the gap is closed depends on the observed test set and the setting that we use.

The `genconv` and the `origobj` settings close more of the gap than the `face` setting, while the `face` setting finds more cuts than the other settings. An explanation for this behaviour might be the impact of the original objective function to find an auxiliary basic LP-feasible solution when using the `genconv` or the `origobj` setting. Remember that the `origobj` setting uses nothing but the original objective function and the `genconv` setting uses a generic convex combination of the original objective function together with the face objective function to find an auxiliary basic LP-feasible solution. In contrast to this, the face objective function is independent of the original objective function. Hence, the objective value of an auxiliary basic LP-feasible solution calculated by using the generic convex or the original objective function is potentially lower than the objective value of an auxiliary basic LP-feasible calculated with the help of the face objective function when considering a minimization problem. So cutting off the solutions calculated by using the generic convex or the original objective function results potentially more often

in a change of the dual bound.



(a) cpmp-nr-easy (b) gap-nr

(c) rap32-nr-easy-middle (d) rap64-nr-easy-middle

Figure 5.3: Time performance profiles showing in how many instances (y-axis in percent) a given setting is at most x times slower (factor on the x-axis).

As we can see, the partly closed integrality gap leads to less nodes in the underlying branch-and-bound tree. Except for the gcgjournal-nr-e-m test set, the `face` setting averages more nodes than the other settings to solve the instances to optimality. Depending on the observed test set one of the other two settings needs the least nodes. In figure 5.2 node performance profiles for some test sets are presented. We cannot say which one, the `genconv` or the `origobj` setting, needs less nodes in general, because it depends on the observed test set.

The number of branch-and-bound nodes used on the more difficult test sets is not that informative. Besides the closed integrality gap, the number of instances which were solved within the time limit of 3600 seconds is interesting, but there is no big difference between the observed settings. We solved one instances less on the mkp-u-dissim-h test set when applying the `face` setting instead of one of the other settings, and on the rap32-nr-h test set we solved one instance by applying the `face` setting, two instances by applying the `genconv` setting and three instances when using the `origobj` setting.

On all other test set we solved the same number of instances to optimality within the given time limit no matter which setting we used.

In the end we want to compare the running time of the algorithm when using the different settings. We disregard the test sets where none of the settings have an influence on the given integrality gap. In all test sets but the gcgjournal-nr-e-m test set the use of the `genconv` setting leads to the most time spent in the separator, while using the `face` setting leads to the least time spent in the separator. This is not observable when looking at the overall solution time. When we use the `face` setting, the algorithm averages the least time in the cpmp-nr-e test set to solve the instances to optimality, but in all other test sets either the `genconv` or the `origobj` helps to solve the instances most rapidly. The differences in the solution time are not that significant, but overall the use of the `origobj` setting is a bit faster than the use of the `genconv` or the `face` setting. The time performance profiles in figure 5.3 confirm this.

All in all, we have seen that each setting influences the solution process on some instances of every problem class but the binpacking, the cutting stock, and the coloring problem class. While the `face` setting finds the most cuts, the other two settings help to close more of the given Dantzig-Wolfe gap. This leads to less branch-and-bound nodes and in some cases to a faster solution process. Since the closed gap is the most important issue when evaluating a separation procedure and the `face` setting does not need less time or less nodes than the other settings, the `genconv` setting and the `origobj` setting are more interesting to examine than the `face` setting.
Therefore, we only consider the `genconv` and the `origobj` setting in the following sections.

We have also seen that the time which was spent in the separator is not strictly related to the overall solution time. So we disregard the separation time for the remainder of this thesis except for the part where we analyze the pricing cuts and the chief-and-slave problem. Furthermore, we disregard the percentage of affected instances.

5.2 Comparison to the Master Separator and Analysis of the Underlying Separators

In this section we compare the basis separator to the *master separator*, which is by default enabled in `GCG`, and analyze the use of the underlying `SCIP` separators as well. In section 3.1 we already explained how to use separators which separate an arbitrary LP-feasible solutions to generate cutting planes which cut off the current solution of the linear master problem and are formulated in the original problem's solution space. The *master separator* applies the *clique separator*, the *complemented mixed integer rounding cuts separator*, the *flowcover separator*, the *implied bounds separator*, and the *multi-commodity-flow network cut separator* in order to generate cutting planes.

In table 5.2 we present the number of cutting planes which were found by the applied `SCIP` separators measured in shifted geometrical mean over a test set. We do not differentiate between cutting planes generated by the Gomory cuts separator and the strong Chvátal-Gomory cuts separator. Both of them are listed under strong Chvátal-Gomory cuts. Note that $\{0, \frac{1}{2}\}$-cuts are a special kind of Chvátal-Gomory cuts.
Each column belongs to an applied separator and is divided into subcolumns which contain the number of cutting planes found by the basis separator (b) and the master separator (m), respectively. cs, bp and col

	clique		cmir		flowcover		impbds		mcf		oddcycle		strongcg		zerohalf	
Name	b	m	b	m	b	m	b	m	b	m	b	m	b	m	b	m
cpmp-nr-e	0.0	0.0	0.3	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	-	5.1	-	3.3	-
cpmp-nr-m	0.0	0.0	0.1	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	-	1.5	-	1.2	-
cpmp-nr-h	0.0	0.0	0.3	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	-	5.8	-	3.7	-
gap-nr	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-	1.0	-	0.5	-
rap32-nr-e-m	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	-	1.8	-	0.3	-
rap32-nr-h	0.0	0.0	0.1	0.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0	-	9.3	-	0.6	-
rap64-nr-e-m	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	-	1.6	-	0.2	-
rap64-nr-h	0.0	0.0	0.2	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	-	2.2	-	0.1	-
gcgjournal-nr-e-m	0.0	0.0	3.1	1.0	0.9	1.0	0.0	0.0	0.6	0.0	0.0	-	0.0	-	0.0	-
gcgjournal-nr-h	0.6	0.7	94.9	32.9	27.7	8.1	3.5	4.0	7.7	7.9	0.3	-	4.7	-	0.9	-

Table 5.2: Number of cutting planes which were found by the separators in the basis separator and the master separator. Additionally, each non-zero entry is colored in blue (basis separator) or in red (master separator).

The master separator does not find any cutting planes on the capacitated p-median problem, the generalized assignment problem, and the resource allocation problem instances, but it finds cutting planes on some instances of the gcgjournal test sets. In the previous sections we have already seen that the basis separator finds cuts on all observed test sets. So this is a huge advantage in comparison to the master separator. Furthermore, the basis separator finds a lot more cuts on shifted geometrical average than the master separator.

Each separator used by the master separator finds cuts on the gcgjournal instances. The complemented mixed integer rounding separator and the flowcover separator find a lot more cuts than the other separators which are applied by the master separator, but both separators find even more cuts when separating auxiliary LP-feasible solutions in the basis separator. A reason for this dominance of the basis separator over the master separator might be the use of additional valid inequalities which strengthen the original LP relaxation. When applying the basis separator, we iteratively generate cutting planes which cut off the current auxiliary LP-feasible solution. These cuts either cut off the current solution of the linear master problem, too, or strengthen the LP relaxation. Therefore, the basis separator uses more information in order to generate cutting planes.

Besides the just described advantage of the basis separator, we are able to use more separators in order to generate cutting planes which cut off a LP-feasible solution. On most test sets, the strong Chvátal-Gomory cuts separator and the $\{0, \frac{1}{2}\}$-cuts separator generate the majority of all found cuts, whereas the oddcycle separator only finds a few cuts on the gcgjournal-nr-h test set. The complemented mixed integer rounding separator as well as the flowcover separator find some cuts on almost every test set, where they find by far the most cutting planes on the gcgjournal instances. On the corresponding test sets, they even generate significantly more cutting planes than all other used separators.

In this section we have seen that the basis separator generates significantly more cutting planes than the master separator. Furthermore, the number of cuts generated by the underlying `SCIP` separators differs extremely depending on the observed test set. All separators but the Chvátal-Gomory cuts separators perform much better on the gcgjournal instances than on the other instances, which formulate combinatorial optimization problems. Especially the clique separator, the implied bounds separator, the multi-commodity-flow network cut separator, and the oddcycle separator do not find any cuts on these instances, although we strengthen the LP relaxtion during the basis

	gen				gen-obj				gen-obj-redcost				gen-redcost			
	Nod	T	Gap cl	Fnd	Nod	T	Gap cl	Fnd	Nod	T	Gap cl	Fnd	Nod	T	Gap cl	Fnd
cpmp-nr-e (42)	24.4	70.0	0.196	8.9	40.0	54.2	0.116	26.6	41.6	44.5	0.029	4.6	34.8	45.0	0.139	7.5
cpmp-nr-m (40)	133.2	501.1	0.102	4.5	134.0	596.7	0.095	28.0	177.6	484.9	0.029	9.1	128.8	398.6	0.084	3.5
gap-nr (35)	5.9	11.1	0.172	1.8	5.7	9.7	0.219	2.0	5.5	10.2	0.135	1.1	5.1	10.0	0.149	1.4
rap32-nr-e-m (38)	21.1	579.8	0.197	2.5	22.6	577.7	0.234	2.4	17.0	662.9	0.282	9.9	19.9	546.7	0.255	8.8
rap64-nr-e-m (30)	9.6	569.1	0.224	1.6	9.9	642.4	0.205	1.4	6.9	499.5	0.357	9.0	7.7	632.6	0.358	7.7
gcgjournal-nr-e-m (4)	141.9	37.2	0.565	22.4	119.2	40.9	0.191	18.3	168.9	50.6	0.035	10.5	187.7	62.1	0.107	5.5

Table 5.3: Shifted geometrical test results for the reduced cost cuts and the original objective cuts.

separation. It would be interesting to investigate the reasons for this behaviour and it might be a subject for future research.

5.3 Impact of the Additional Valid Inequalities

In this section we analyze the influence of the additional valid inequalities, we presented in this thesis. We introduced the original objective cuts, the pricing cuts, and the reduced cost cuts in section 4.3.1. Furthermore, we presented the chief-and-slave algorithm 4.2 in section 4.3.2.

To begin with, we evaluate the impact of the original objective and the reduced cost cuts. Note that in each separation round we generate exactly one original objective cut and K reduced cost cuts, one for each pricing problem.
In table 5.3 we compare the results achieved by applying the `genconv` setting and adding the original objective cuts (`genconv-obj`), the reduced cost cuts (`genconv-redcost`), both types of cutting planes (`genconv-obj-redcost`), or no additional inequalities.

As we can see, the original objective cuts help us to generate significantly more cuts on shifted geometrical average on the capacitated p-median problem test sets and the reduced cost cuts in combination with the original objective cuts lead to more cuts in the resource allocation problem test sets in comparison to the setting without any further inequalities. The `genconv` setting generates the most cuts on the gcgjournal-nr-e-m test set. The additional inequalities do not improve the dual bound in the capacitated p-median problem test sets or the gcgjournal test set. On the contrary, at least one of the settings using additional inequalities closes more of the Dantzig-Wolfe gap than the

`genconv` setting on the other test sets when comparing the shifted geometrical means over a test set. This transfers to the number of used branch-and-bound nodes except that the `genconv-redcost` setting needs the least nodes in the cpmp-nr-middle test set.

The results for the basis separator including both original objective cuts and reduced cost cuts are similar to the the results of the basis separator including just the reduced cost cuts, whereby the difference to the `genconv` setting is even bigger. Applying the `genconv-obj-redcost` setting leads to less nodes and a smaller gap than applying the other settings on the capacitated p-median problem test sets, but the contrary holds on the resource allocation problem test sets. Whereas on the generalized assignment instances the setting which uses both types of inequalities reduces the size of the Dantzig-Wolfe gap less than the setting with no additional inequalities. On the contrary, the `genconv-obj-redcost` setting solves the instances faster to optimality and therefore it needs less branch-and-bound nodes when comparing the shifted geometrical mean of these values over the test set.

Now, we compare the solution time of the different settings. Depending on the observed test set, the solution times clearly distinguish from each other. Every setting but the `gen-obj` setting is the fastest one on at least one test set. It is not possible to determine the overall fastest setting and apparently, the solution time is not cohererent with the size of the integrality gap in the root node or the number of used branch-and-bound nodes. An explanation for this behaviour might be the already fast default setting. If we only find few cuts which hardly affect the solution process, we might solve the problem faster to optimality than in the case of finding a lot of cuts which rather affect the solution process. In section 5.4 we analyze the performance of the default setting in comparison to the use of the basis separator.

Next, we consider the pricing cuts and valid inequalities, generated by the chief-and-slave algorithm. In comparison to the previously evaluated cuts, the number of pricing cuts is not strictly limited and the chief-and-slave cuts are not obtained during the solution process without any further effort. Additionally, we have to solve the pricing problems as a MIP in order to generate pricing cuts.

The impact of the pricing cuts (`genconv-ppcuts`) and the chief-and-slave algorithm (`genconv-chief`) on the performance of the basis separator with the use of the generic convex objective function can be seen in table 5.4. Similarly to the previously analyzed

additional valid inequalities, the inequalities do not reduce the integrality gap in the root node in comparison to the `genconv` setting with the exception that the chief-and-slave algorithm helps to reduce the size of the integrality gap by 0.5% more than the basis separator without any additional inequalities. The impact of both approaches is a bit significant on the other test sets. Either of them reduces the Dantzig-Wolfe integrality gap by at least 2% and at most 8% more than the `genconv` setting measured in the shifted geometrical mean over a test set. This partly transfers to the number of used branch-and-bound nodes.

The problem with the additional inequalities affect the solution time. If we disregard the easy capacitated p-median instances and the generalized assignment instances, which can for the most part also be solved within 60 seconds, the solution time in comparison to using the basis separator without any further inequalities increases drastically. In the following we give the reasons for the increased solution time.

First, we consider the pricing cuts. On the one hand, we solve the pricing problems as mixed integer programs. This has an effect on all problems with knapsack pricing problems since in this case `GCG` solves the pricing problem by applying a combinatorial algorithm. On the other hand, in order to solve the pricing problems, a huge number of cutting planes might be generated. All these cutting planes are temporarily added to the current original LP relaxation. Hence, solving this linear program as well as generating cutting planes for the current basic LP-feasible solution takes more time and on some instances it even leads to memory space problems.

In contrast to the potentially huge number of pricing cuts, the number of cutting planes, calculated by the chief-and-slave algorithm is limited by the dimension n of the underlying vector space. Note that each solution of the original LP relaxation is a n-dimensional vector. As we can see, the number of generated cuts cannot be the reason for the drastically increased solution time. The large solution time is mainly a result of the calculation of the chief-and-slave cutting planes. In each step we alternatingly solve a linear program and an integer program in order to generate a valid inequality, where the number of steps is just limited by the number of possible solutions to the corresponding pricing problem. Thus, the number of steps can be extremely large. Furthermore, in each step we add a new constraint to the chief problem such that the chief problem can be extremely large as well. On some instances, this also leads to memory space problems.

	genconv					genconv-new					genconv-ppcuts				
	Nod	T	Gap cl	Fnd	BT	Nod	T	Gap cl	Fnd	BT	Nod	T	Gap cl	Fnd	BT
cpmp-nr-easy (19)	23.9	66.3	0.190	8.7	14.6	32.6	66.4	0.167	9.5	7.6	37.8	30.4	0.129	9.1	7.5
cpmp-nr-middle (32)	122.4	381.3	0.128	6.9	54.1	105.4	428.8	0.133	6.6	47.0	117.0	486.6	0.075	5.4	170.9
gap-nr (33)	5.6	8.1	0.183	1.9	1.0	5.1	7.0	0.208	2.0	1.8	6.4	8.7	0.217	3.0	0.8
rap32-nr-easy-middle (37)	23.6	585.5	0.206	2.6	31.2	22.2	923.4	0.230	2.6	121.3	23.7	728.0	0.201	2.8	105.4
rap64-nr-easy-middle (26)	9.8	468.6	0.233	1.9	19.6	-	-	-	-	-	7.1	688.3	0.317	1.8	52.8

Table 5.4: Shifted geometrical test results for the pricing cuts and the chief-and-slave algorithm.

All in all, additional valid inequalities in combination with the basis separator can be useful to further improve the dual bound in the root node, but it can also have a negative impact on the performance of the basis separator. Especially, the pricing cuts and the chief-and-master cuts lead to a larger solution time and memory space problems. With the exception of the chief-and-slave cuts, the problem is that the original LP relaxation, which is used in the basis separator, becomes too big. We could select only a subset of the inequalities to enter this LP depending on the satisfaction of given orthogonality properties. This is also done by `SCIP` when adding cutting planes to the current LP. A subject for future research might be to adopt this selection of cutting planes. This could lead to better cutting planes and an overall more competitive performance of the basis separation algorithm in combination with additional valid inequalities.

5.4 Comparison to the Default Setting

1. use cgmip

Finally, we want to compare the performance of `GCG` using the basis separator to the performance of `GCG`'s default setting. Therefore, we use the `genconv` and the `origobj` setting, which we have already examined in the previous sections. We apply these settings without any additionally generated valid inequalties, because all approaches to generate additional valid inequalities either lead to a drastically increased solution time or the generated inequalities affected the overall performance on some test sets positively as well as on some test sets negatively as seen in section 5.3. So none of the presented approaches has a sole favorable influence on the solution process.

In table 5.5 we can see the results of `GCG`'s default setting in comparison to the use of the basis separator 4.1 when using the generic convex or the original objective function. The

	default		genconv				origobj			
	Nodes	Time	Nodes	Time	gap closed	bCuts	Nodes	Time	gap closed	bCuts
bp-nr-easy	186	6.2	186	19.0	0.000	0.0	187	43.9	0.000	0.0
bp-nr-middle	8 941	286.6	8 941	480.4	0.000	0.0	8 901	685.0	0.000	0.0
cs-nr-easy	222	10.4	222	26.7	0.000	0.0	222	21.8	0.000	0.0
cs-nr-middle	1 136	218.6	1 136	238.8	0.000	0.0	1 136	235.4	0.000	0.0
coloring-nr-easy	65	10.9	65	28.4	0.000	0.0	65	29.8	0.000	0.0
coloring-nr-middle	1 398	455.1	1 398	717.3	0.000	0.0	1 398	995.9	0.000	0.0
cpmp-nr-easy	43	35.1	24	70.0	0.196	4.5	32	61.4	0.197	3.0
cpmp-nr-middle	180	331.7	131	543.1	0.103	2.8	118	517.7	0.109	2.5
gap-nr	8	10.2	5	11.1	0.172	0.9	5	10.8	0.149	0.9
rap32-nr-easy-middle	35	733.0	24	644.1	0.195	1.0	21	571.5	0.267	1.1
rap64-nr-easy-middle	12	621.2	9	571.1	0.224	0.6	10	603.1	0.187	0.6
gcgjournal-nr-easy-middle	338	63.5	334	68.0	0.348	2.2	302	65.0	0.300	2.2
cpmp-nr-hard (80)	1377.3	3560.0	640.2	3539.5	0.053	12.9	1941.1	3357.5	0.057	12.8
rap32-nr-hard	82	3600.0	73	3289.8	0.204	3.1	69	3327.3	0.142	2.2
rap64-nr-hard	31.4	3600.0	26.4	3572.8	0.165	2.6	23.7	3585.4	0.142	3.7
gcgjournal-nr-hard	492	3600.0	369	3600.0	0.075	23.4	350	3600.0	0.062	22.4
mkp-lamani-uncor-dissim-nr	744	2585.2	645	2522.5	0.048	0.4	515	2553.4	0.039	0.3
mkp-lamani-uncor-sim-nr	544	3281.7	512	3303.6	0.053	0.3	554	3360.6	0.052	0.3
mkp-lamani-weak-dissim	488	3615.9	398	3487.5	0.005	0.2	439	3486.9	0.005	0.2
mkp-lamani-weak-sim	642	3376.1	577	3118.8	0.032	0.3	631	3068.4	0.031	0.3

Table 5.5: Shifted geometrical test results for the default setting and two different approaches to obtain an auxiliary basic LP-feasible solution.

form of this table is similar to table 5.1 excluding the percentage of affected instances and the time that was spent in the basis separator.
As might be expected, the default setting outperforms the other settings on the bin packing, the cutting stock and the coloring instances since the size of the Dantzig-Wolfe gap is not reduced by the basis separator, while additional time is spent to generate cutting planes. Hence, the number of used nodes is almost equal for all examined settings, but the default setting's solution time is much lower than the solution time of the other two settings.

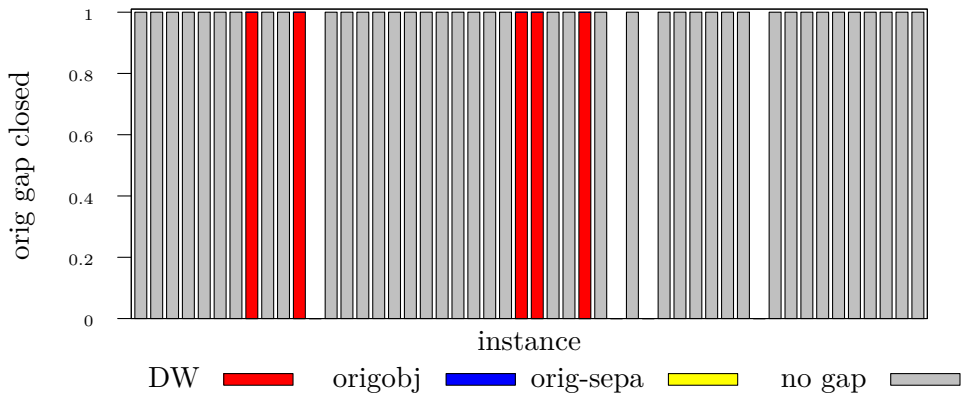


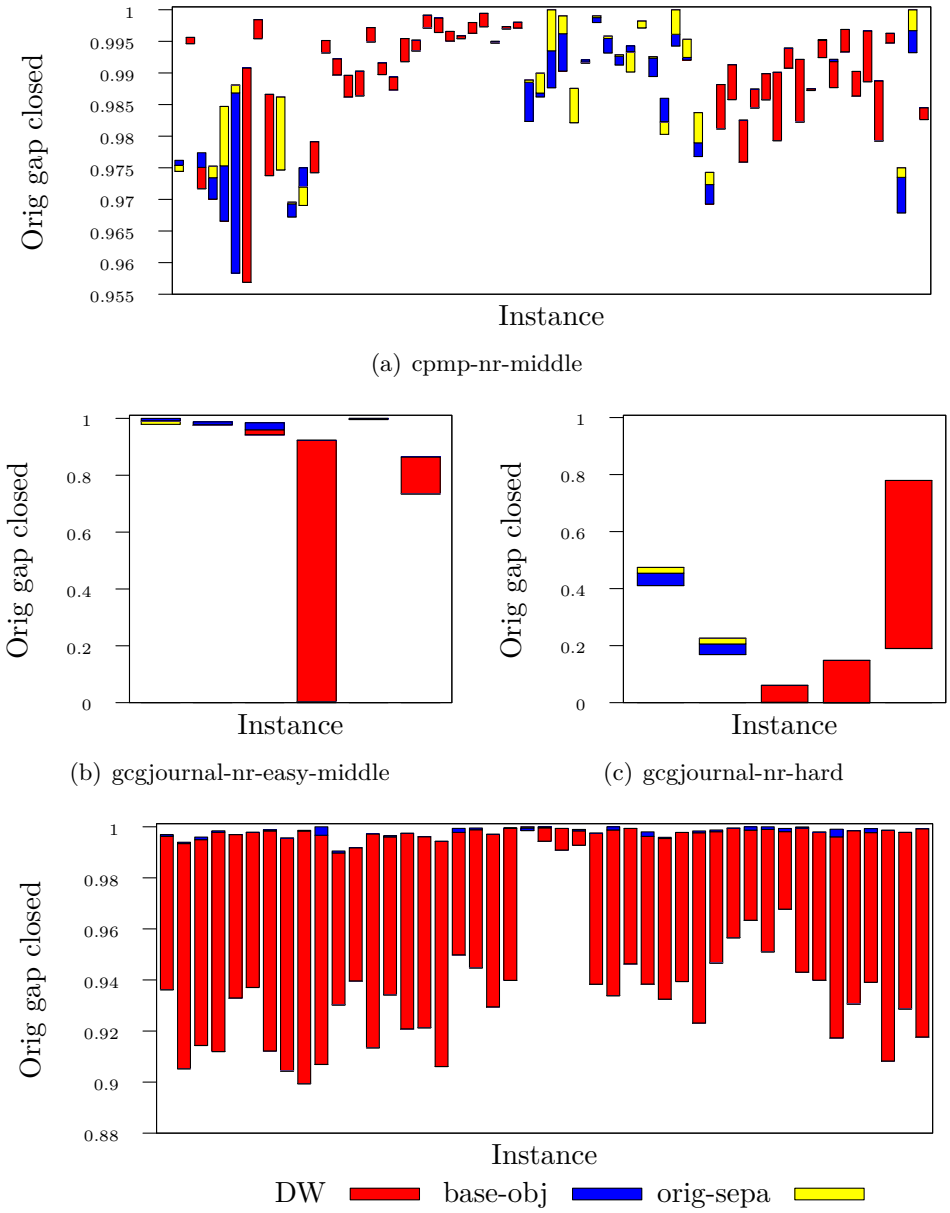
Figure 5.4: Comparison of the original gap that was closed due to the Dantzig-Wolfe reformulation, `GCG` with the use of the basis separator, and `SCIP` with the use of separation. The difference between the closed gaps is similarly colored to figure 5.1.

In figure 5.1 we compare how much of the original integrality gap, which is given by the difference of the optimal solution value and the dual bound obtained due to the original LP relaxation, is closed in the root node by applying `GCG`'s default setting, `GCG`'s `origobj` setting, and `SCIP` both with and without separation to the one of the binpacking test sets. As already mentioned, the basis separator does not find any cuts which help to reduce the size of the original integrality gap. This also holds for `SCIP` with separation. Note that the dual bound given by the original LP relaxation is already close to the optimal solution value, but the dual bound given by the linear master problem, which is used by `GCG`, is even better. Most often the Dantzig-Wolfe reformulation either closes the original integrality gap completely or does not reduce the size of the gap at all. These observations can also be made when considering cutting stock and coloring problem instances except that the dual bound given by the original LP relaxation is not that often equal to the optimal solution value.

In section 5.1 we have already seen that the basis separator is able to generate cutting planes which help to reduce the integrality gap for instances of all other test sets. In figure 5.5 we compare the reduction of the original integrality gap for some test sets similarly to figure 5.1. Of course, the figures for test sets where the basis separator does affect the dual bound and for test sets where it does not are different, but as we can see the subfigures in figure 5.5 also differ from each other, although the basis separator affects the dual bound in all evaluated test sets.
First of all, we can see that the original LP relaxation seems to be weak in comparison to the relaxation obtained by applying Dantzig-Wolfe reformulation, because the original integrality gap is much bigger than the Dantzig-Wolfe integrality gap. Furthermore, the cutting planes generated by `SCIP` significantly reduce the original integrality gap, too.

When considering the capacitated p-median test set in figure 5.5, most often one of the following two cases applies. On the one hand, there are instances where the dual bound given by the linear master problem is better than the one given by `SCIP` with separation in the root node. On these instances, the basis separator does not affect the already good dual bound given by the linear master problem most of the time. On the other hand, when `SCIP`'s dual bound in the root node which is obtained by the help of separation is better than the dual bound given by the linear master problem, the basis separator generates cutting planes which improve the dual bound given by the linear master problem. This transfers to both the gcgjournal instances as we can see in figure 5.5 and the generalized assignment problem instances.

Note that on several instances `SCIP` with separation calculates a better dual bound in the root node than `GCG` with the use of the basis separator. Since both use the same separators to cut off a basic LP-feasible solution and we disabled propagation in `SCIP`, which is by default applied after a separation round, there are only two reasons why this could happen. First, the calculated basic LP-feasible solutions may differ from each other. Even if we use `GCG`'s `origobj` setting, it is not guaranteed that the same optimal solution is calculated. Hence, we generate different cutting planes. And second, `SCIP` does not apply all generated cuts to the current linear program. Only cutting planes which efficiently cut off a given solution with respect to the cutting plane's norm together with a given minimum cut efficiency and that satisfy some given orthogonality properties are used. This leads to a smaller linear program. However, the basis separator only uses the minimum cut efficiency, which results in a potentially greater linear program. Besides



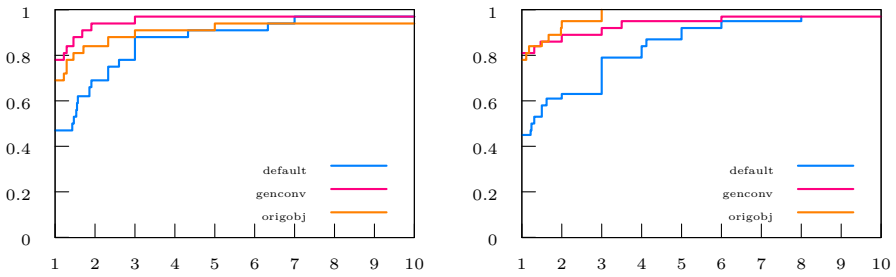
(a) cpmp-nr-middle

(b) gcgjournal-nr-easy-middle

(c) gcgjournal-nr-hard

(d) rap32-nr-easy-middle

Figure 5.5: Comparison of the original gap that was closed due to the Dantzig-Wolfe reformulation, `GCG` with the use of the basis separator, and `SCIP` with the use of separation. The difference between the closed gaps is colored similarly to figure 5.1 with the exception that we use the color of the Dantzig-Wolfe reformulation if the corresponding dual bound is equal to the dual bound that was computed with the use of the basis separator.



(a) rap64-nr-easy-middle (b) gap-nr

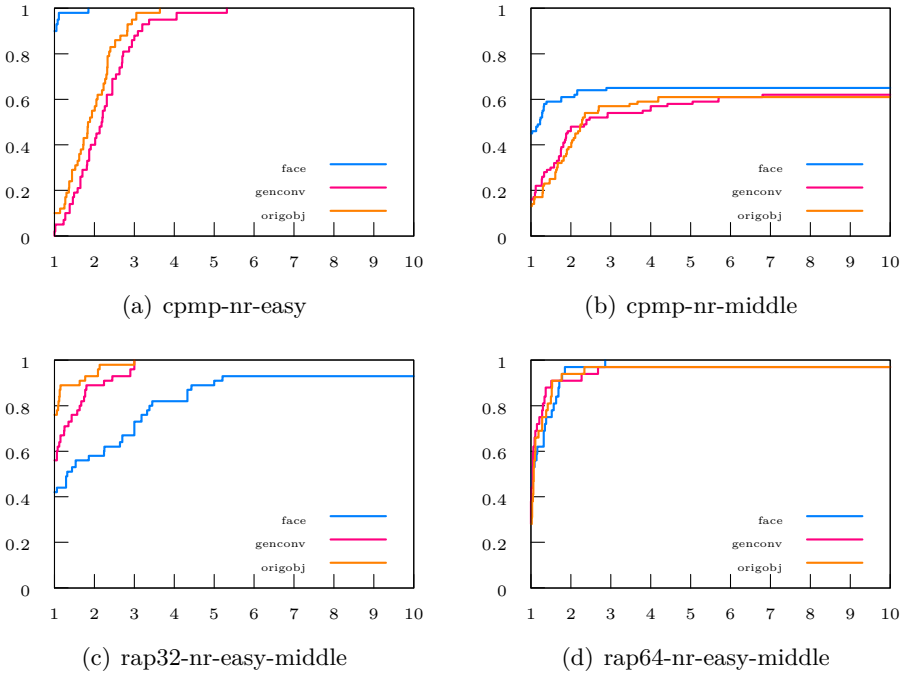
Figure 5.6: Node performance profiles showing in how many instances (y-axis in percent) a given setting needs at most x times less nodes (factor on the x-axis).

the fact that solving a greater LP needs more solution time, the efficiency of generated cuts could be reduced due to a potentially greater norm. For example, Chvátal-Gomory inequalities tend to have more non-zero coefficients, which leads to a potentially greater norm.

The subfigures in figure 5.5 corresponding to the resource allocation problem are completely different from the subfigures of the other problem classes. As we can see, the dual bound obtained by the Dantzig-Wolfe reformulation is much better in comparison to the dual bound in the root node, which is due to `SCIP` with the use of separation. Contrary to the previously examined test sets, the basis separator still improves the dual bound obtained by the Dantzig-Wolfe reformulation. So in this case the Dantzig-Wolfe reformulation combined with the cutting planes formulated in the original problem's solution space leads to an improved dual bound in comparison to the application of either of them. This holds for almost all resource allocation problem instances. For the time being, we do not know the reasons why the combination improves the dual bound, although the cutting planes do not have a strong impact on the dual bound due to the original LP relaxation. It would be interesting to investigate this and it could be a subject for future research.

Because of the smaller integrality gap in comparison to `GCG`'s default setting, both settings using the basis separator average less branch-and-bound nodes than the default setting. This can also be seen in the node performance profiles in figure 5.6.

At the end we examine how the improved dual bound in the root node and the reduced number of nodes in case of using the basis separator affects the overall solution time.



(a) cpmp-nr-easy (b) cpmp-nr-middle

(c) rap32-nr-easy-middle (d) rap64-nr-easy-middle

Figure 5.7: Time performance profiles showing in how many instances (y-axis in percent) a given setting is at most x times slower (factor on the x-axis).

	default	genconv	origobj
cpmp-nr-h (80)	2	6	6
rap32-nr-h (32)	0	2	3
rap64-nr-h (10)	0	1	1
gcgjournal-nr-h (11)	0	1	1
mkp-u-dissim-nr-h (49)	1	3	3
mkp-u-sim-nr-h (49)	1	1	1
mkp-w-dissim-nr-h (73)	0	1	1
mkp-w-sim-nr-h (83)	0	2	2

Table 5.6: Number of solved instances for the more difficult test sets.

The number of solved instances for each relevant test set is presented in table 5.6. We are able to solve at least one instance more when using the basis separator in comparison to the default setting with the exception that we solved the same number of instances of the mkp-u-dissim-h test set. Since it is not possible to reasonably analyze the solution time on the difficult instances, we take a look at the test sets where the instances were solved to optimality within the given time limit.

As we can see in table 5.5, the solution time increases slightly on the generalized assignment problem test set and the gcgjournal-nr-easy-middle test set, but it increases significantly on the capacitated p-median problem test sets when considering the shifted geometrical mean over a test set. In contrast, the solution time decreases on the resource allocation problem test sets. The time performance profiles in figure 5.7 confirm this observation.

All in all, the proposed basis separator using either the original objective function or the generic convex objective function affects the solution process on the majority of the observed problem classes and it reduces the integrality gap in the root node as well as the number of used branch-and-bound nodes in comparison to the use of the default setting. On the one hand, the solution time increases on most instances when using the basis separator. On the other hand, the solution process on resource allocation problem instances is accelerated. Additionaly, on the majority of the more difficult test sets where the basis separator reduces the integrality gap, we solve a bit more instances to optimality within the given time limit by applying basis separation than without the use of the basis separator.

Chapter 6

Conclusion

1. comparison to other dual bounds (orig+sepa, cgmip)
2. how can we accelerate the solution process then? cancel/abort after a few iterations?
3. what is to do next? separation in master on general problems?

In this thesis we presented a separation procedure – the basis separator – which is able to generate cutting planes in the original problem's solution space by the use of basis information. This separator was initially introduced by Range [3], but, to our knowledge, we are the first to present implementational results for a separator like this. Therefore, we implemented a separator in the generic branch-price-and-cut solver `GCG` [2]. Furthermore, we discussed some variations and extensions which improve the performance of the basis separator. Especially the use of the generic convex objective function leads to cutting planes which improve the dual bound in the root node as well as reduce the number of used branch-and-bound nodes in comparison to the use of the originally proposed face objective function. Using the original objective function instead of the face objective function also improves the performance of the basis separator, but we were not able to figure out if the original objective function or the generic convex objective function improves the basis separator's performance the most. Depending on the observed test set, one of them reduces more of the integrality gap and solves the instances faster to optimality.

We also introduced valid inequalities which can be used to strengthen the original LP relaxation and we tested their impact on the basis separation procedure. Unfortunately, the additional inequalities do not have a sole favorable impact. On the one hand, the

inequalities lead to the separation of less efficient cutting planes and, on the other hand, we generated a huge number of valid inequalities such that solving the original LP relaxation in order to calculate an auxiliary basic LP-feasible solution takes much longer. A task of future research might be to add only a useful subset of these inequalities similar to `SCIP`'s selection of cutting planes. Only valid inequalities which satisfy some given orthogonality properties are applied to the current LP relaxation.

Although, the basis separator reduces the integrality gap in the root node and the number of used branch-and-bound nodes, it does not accelerate the solution process in comparison to `GCG`'s default setting with the exception of resource allocation problem instances with block diagonal structure. The just mentioned selection of inequalities which are applied to the current LP relaxation may change this. Since we iteratively add all cutting planes which cut off the auxiliary basic LP-feasible solution for the original LP relaxation, adding a useful subset of these cutting planes may accelerate the basis separation procedure and lead to an overall reduced solution time.

Another subject of future research might be the analysis of the problem classes which are affected by the basis separator. We have seen that problem classes with much symmetry like the bin packing problems or the coloring problems are not affected by the basis separator, although a few cutting planes are generated. In addition, it might be difficult to separate cutting planes formulated in the original problem's solution space for problems where the pricing problem's LP relaxation is very weak. All problem classes where the basis separator reduces the integrality gap in the root node have dissimilar pricing problems. We guess that this helps to generate cutting planes which cut off the current solution and it would be interesting to investigate the influence of the asymmetric structure.

Appendices

Appendix A

Problem Classes

In this part of the appendices we give the problem definitions and the original formulation for each problem class. We specify which constraints remain in the master problem and which constraints enter which pricing problem. This leads to a clearly defined extended formulation if the reformulation approach, namely convexification or discretization, is stated, too. By default, GCG uses the discretization approach to reformulate the original problem.
Furthermore, we name the test sets which we used in chapter 5. do this!

A.1 The Bin Packing Problem

Consider a set of bins J with capacity C and a set of items I with given weights w_i for all $i \in I$. The *bin packing problem* is to minimize the number of used bins such that each item is packed into a bin respecting the given capacity. This problem can be formulated as follows:

$$
\begin{aligned}
\min \quad & \sum_{j \in J} y_j & & \\
\text{s.t.} \quad & \sum_{j \in J} x_{ij} = 1 & \forall i \in I & \quad \text{(A.1)} \\
& \sum_{i \in I} w_i x_{ij} \leq C y_j & \forall j \in J & \quad \text{(A.2)} \\
& x_{ij} \in \{0,1\} & \forall\, i \in I,\ j \in J & \quad \text{(A.3)} \\
& y_j \in \{0,1\}. & \forall\, j \in J & \quad \text{(A.4)}
\end{aligned}
$$

This formulation is used as the original problem formulation and the extended formulation is obtained by independently reformulating the constraints (A.2) together with the binary constraints (A.3) and (A.4) for all $j \in J$. So the constraints (A.1) stay in the master problem and we obtain a knapsack pricing problem for each $j \in J$ with constraints:

$$\begin{aligned} \sum_{i \in I} w_i x_{ij} &\leq C y_j \\ x_{ij} &\in \{0,1\} \qquad \forall\, i \in I \\ y_j &\in \{0,1\}. \end{aligned}$$

For our computations we used bin packing problem instances of [29], [30], and, some randomly generated instances. Furthermore, we used the bin packing versions of the cutting stock problem instances.

A.2 The Cutting Stock Problem

The *cutting stock problem* is similar to the bin packing problem despite the fact that each item $i \in I$ has to be packed d_i times, where $d_i \in \mathbb{N}$ is a given demand of item $i \in I$, and that an item $i \in I$ can be packed multiple times into a an individual bin. This is why the items are called orders and the bins are called rolls in the context of the cutting stock problem. Using the same notations as in the bin packing context, we formulate the cut packing problem as

$$\begin{aligned} \min \quad & \sum_{j \in J} y_j \\ \text{s.t.} \quad & \sum_{j \in J} x_{ij} = d_i && \forall i \in I && \text{(A.5)} \\ & \sum_{i \in I} w_i x_{ij} \leq C y_j && \forall j \in J && \text{(A.6)} \\ & x_{ij} \in \mathbb{Z}_{\geq 0} && \forall\, i \in I,\ j \in J && \text{(A.7)} \\ & y_j \in \{0,1\} && \forall\, j \in J. && \text{(A.8)} \end{aligned}$$

Similar to the bin packing problem we independently reformulate the constraints (A.6). This leads to the same pricing problems as in the reformulated bin packing problem, one knapsack pricing problem for each stock $j \in J$.
We tested the basis separator on instances that were presented in [31], [32], and [33].

Additionally, we used the cutting stock versions of the bin packing problem instances and cutting stock problem instances generated with `CUTGEN1` [34].

A.3 The Vertex Coloring Problem

Given a graph $G = (V, E)$, where V is the set of vertices and E the set of edges, together with a set of colors C with $|C| = |V|$. We want to assign a color to each vertex such that vertices assigned with the same colour are not adjacent. This problem is called the *vertex coloring problem* and an integer linear program formulation is given by

$$\begin{aligned}
\min \quad & \sum_{c \in C} y_c & & \\
\text{s.t.} \quad & \sum_{c \in C} x_{vc} = 1 & \forall v \in V & \quad \text{(A.9)} \\
& x_{vc} \leq y_c & \forall v \in V,\ c \in C & \quad \text{(A.10)} \\
& x_{uc} + x_{vc} \leq 1 & \forall \{u, v\} \in E,\ c \in C & \quad \text{(A.11)} \\
& x_{vc} \in \{0, 1\} & \forall\ v \in V,\ c \in C & \quad \text{(A.12)} \\
& y_c \in \{0, 1\} & \forall\ c \in C. & \quad \text{(A.13)}
\end{aligned}$$

We use this formulation as the original problem and obtain the extended formulation by independently reformulating the constraints (A.10) together with (A.11). This leads to a maximum wighted independent set pricing problem for each color $c \in C$.
We used instances that are stated in the OR library [35] and some additional instances.

A.4 The Capacitated p-Median Problem

Suppose a set of locations I and a subset $J \subseteq I$ of medians is given. Each location $i \in I$ is assigned with a demand q_i and each median $j \in J$ with a capacity Q_j. Additionally, let $p \in \mathbb{N}$ be a natural number and let $d_{ij} \in \mathbb{Z}_{\geq 0}$ be the distance of location $i \in I$ and median $j \in J$. Our goal is to minimize the used distances, while assigning each location to a median, respecting the capacities and using exactly p medians. This problem is called the *capacitated p-median problem* and we use the following formulation as the

original problem:

$$\begin{aligned}
\min \quad & \sum_{i\in I}\sum_{j\in J} d_{ij}x_{ij} & & \\
\text{s.t.} \quad & \sum_{j\in J} x_{ij} = 1 & \forall i \in I & \quad (A.14)\\
& \sum_{j\in J} y_j \leq p & & \quad (A.15)\\
& \sum_{i\in I} q_{ij} \leq Q_j y_j & \forall j \in J & \quad (A.16)\\
& x_{ij} \in \{0,1\} & \forall\ i \in I,\ j \in J & \quad (A.17)\\
& y_j \in \{0,1\} & \forall\ j \in J. & \quad (A.18)
\end{aligned}$$

The extended formulation is obtained by independently reformulating the constraints (A.16). For each median $j \in J$ exists a knapsack pricing problem just like in the reformulated bin packing and the reformulated cutting stock problem.
A majority of the observed instances is available in the OR library [35].

A.5 The Generalized Assignment Problem

Given a set of items I and a set of knapsacks J with capacities C_j for all $j \in J$. Furthermore, let $w_{ij} \in \mathbb{Z}_{\geq 0}$ be the weight of item $i \in I$ in knapsack $j \in J$. There are two versions of the *generalized assignment problem*. On the one hand, there is the maximization version, where we want to maximize given profits $p_{ij} \in \mathbb{Z}_{\geq 0}$ of packing item $i \in I$ into knapsack $j \in J$, while satisfying the given capacities:

$$\begin{aligned}
\max \quad & \sum_{i\in I}\sum_{j\in J} p_{ij}x_{ij} & & \\
\text{s.t.} \quad & \sum_{j\in J} x_{ij} = 1 & \forall i \in I & \quad (A.19)\\
& \sum_{i\in I} w_{ij} \leq C_j & \forall j \in J & \quad (A.20)\\
& x_{ij} \in \{0,1\} & \forall\ i \in I,\ j \in J. & \quad (A.21)
\end{aligned}$$

On the other hand, in the minimization version we want to minimize the sum over given costs $c_{ij} \in \mathbb{Z}_{\geq 0}$ of packing item $i \in I$ into knapsack $j \in J$:

$$\begin{aligned}
\min \quad & \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j \in J} x_{ij} = 1 && \forall i \in I && \text{(A.22)} \\
& \sum_{i \in I} w_{ij} \leq W_j && \forall j \in J && \text{(A.23)} \\
& x_{ij} \in \{0,1\} && \forall\ i \in I,\ j \in J. && \text{(A.24)}
\end{aligned}$$

In both cases we obtain the extended formulation by independently applying Dantzig-Wolfe reformulation to the constraints (A.20) and (A.23), respectively. Once again, there exists a knapsack pricing problem for each knapsack $j \in J$.
For our tests we used the instances presented in [36], [37], and [38].

A.6 The Resource Allocation Problem

In this section we define the resource allocation problem and follow the explanations in [2]. Suppose we are given an ordered set of periods $T := \{1, \ldots, N\}$ and a set of items I. Each item $i \in I$ is assigned with a profit p_i, a weight w_i, as well as a starting and an ending period. We define for each $t \in T$ the set $I(t) \subseteq I$ as the set of all items which are alive in period t, i.e. all items, where the period t is greater than or equal to the starting period and lower than or equal to the ending period. Additionally, there exists a knapsack with capacity C over all periods. Our goal is to maximize the sum of profits of selected items such that in each period the capacity is not exceeded. This problem can be formulated as follows:

$$\begin{aligned}
\max \quad & \sum_{i \in I} p_i x_i \\
\text{s.t.} \quad & \sum_{i \in I(t)} w_i x_i \leq C && \forall t \in T && \text{(A.25)} \\
& x_i \in \{0,1\} && \forall\ i \in I. && \text{(A.26)}
\end{aligned}$$

This problem has no block diagonal structure, but `GCG` is able to handle this type of problems, too. Unfortunately, the observed resource allocation problem instances were solved to optimality in the root node or we were not able to solve the root node at all by

applying `GCG` and its structure detectors. Therefore, we were not able to test the basis separator with this formulation.

Nevertheless, we are able to reformulate the resource allocation problem such that it has block diagonal structure. We split the periods T into groups of size M, where M is a fixed natural number. We define $G := \left\lceil \frac{N}{M} \right\rceil$ as well as $N(G) := \{(g-1)M + 1, \ldots, gM\} \cap T$ for all $g \in \{1, \ldots, G\}$.

Furthermore, let $G(i)$ be the set of groups where item i is alive and let $I(g)$ be the set of items which are alive in at least one period of group $g \in \{1, \ldots, G\}$. Additionally, let $g(i) := \min\{g \in G(i)\}$ be the first group in which item $i \in I$ is active.

Creating copies x_i^g for all $g \in G(i)$ of each variable x_i and ensuring that these copies take the same value as well as substituting the variable x_i by $x_i^{g(i)}$ in the objective function for all $i \in I$ leads to the following formulation

$$\begin{aligned}
\max \quad & \sum_{i \in I} p_i x_i^{g(i)} \\
\text{s.t.} \quad & x_i^g = x_i^{g(i)} && \forall i \in I,\ g \in G(i) \backslash \{g(i)\} && \text{(A.27)} \\
& \sum_{i \in I(t)} w_i x_i^g \leq C && \forall t \in T(g),\ g \in G && \text{(A.28)} \\
& x_i^g \in \{0, 1\} && \forall\ g \in G(i),\ i \in I. && \text{(A.29)}
\end{aligned}$$

We will use this formulation with either blocks of size $M = 32$ or $M = 64$ as the original formulation. We get the extended formulation by independently reformulation the set of constraints for each $g \in G$. Hence, we obtain a pricing problem for each $g \in G$ with constraints

$$\sum_{i \in I(t)} w_i x_i^g \leq C \quad \forall t \in T(g).$$

For our computational tests we used the instances proposed in [39].

A.7 The Multiple Knapsack Problem

In the *multiple knapsack problem* we are given a set of knapsacks J with capacities C_j for all $j \in J$ as well as a set of items I with weights $w_i \in \mathbb{Z}_{\geq 0}$ and profits $p_i \in \mathbb{Z}_{\geq 0}$ for all $i \in I$. We want to maximize the profit of packing the items into the knapsacks, while

respecting the capacities of the knapsacks:

$$\begin{aligned}
\max \quad & \sum_{i \in I} \sum_{j \in J} p_i x_{ij} & & \\
\text{s.t.} \quad & \sum_{j \in J} x_{ij} \leq 1 & \forall i \in I & \quad \text{(A.30)} \\
& \sum_{i \in I} w_i x_{ij} \leq C_j & \forall j \in J & \quad \text{(A.31)} \\
& x_{ij} \in \{0,1\} & \forall\ i \in I,\ j \in J. & \quad \text{(A.32)}
\end{aligned}$$

Just like in many previously introduced problem classes, we obtain the extended formulation by independently reformulating the knapsack constraint (A.31) which results in a knapsack pricing problem for each knapsack $j \in J$.
For the multiple knapsack problem we used the instances presented in [40].

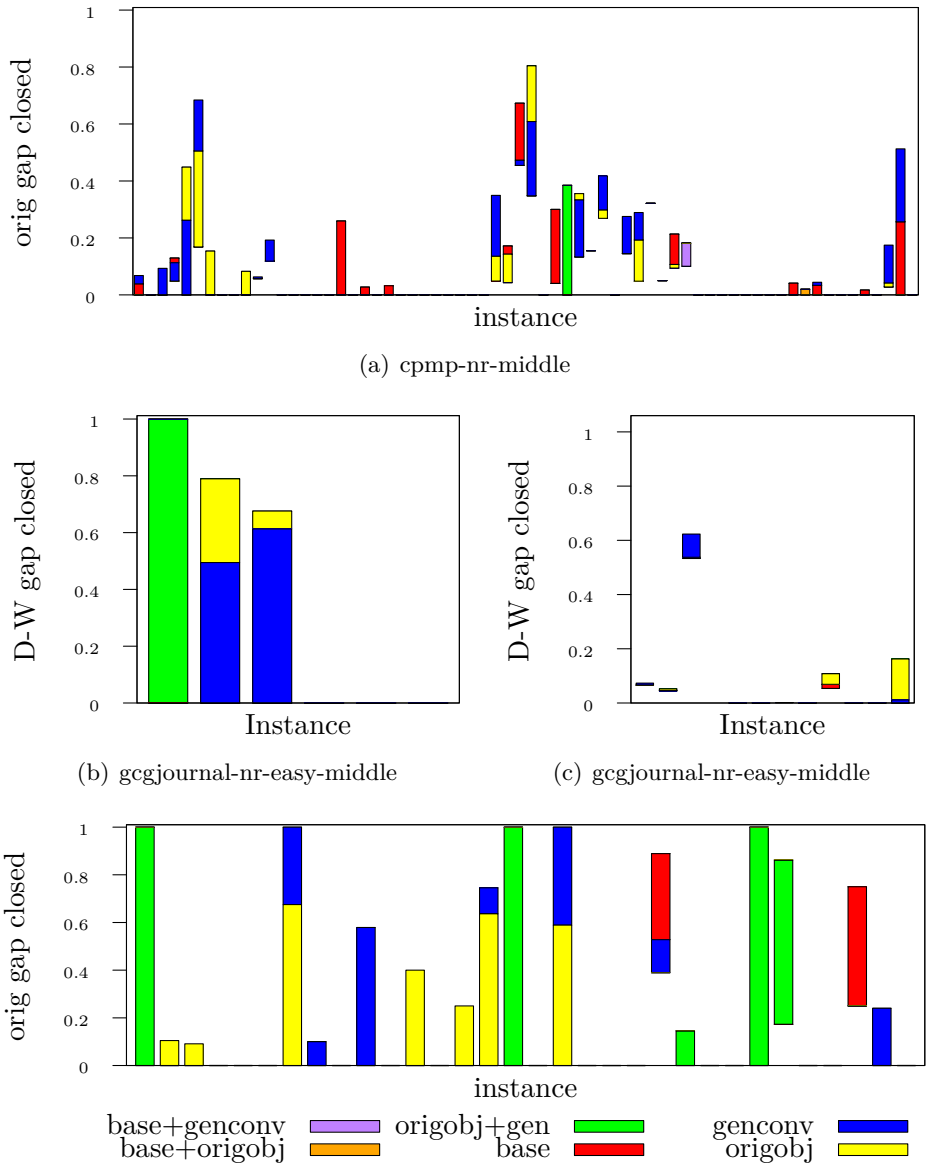
A.8 MIPLIB Instances

The MIBLIB instances which we examine were successfully tested with `GCG` by Bergner et al. [27]. Since these instances do not have a common structure, we obtain the extended formulation by applying `GCG`'s structure detectors.

Appendix B

Figures

In this part of the appendix we show the same kind of figures as in section 5.1 and section 5.4. We presented only a some figures in the main part of the thesis and the rest is shown in the following.



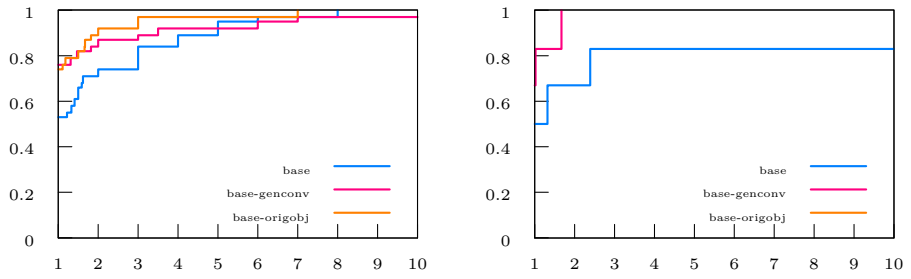
(a) cpmp-nr-middle

(b) gcgjournal-nr-easy-middle

(c) gcgjournal-nr-easy-middle

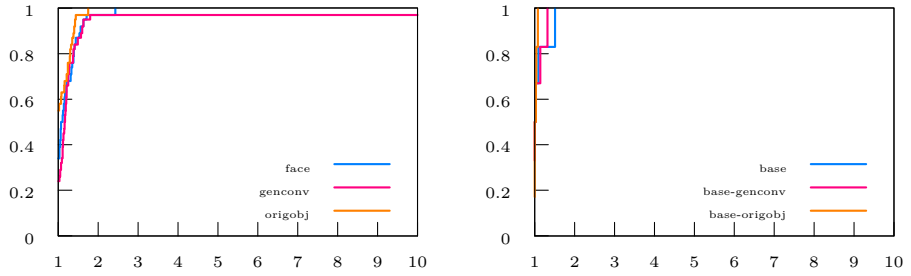
(d) rap64-nr-easy-middle

Figure B.1: Comparison of the Dantzig-Wolfe gap that was closed due to the basis separator with the use of the different objective functions. When we order the settings for a given instance depending on how much of the original gap is closed, the difference between two consecutive settings is marked with the color of the setting that reduces the size of the original gap more on the given instance. If we obtain the same gaps for two settings, we use the color of the combination of both settings.



(a) gap-nr (b) gcgjournal-nr-easy-middle

Figure B.2: Node performance profiles showing in how many instances (y-axis in percent) a given setting needs at most x times less nodes (factor on the x-axis).



(a) gap-nr (b) gcgjournal-nr-easy-middle

Figure B.3: Time performance profiles showing in how many instances (y-axis in percent) a given setting is at most x times slower (factor on the x-axis).

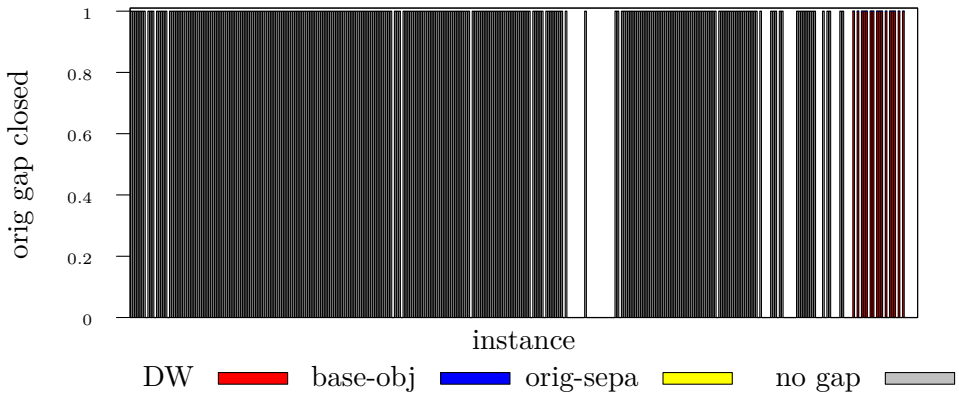
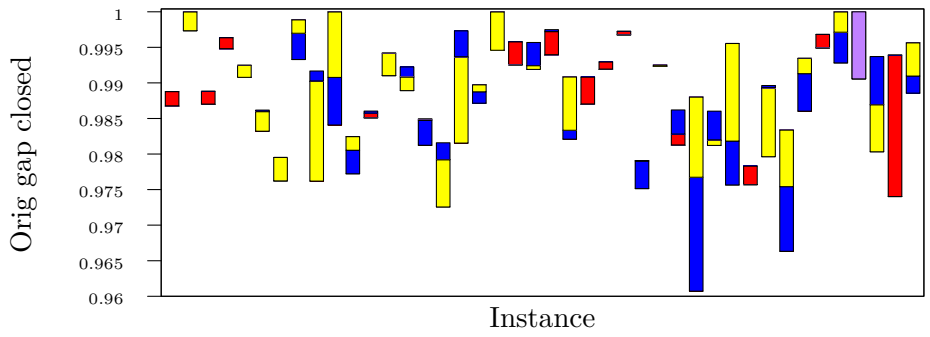
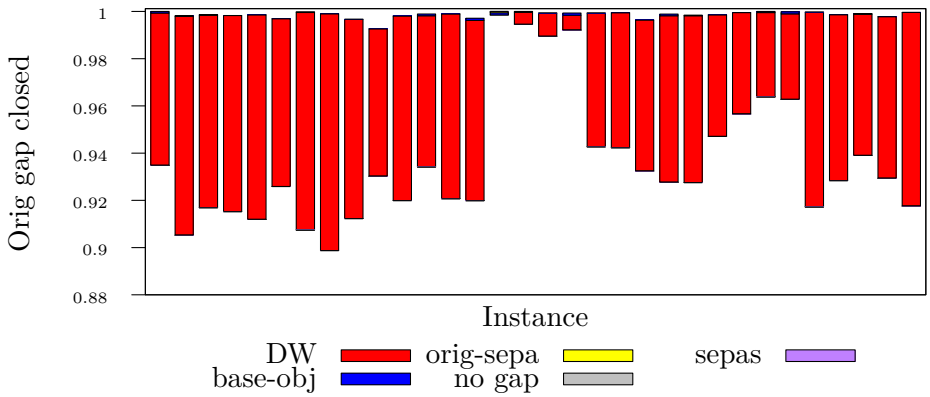


Figure B.4: Comparison of the original gap that was closed due to the Dantzig-Wolfe reformulation, `GCG` with the use of the basis separator, and `SCIP` with the use of separation. The difference between the closed gaps is colored similarly to figure 5.1.

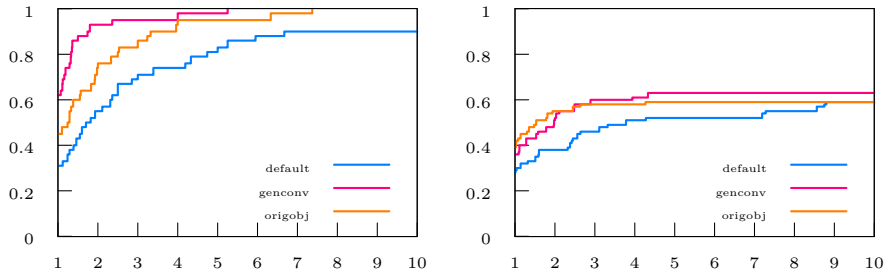


(a) cpmp-nr-easy



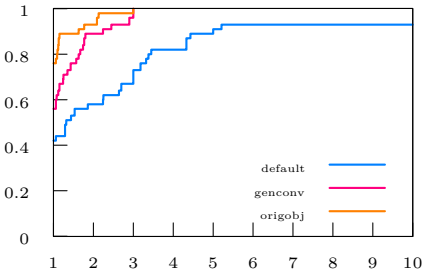
(b) rap64-nr-easy-middle

Figure B.5: Comparison of the original gap that was closed due to the Dantzig-Wolfe reformulation, `GCG` with the use of the basis separator, and `SCIP` with the use of separation. The difference between the closed gaps is colored similarly to figure 5.1 with the exception that we use the color of the Dantzig-Wolfe reformulation if the corresponding dual bound is equal to the dual bound that was computed with the use of the basis separator.



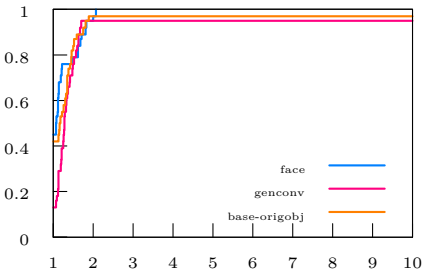
(a) cpmp-nr-easy

(b) cpmp-nr-middle

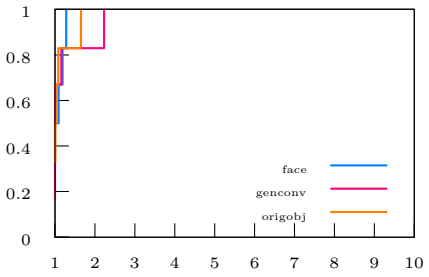


(c) rap32-nr-easy-middle

Figure B.6: Node performance profiles showing in how many instances (y-axis in percent) a given setting needs at most x times less nodes (factor on the x-axis).



(a) gap-nr



(b) gcgjournal-nr-easy-middle

Figure B.7: Time performance profiles showing in how many instances (y-axis in percent) a given setting is at most x times slower (factor on the x-axis).

List of Figures

List of Tables

Bibliography

[1] F. Vanderbeck and M. W. P. Savelsbergh. A generic view of Dantzig-Wolfe decomposition in mixed integer programming. *Operations Research Letters*, 34(3):296–306, 2006.

[2] G. Gamrath. Generic branch-cut-and-price. Master's thesis, Technische Universität Berlin, 2010.

[3] T. M. Range. An integer cutting-plane procedure for the Dantzig-Wolfe decomposition: Theory. Working paper, 2006.

[4] J. Desrosiers and M. E. Lübbecke. Branch-price-and-cut algorithms. In J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010.

[5] J. Desrosiers and M. E. Lübbecke. A primer in column generation. In G. Desaulniers, J. Desrosiers, and M. M. Solomon., editors, *Column Generation*, pages 1–32. Springer US, 2005.

[6] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[7] G. B. Dantzig and M. N. Thapa. *Linear Programming 2: Theory and Extensions*. Springer, 2006.

[8] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[9] A. M. Geoffrion. Lagrangean relaxation for integer programming. In M.L. Balinski, editor, *Approaches to Integer Programming*, volume 2 of *Mathematical Programming Studies*, pages 82–114. Springer Berlin Heidelberg, 1974.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[11] T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[12] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[13] Friedrich Eisenbrand. Note – on the membership problem for the elementary closure of a polyhedron. *Combinatorica*, 19(2):297–300, 1999.

[14] M. Fischetti and A. Lodi. Optimizing over the first Chvátal closure. *Mathematical Programming*, 110(1):3–20, 2007.

[15] A. M. C. A. Koster, M. Kutschka, and A. Zymolka. Algorithms to separate 0, $frac12$ Chvátal-Gomory cuts. Technical Report 07-10, Zuse Institute Berlin, Berlin, Germany, 2007. http://opus.kobv.de/zib/volltexte/2007/953/.

[16] R. E. Bixby and M. J. Saltzman. Recovering an optimal LP basis from an interior point solution. *Operations Research Letters*, 15(4):169–178, 1994.

[17] H. B. Amor, J. Desrosiers, and F. Soumis. Recovering an optimal LP basis from an optimal dual solution. *Operations research letters*, 34(5):569–576, 2006.

[18] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.

[19] G. Desaulniers, J. Desrosiers, and S. Spoorendonk. Cutting planes for branch-and-price algorithms. *Networks*, 58(4):301–310, 2011.

[20] S. Spoorendonk and G. Desaulniers. Clique inequalities applied to the vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 48(1):53–67, 2010.

[21] S. Dash and M. Goycoolea. A heuristic to generate rank-1 GMI cuts. *Mathematical Programming Computation*, 2(3-4):231–257, 2010.

[22] T. K. Ralphs and M. V. Galati. Decomposition and dynamic cut generation in integer linear programming. *Math. Program.*, 106(2):261–285, may 2006.

[23] Q. Louveaux, L. Poirrier, and D. Salvagnin. The strength of multi-row models. Working paper, 2012.

[24] E. Balas and M. Perregaard. Lift-and-project for mixed 0-1 programming: recent progress. *Discrete Applied Mathematics*, 123(1-3):129–154, 2002.

[25] T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):361–372, 2006.

[26] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011.

[27] M. Bergner, A. Caprara, F. Furini, M. E. Lübbecke, E. Malaguti, and E. Traversi. Partial convexification of general MIPs by Dantzig-Wolfe reformulation. In *Integer Programming and Combinatoral Optimization*, pages 39–51. Springer, 2011.

[28] D. an Mey, C. Terboven, P. Kapinos, D. Schmidl, S. Wienke, T. Cramer, and M. Wirtz. *The RWTH HPC-Cluster User's Guide Version 8.2.6.* Rechen- und Kommunikationszentrum der RWTH Aachen, 2013.

[29] P. Schwerin and G. Wäscher. The bin-packing problem: A problem generator and some numerical experiments with ffd packing and mtp. *International Transactions in Operational Research*, 4(5-6):377–389, 1997.

[30] E. Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of heuristics*, 2(1):5–30, 1996.

[31] G. Belov and G. Scheithauer. A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European journal of operational research*, 171(1):85–106, 2006.

[32] G. Wäscher and T. Gau. Heuristics for the integer one-dimensional cutting stock problem: a computational study. *OR Spectrum*, 18(3):131–144, 1996.

[33] P. A. Hartman and L. A. Martin-Vega. An analysis of cutting stock problems generated by cutgen1.

[34] T. Gau and G. Wascher. Cutgen1: A problem generator for the standard one-dimensional cutting stock problem. *European Journal of Operational Research*, 84(3):572–579, August 1995.

[35] J. E. Beasley. OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[36] I. H. Osman. Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. *OR Spectrum*, 17(4):211–225, 1995.

[37] D. G. Cattrysse, M. Salomon, and L. N. Van Wassenhove. A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research*, 72(1):167–174, 1994.

[38] P. C. Chu and J. E. Beasley. A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, 24(1):17–23, 1997.

[39] A. Caprara, F. Furini, and E. Malaguti. Exact algorithms for the temporal knapsack problem. Technical report, 2010.

[40] M. E. Lalami, M. Elkihel, D. El Baz, and V. Boyer. Heuristics for the 0-1 multiple knapsack problem. Laas report 10035, 2008.

[41] M. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2004.

[42] R. Sahraeian and P. Kaveh. Solving capacitated p-median problem by hybrid k-means clustering and fixed neighborhood search algorithm.