Component Bound Branching in a Branch-and-Price Framework

Master Thesis in Computer Science
RWTH Aachen University

Til Mohr

til.mohr@rwth-aachen.de
Student ID: 405959

May 6, 2024

1st Examiner
Prof. Dr. Peter Rossmanith
Chair of Theoretical Computer Science
RWTH Aachen University

2nd Examiner
Prof. Dr. Marco Lübbecke
Chair of Operations Research
RWTH Aachen University

Eidesstattliche Versicherung

______________________________ ______________________________

Name, Vorname Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

____

____

____

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

______________________________ ______________________________

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

______________________________ ______________________________

Ort, Datum Unterschrift

Abstract

This master thesis integrates the component bound branching rule, proposed by Vanderbeck et al. [1, 2], into the branch-price-and-cut solver GCG. This rule, similarly to Vanderbeck's generic branching scheme [3], exclusively operates within the Dantzig-Wolfe reformulated problem, where branching decisions generally have no corresponding actions in the original formulation. The current GCG framework requires modifications for such branching rules, especially within the pricing loop, as seen in Vanderbeck's method implementation. These rules also fail to utilize enhancements like dual value stabilization.

A significant contribution of this thesis is the enhancement of the GCG architecture to facilitate the seamless integration of new branching rules that operate solely on the reformulated problem. This allows these rules to benefit from current and future improvements in the branch-price-and-cut framework, including dual value stabilization, without necessitating alterations to the branching rule itself.

The thesis proposes an interface to manage constraints in the master problem that lack counterparts in the original formulation. These constraints require specific modifications to the pricing problems to ensure their validity in the master. The 'generic mastercut' interface, tightly integrated into the GCG solver, spans the pricing loop, column generation, and dual value stabilization. Enhancements to the existing branching rule interface complement this integration, enabling effective utilization of the generic mastercuts.

Finally, the component bound branching rule will be implemented using this new interface and evaluated on a set of benchmark instances. Its performance will be benchmarked against the existing Vanderbeck branching rule, offering a practical comparison of both approaches.

Contents

Chapter 1

Introduction

Chapter 2

Preliminaries

In this preliminary chapter we will provide a brief rundown of theorems and algorithms on which the techniques described in later chapters, such as Column Generation in Section 3.1, are building upon. Understanding these concepts is essential to understanding the theory later presented. If, however, one is familiar with these, we invite the reader to skip ahead to Chapter 3.

2.1 Polyhedron Representation

Definition 2.1. *Given k points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathbb{R}^n$, any $\boldsymbol{x} = \sum_{i=1}^{k} \alpha_i \boldsymbol{x}_i$ is a* ***conic combination*** *of the $\boldsymbol{x}_i$, iff $\forall i \in \{1, \ldots, k\}.\alpha_i \geq 0$.*

Definition 2.2. *Given k points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathbb{R}^n$, any $\boldsymbol{x} = \sum_{i=1}^{k} \alpha_i \boldsymbol{x}_i$ is a* ***convex combination*** *of the $\boldsymbol{x}_i$, iff $\sum_{i=1}^{k} \alpha_i = 1 \wedge \forall i \in \{1, \ldots, k\}.\alpha_i \geq 0$.*

The set of all convex combinations of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ is therefore defined as:

$$\operatorname{conv}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) \coloneqq \{\sum_{i=1}^{k} \alpha_i \boldsymbol{x}_i \mid \sum_{i=1}^{k} \alpha_i = 1 \wedge \forall i \in \{1, \ldots, k\}.\alpha_i \geq 0\}$$

Corollary 2.1. *The intersection of two convex sets is convex.*

Definition 2.3. *Let $\mathcal{P}$ be a convex set. A point $\boldsymbol{p} \in \mathcal{P}$ is an* ***extreme point*** *of $\mathcal{P}$ if there is no non-trivial convex combination of any two points in $\mathcal{P}$ expressing $\boldsymbol{p}$, i.e.*

$$\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{P}.\forall \alpha \in \mathbb{R}_+ \backslash \{0\}.\boldsymbol{x}_1 \neq \boldsymbol{x}_2 \implies \boldsymbol{p} \neq \alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2$$

Definition 2.4. *Let $\mathcal{P}$ be a convex set. A vector $\boldsymbol{r} \in \mathbb{R}_0^n \backslash \{0\}$ is a* ***ray*** *of $\mathcal{P}$ iff $\forall \boldsymbol{x} \in \mathcal{P}.\forall \beta \in \mathbb{R}_+.\boldsymbol{x} + \beta \boldsymbol{r} \in \mathcal{P}$.*

The span of rays $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_k \in \mathbb{R}_+^n$ we denote as:

$$\operatorname{rayspan}(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_k) \coloneqq \bigcup_{i=1}^{k} \{\omega \boldsymbol{r}_i \mid \omega \in \mathbb{R}_+\}$$

Definition 2.5. *A ray* $\boldsymbol{r}$ *of* $\mathcal{P}$ *is an* ***extreme ray*** *of* $\mathcal{P}$ *if there is no non-trivial conic combination of any two rays in* $\mathcal{P}$ *expressing* $\boldsymbol{r}$*, i.e.*

$$\forall \boldsymbol{r}_1, \boldsymbol{r}_2 \in \mathcal{P}. \forall \alpha_1, \alpha_2, \beta \in \mathbb{R}_+ \backslash \{0\}. \boldsymbol{r}_1 \neq \beta \boldsymbol{r}_2 \implies \boldsymbol{r} \neq \alpha_1 \boldsymbol{r}_1 + \alpha_2 \boldsymbol{r}_2$$

Definition 2.6. *A* ***hyperplane*** $\mathcal{H} \subset \mathbb{R}^n$ *of a n-dimensional space is a subspace of dimension* $n-1$*, and can therefore be described using a vector* $\boldsymbol{f} \in \mathbb{R}^n$ *and a scalar* $f \in \mathbb{R}$ *as* $\mathcal{H} = \{\boldsymbol{x} \mid \boldsymbol{f}^\intercal \boldsymbol{x} = f\}$*.*

Corollary 2.2. *Any hyperplane is a convex set.*

Proof. Let $\mathcal{H} = \{\boldsymbol{x} \mid \boldsymbol{f}^\intercal \boldsymbol{x} = f\}$ be a hyperplane. Let $k \in \mathbb{N}$, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathcal{H}$. For any $\alpha_1, \ldots, \alpha_k \in \mathbb{R}_+$ with $\sum_{i=1}^k$:

$$\begin{aligned} \boldsymbol{f}^\intercal \left(\sum_{i=1}^k \alpha_i \boldsymbol{x}_i \right) &= \sum_{i=1}^k \alpha_i \boldsymbol{f}^\intercal \boldsymbol{x}_i \\ &= \sum_{i=1}^k \alpha_i \cdot f \\ &= f \cdot \sum_{i=1}^k \alpha_i \\ &= f \end{aligned}$$

Therefore, the convex combination $\sum_{i=1}^k \alpha_i \boldsymbol{x}_i$ is in the hyperplane $\mathcal{H}$. □

Definition 2.7. *A* ***halfspace*** *is the set above or below a hyperplane. A halfspace is open if the points on the hyperplane are excluded, otherwise closed.*

Corollary 2.3. *Any halfspace is a convex set.*

Proof. Let $\mathcal{H}^+ = \{\boldsymbol{x} \mid \boldsymbol{f}^\intercal \boldsymbol{x} > f\}$ be an open halfspace (analogous for $\mathcal{H}^- = \{\boldsymbol{x} \mid \boldsymbol{f}^\intercal \boldsymbol{x} < f\}$, and for the closed halfspaces). Let $k \in \mathbb{N}$, $\boldsymbol{x_1}, \ldots, \boldsymbol{x_k} \in \mathcal{H}$. For any $\alpha_1, \ldots, \alpha_k \in \mathbb{R}_+$ with $\sum_{i=1}^k$:

$$\begin{aligned} \boldsymbol{f}^\intercal \left(\sum_{i=1}^k \alpha_i \boldsymbol{x}_i \right) &= \sum_{i=1}^k \alpha_i \boldsymbol{f}^\intercal \boldsymbol{x}_i \\ &> \sum_{i=1}^k \alpha_i \cdot f \\ &= f \cdot \sum_{i=1}^k \alpha_i \\ &= f \end{aligned}$$

Therefore, the convex combination $\sum_{i=1}^k \alpha_i \boldsymbol{x}_i$ is in the halfspace $\mathcal{H}$. □

Definition 2.8. *A **polyhedron** $\mathcal{P} \subseteq \mathbb{R}^n$ is defined by the intersection of a set of closed halfspaces, i.e. $\mathcal{P} \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$, with $\boldsymbol{A} \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m$.*

By Corollaries 2.1 and 2.3, a polyhedron is also a convex set of points.

Definition 2.9. *The **Minkowski sum** of two sets P, Q is defined by:*

$$P \oplus Q \coloneqq \{\boldsymbol{p} + \boldsymbol{q} \mid \boldsymbol{p} \in P \wedge \boldsymbol{q} \in Q\}$$

Theorem 2.1 (Minkowski-Weyl)**.** *For $\mathcal{P} \subseteq \mathbb{R}^n$ the following statements are equivalent:*

1. *$\mathcal{P}$ is a polyhedron, i.e., there exists some finite matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and some vector $\boldsymbol{b} \in \mathbb{R}^m$ such that $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$*

2. *There exist fine vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s \in \mathbb{R}^n$ and finite vectors $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_t \in \mathbb{R}^n_+$, such that $P = \operatorname{conv}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s) \oplus \operatorname{rayspan}(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_t)$*

In simple terms, the Minkowski-Weyl theorem states that any polyhedron can always be defined in two ways: either by its faces, i.e. closed halfspaces, or by its vertices and rays. Because of their unique properties, for such representation of a polyhedron it is sufficient to select its the extreme points and extreme rays. Figure xyz. illustrates this TODO-til

The following theorem builds upon the Minkowski-Weyl theorem to describe a polyhedron, which is represented by its extreme points $\{\boldsymbol{x}_p\}_{p \in P}$ and extreme rays $\{\boldsymbol{x}_r\}_{r \in R}$, using hyperplanes. Here, the sets P, R are used to index the extreme points and extreme rays, respectively.

Theorem 2.2 (Nemhauser-Wolsey)**.** *Consider the polyhedron $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{Q}\boldsymbol{x} \geq \boldsymbol{b}\}$ with full row rank matrix $\boldsymbol{Q} \in \mathbb{R}^{m \times n}$, i.e. $\operatorname{rank}(\boldsymbol{Q}) = m \leq n \wedge \mathcal{P} \neq \emptyset$. An equivalent description of $\mathcal{P}$ using its extreme points $\{\boldsymbol{x}_p\}_{p \in P}$ and extreme rays $\{\boldsymbol{x}_r\}_{r \in R}$ is:*

$$\mathcal{P} = \left\{ \boldsymbol{x} \in \mathbb{R}^n \left| \begin{array}{rlll} \sum\limits_{p \in P} \boldsymbol{x}_p \lambda_p + & \sum\limits_{r \in R} \boldsymbol{x}_r \lambda_r & = \boldsymbol{x} & \\ \sum\limits_{p \in P} \lambda_p & & = 1 & \\ \lambda_p & & \geq 0 & \forall p \in P \\ & \lambda_r & \geq 0 & \forall r \in R \end{array} \right. \right\} \tag{2.1}$$

In the Nemhauser-Wolsey theorem, the conditions of the Minkowski-Weyl theorem are clearly encoded: the second and third lines ensure that the convex set of the extreme points are considered in the first line (Definition 2.2), the last playing a part in the span of extreme rays (Definition 2.4), and the first line being the Minkowski sum of the convex hull of extreme rays and the span of extreme rays.

2.2 Primal Simplex Algorithm

Have the following linear program in standard form:

$$\begin{aligned} \min \quad & \boldsymbol{c}^\intercal \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad [\boldsymbol{\pi}] \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned} \tag{2.2}$$

The primal simplex algorithm finds an optimal solution by moving from one extreme point of the polyhedron to the next, therefore always remaining feasible. A central part of this algorithm is the sufficient optimality condition. For a basic solution $\boldsymbol{X} = [\boldsymbol{x}_\mathcal{B}, \boldsymbol{x}_\mathcal{N}]$ at a given extreme point to be optimal, the reduced costs $\bar{c}_j := c_j - \boldsymbol{\pi}^\intercal \boldsymbol{a}_j$ for $j \in \mathcal{N}$ must be non-negative.

This sufficient optimality condition gives rise to the **pricing problem**, which either verifies the optimality of the current basic solution, and otherwise determines the non-basic variable x_l, $l \in \mathcal{N}$ with the least reduced cost ($\bar{c}_l < 0$) to be swapped into the basis next, according to Dantzig's rule (TODO cite). Formally, this can be written as:

$$l \in \arg\min_{j \in \mathcal{N}} c_j - \boldsymbol{\pi}^\intercal \boldsymbol{a}_j \tag{2.3}$$

or as the linear program:

$$\bar{c}(\boldsymbol{\pi}) = \min_{j \in \mathcal{N}} c_j - \boldsymbol{\pi}^\intercal \boldsymbol{a}_j \tag{2.4}$$

Solving the pricing problem thus plays an integral role in the primal simplex algorithm:

Algorithm 2.1: Primal simplex algorithm with Dantzig's rule

Input: LP in standard form (2.2); Basic and non-basic index-sets $\mathcal{B}, \mathcal{N}$

Output: Optimal Solution $(\boldsymbol{x}, z)$

loop

- $\boldsymbol{\pi}^\intercal \leftarrow \boldsymbol{c}_\mathcal{B}^\intercal \boldsymbol{A}_\mathcal{B}^{-1}$; $\bar{\boldsymbol{b}} \leftarrow \boldsymbol{A}_\mathcal{B}^{-1} \boldsymbol{b}$;
- $\bar{c}_j \leftarrow c_j - \boldsymbol{\pi}^\intercal \boldsymbol{a}_j; \qquad \forall j \in \mathcal{N}$
- $l \leftarrow \arg\min_{j \in \mathcal{N}} \bar{c}_j$; $\bar{c}(\boldsymbol{\pi}) \leftarrow \bar{c}_l$;
- **if** $\bar{c}(\boldsymbol{\pi}) \geq 0$ **then**
  - **return** $\left(\left[\bar{\boldsymbol{b}}, \boldsymbol{0}\right], \boldsymbol{c}_\mathcal{B}^\intercal \boldsymbol{x}_\mathcal{B}\right)$ *by optimality*
- **end**
- $\bar{\boldsymbol{a}}_l \leftarrow \boldsymbol{A}_\mathcal{B}^{-1} \boldsymbol{a}_l$;
- **if** $\bar{\boldsymbol{a}}_l \leq \boldsymbol{0}$ **then**
  - **return None** *by unboundedness*
- **end**
- $s \leftarrow \arg\min_{i \in \{1,\dots,m\}} \frac{\bar{b}_i}{\bar{a}_{il}}$; $x_l \leftarrow \frac{\bar{b}_s}{\bar{a}_{sl}}$; $\mathcal{B} \leftarrow \mathcal{B} \cup \{l\} \subseteq \{s\}$; $\mathcal{N} \leftarrow \mathcal{N} \cup \{s\} \subseteq \{l\}$;

Chapter 3

Column Generation and Branch-and-Price

3.1 Column Generation

Let us consider the following linear program, which we will henceforth call the **master problem** *MP*, where $c_{\boldsymbol{x}} \in \mathbb{R}, \boldsymbol{a}_{\boldsymbol{x}}, \boldsymbol{b} \in \mathbb{R}^m, \forall \boldsymbol{x} \in \mathcal{X}$:

$$\begin{aligned} z_{MP} = \min \quad & \sum_{\boldsymbol{x} \in \mathcal{X}} c_{\boldsymbol{x}} \lambda_{\boldsymbol{x}} \\ \text{s.t.} \quad & \sum_{\boldsymbol{x} \in \mathcal{X}} \boldsymbol{a}_{\boldsymbol{x}} \lambda_{\boldsymbol{x}} \geq \boldsymbol{b} \quad [\boldsymbol{\pi}] \\ & \lambda_{\boldsymbol{x}} \geq \boldsymbol{0} \qquad \forall \boldsymbol{x} \in \mathcal{X} \end{aligned} \tag{3.1}$$

Assume the number of variables is huge, i.e. a lot larger than the number of constraints ($m \ll |\mathcal{X}| < \infty$). Because of this, solving *MP* in a reasonable amount of time, sometimes at all, is infeasible.

We can, however, make use of a crucial property of the primal simplex algorithm: at any given vertex solution, only few variables are in the basis. Most variables are in the non-basis, and therefore have a solution value of 0. Having a solution value of 0 is equivalent to not being in the linear program at all. Therefore, the primal simplex algorithm can also function using a manageable subset of variables $\mathcal{X}' \subseteq \mathcal{X}$, finding a possibly non-optimal, yet still feasible solution for the entire optimization problem *MP*. We denote this master problem restricted to a subset of variables as the **restricted master problem** *RMP*:

$$\begin{aligned} z_{RMP} = \min \quad & \sum_{\boldsymbol{x} \in \mathcal{X}'} c_{\boldsymbol{x}} \lambda_{\boldsymbol{x}} \\ \text{s.t.} \quad & \sum_{\boldsymbol{x} \in \mathcal{X}'} \boldsymbol{a}_{\boldsymbol{x}} \lambda_{\boldsymbol{x}} \geq \boldsymbol{b} \quad [\boldsymbol{\pi}] \\ & \lambda_{\boldsymbol{x}} \geq \boldsymbol{0} \qquad \forall \boldsymbol{x} \in \mathcal{X}' \end{aligned} \tag{3.2}$$

Assuming *MP* is feasible, two important aspects of finding an optimal solution to *MP* are still missing: first, how do we find a subset $\mathcal{X}'$ of the variables, such that *RMP* stays feasible? Without this property of the set of variables, no solution of *RMP* can be found, and therefore none can be found for *MP*, which would contradict the feasibility of *MP*. Secondly, assuming a solution of *RMP* was found, possibly even optimal within *RMP*, how could we build upon this solution to eventually find an optimal solution for *MP*?

In the following we will dive into these two questions in detail (Sections 3.1.1 and 3.1.2), making way for the final column generation algorithm (Section 3.1.3).

3.1.1 Farkas Pricing

Let us assume *MP* is feasible, but our current selection of variables $\mathcal{X}' \subset \mathcal{X}$ results in the *RMP* being infeasible. The task is now to find additional variables such that a new set $\mathcal{X}''$ with $\mathcal{X}' \subset \mathcal{X}'' \subseteq \mathcal{X}$ makes the *RMP* feasible. For this, consider Farkas' lemma:

Theorem 3.1 (Farkas' lemma)**.** *Given $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, then exactly one of the following statements holds:*

1. $\exists \boldsymbol{x} \in \mathbb{R}^n_+ .\ \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}$
2. $\exists \boldsymbol{\pi} \in \mathbb{R}^n_+ .\ \boldsymbol{\pi}^\intercal \boldsymbol{A} \leq \boldsymbol{0} \wedge \boldsymbol{\pi}^\intercal \boldsymbol{b} > 0$

Given that the *MP* is feasible, the following must hold for the *MP* with $\boldsymbol{A} = \boldsymbol{A}_{|\mathcal{X}}$:

$$\begin{aligned}
&\exists \boldsymbol{x} \in \mathbb{R}^n_+ .\ \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} && \wedge \neg\exists \boldsymbol{\pi} \in \mathbb{R}^n_+ .\ \boldsymbol{\pi}^\intercal \boldsymbol{A} \leq \boldsymbol{0} \wedge \boldsymbol{\pi}^\intercal \boldsymbol{b} > 0 \\
\Leftrightarrow &\exists \boldsymbol{x} \in \mathbb{R}^n_+ .\ \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} && \wedge \forall \boldsymbol{\pi} \in \mathbb{R}^n_+ .\ \neg\left(\boldsymbol{\pi}^\intercal \boldsymbol{A} \leq \boldsymbol{0} \wedge \boldsymbol{\pi}^\intercal \boldsymbol{b} > 0\right) \\
\Leftrightarrow &\exists \boldsymbol{x} \in \mathbb{R}^n_+ .\ \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b} && \wedge \forall \boldsymbol{\pi} \in \mathbb{R}^n_+ .\ \boldsymbol{\pi}^\intercal \boldsymbol{A} > \boldsymbol{0} \vee \boldsymbol{\pi}^\intercal \boldsymbol{b} \leq 0 \\
\Rightarrow &\exists \boldsymbol{x} \in \mathbb{R}^n_+ .\ \exists \boldsymbol{\pi} \in \mathbb{R}^n_+ .\ \boldsymbol{\pi}^\intercal \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{\pi}^\intercal \boldsymbol{b}
\end{aligned} \tag{3.3}$$

Furthermore, from the infeasibility of *RMP* we can also derive the following statement:

$$\begin{aligned}
&\left(\forall \boldsymbol{\pi} \in \mathbb{R}^n_+ .\ \boldsymbol{\pi}^\intercal \boldsymbol{A} > \boldsymbol{0} \vee \boldsymbol{\pi}^\intercal \boldsymbol{b} \leq 0\right) \wedge \left(\exists \boldsymbol{\pi} \in \mathbb{R}^n_+ .\ \boldsymbol{\pi}^\intercal \boldsymbol{A}_{|\mathcal{X}'} \leq \boldsymbol{0} \wedge \boldsymbol{\pi}^\intercal \boldsymbol{b} > 0\right) \\
\Rightarrow &\left(\neg\forall \boldsymbol{\pi} \in \mathbb{R}^n_+ .\ \boldsymbol{\pi}^\intercal \boldsymbol{b} \leq 0\right) \wedge \left(\exists \boldsymbol{\pi} \in \mathbb{R}^n_+ .\ \boldsymbol{\pi}^\intercal \boldsymbol{A} > \boldsymbol{0}\right)
\end{aligned} \tag{3.4}$$

Therefore, there is some variable $\boldsymbol{x} \in \mathcal{X}\backslash\mathcal{X}'$ such that its column $\boldsymbol{a_x} \coloneqq \boldsymbol{A}_{|\{\boldsymbol{x}\}}$ is $\boldsymbol{\pi}^\intercal \boldsymbol{a_x} > 0$ for some $\boldsymbol{\pi} \in \mathbb{R}^n_+$. If none existed, *MP* would not be feasible.

This process of finding corresponding columns $\boldsymbol{a_x}$ to add to the *RMP* can be formalized as a pricing problem with cost coefficients $c_{\boldsymbol{x}} = 0$ (see Equation (2.4)):

$$\mathrm{F}(\boldsymbol{\pi}) = \min_{x\in\mathcal{X}} \ -\boldsymbol{\pi}^\intercal \boldsymbol{a}_x \tag{3.5}$$

We can add all solutions $\boldsymbol{x}$ with a solution value of $\mathrm{F}(\boldsymbol{\pi}) < 0$ to $\mathcal{X}'' \coloneqq \mathcal{X}' \cup \{\boldsymbol{x}_i\}$, adding the corresponding column $\begin{bmatrix} 0 \\ \boldsymbol{a}_{\boldsymbol{x}} \end{bmatrix}$ to the problem, thus turning any infeasible *RMP* feasible.

3.1.2 Reduced Cost Pricing

3.1.3 Column Generation Algorithm

3.2 Dantzig-Wolfe Reformulation

3.3 Dantzig-Wolfe Reformulation for Mixed Integer Programs

3.4 Branch-and-Price

3.5 Branch-Price-and-Cut

3.6 Dual Value Stabilization

Chapter 4

SCIP Optimization Suite

4.1 SCIP

4.2 GCG

Chapter 5

Component Bound Branching

5.1 Overview of the branching scheme

5.2 Separation Procedure

5.3 Parameterizations

Which block to separate in? Which components to branch on?

Chapter 6

Master Constraints without corresponding Original Problem Constraints

6.1 Definition of the Generic Mastercuts

6.2 Application of the Generic Mastercuts

6.3 Dual Value Stabilization for Generic Mastercuts

6.4 Mastervariable Synchronization across the entire B&B-Tree

6.4.1 Problem Statement

6.4.2 Current Approach used by the Implementation of Vanderbeck's Generic Branching

6.4.3 History Tracking Approach

6.4.4 History Tracking using Unrolled Linked Lists Approach

Chapter 7

Implementation

7.1 Generic Mastercuts

7.2 Mastervariable Synchronization

7.3 Component Bound Branching

Chapter 8

Evaluation

Chapter 9

Conclusion

Bibliography

[1] François Vanderbeck and Laurence A Wolsey. *Reformulation and decomposition of integer programs*. Springer, 2010.

[2] François Vanderbeck and Laurence A Wolsey. "An exact algorithm for IP column generation". In: *Operations research letters* 19.4 (1996), pp. 151–159.

[3] François Vanderbeck. "Branching in branch-and-price: a generic scheme". In: *Mathematical Programming* 130 (2011), pp. 249–294.