Otto-von-Guericke-Universität Magdeburg
Fakultät für Mathematik
Institut für Mathematische Optimierung

Diplomarbeit

Erzeugende Mengen für gemischt-ganzzahlige Programme

Matthias Köppe

30. August 1999

Betreuer: Prof. Dr. Robert Weismantel
Gutachter: Priv.-Doz. Dr. Martin Henk

E-Mail-Adresse des Autors: mkoeppe@mail.math.uni-magdeburg.de

Gesetzt aus der *Times* und der *Helvetica* auf einer GNU/Linux-Workstation unter Verwendung des TEX-Systems von Donald E. Knuth. Der mathematische Satz verwendet darüber hinaus die *Formal Script* von Ralph A. Smith sowie Symbole aus der *Adobe Symbol*, aus der *Computer Modern* von Donald E. Knuth und aus den *AMS Fonts* der American Mathematical Society. Technische Informationen zu dieser Arbeit sind auf der Internetseite http://www.math.uni-magdeburg.de/~mkoeppe/art/da-tech.html zu finden. Das verwendete Papier ist aus chlorfrei gebleichten Rohstoffen hergestellt.

Zusammenfassung

Die vorliegende Diplomarbeit beschäftigt sich mit einem primalen Verfahren zur Lösung von ganzzahligen und gemischt-ganzzahligen Optimierungsproblemen. Dieses basiert auf der wiederholten Lösung des folgenden *Augmentierungsproblems*:

> Gegeben ein zulässiger Punkt **x**, finde einen verbessernden Vektor **t** oder versichere, daß ein solcher nicht existiert.

Das Verfahren besitzt eine komplexitätstheoretische Begründung: Nach einem Satz von Schulz–Weismantel–Ziegler (1995) sind Augmentieren und Optimieren für den Fall von 0-1-Programmen polynomiell äquivalent; für allgemeine ganzzahlige Programme läßt sich das Optimierungsproblem auf eine stärkere Fassung des Augmentierungsproblems polynomiell reduzieren.

Für Familien von ganzzahligen Optimierungsproblemen sind in der Literatur verschiedene Klassen von *universellen Testmengen* untersucht worden; dies sind endliche Mengen von Vektoren, die für jeden nicht-optimalen Punkt eine verbessernde Richtung enthalten. Ist eine universelle Testmenge bekannt, so kann das Augmentierungsproblem gelöst werden. *Minimale erzeugende Mengen* sind eine spezielle Klasse von universellen Testmengen; sie erlauben die Lösung von stärkeren Augmentierungsproblemen.

Die Berechnung einer minimalen erzeugenden Menge ist ein $\mathcal{NP}$-schweres Problem. *General-purpose*-Algorithmen für deren Berechnung sind daher nur für sehr geringe Problemgrößen einsetzbar. In der Dissertation von Urbaniak (1998) wurde ein spezielles kombinatorisches Verfahren vorgeschlagen, welches für die Berechnung der minimalen erzeugenden Menge einer Familie von Rucksackproblemen besser geeignet ist. In der vorliegenden Arbeit werden Verfeinerungen und Implementierungstechniken für dieses Verfahren diskutiert, welche die Behandlung größerer Probleme erlauben. Weiterhin wird eine Technik entwickelt, die es ermöglicht, diesen Algorithmus auf eine erweiterte Klasse von Problemen anzuwenden.

Für gemischt-ganzzahlige Programme wird in der vorliegenden Arbeit ein primaler Zugang entwickelt. Nach dem Vorbild der erzeugenden Mengen für ganzzahlige Programme wird eine Menge eingeführt, welche eine endliche Beschreibung besitzt und eine endliche, verträgliche Zerlegung von Differenzvektoren zulässiger Punkte gestattet. Ist die Menge berechnet, so lassen sich verschiedene Augmentierungsprobleme lösen; dazu wird ein Algorithmus angegeben, in dessen Verlauf eine Folge von linearen Programmen gelöst wird.

Inhaltsverzeichnis

1 Grundlagen

1.1 Reduktionsbegriffe

Definition 1.1. Eine Zahl $v \in \mathbf{R}$ *reduziere* eine Zahl $w \in \mathbf{R}$, in Zeichen $v \preceq w$, wenn $0 \le v \le w$ oder $w \le v \le 0$ gelten. Ein Tupel $\mathbf{v} \in \mathbf{R}^n$ *reduziere* ein Tupel $\mathbf{w} \in \mathbf{R}^n$, in Zeichen $\mathbf{v} \preceq \mathbf{w}$, wenn jede Komponente von $\mathbf{v}$ die entsprechende Komponente von $\mathbf{w}$ reduziert. Ein Tupel $\mathbf{v} \in \mathbf{R}^n$ *reduziere* ein Tupel $\mathbf{w} \in \mathbf{R}^n$ *bezüglich der Matrix* $\mathbf{A} \in \mathbf{R}^{m \times n}$, in Zeichen $\mathbf{v} \preceq_{\mathbf{A}} \mathbf{w}$, wenn $(\mathbf{v}, \mathbf{A}\mathbf{v}) \preceq (\mathbf{w}, \mathbf{A}\mathbf{w})$ gilt.

Ist $\mathbf{a} \in \mathbf{R}^n$, so schreiben wir vereinfachend $\mathbf{v} \preceq_{\mathbf{a}} \mathbf{w}$ für $\mathbf{v} \preceq_{\mathbf{a}^\top} \mathbf{w}$ und sagen, $\mathbf{v}$ *reduziere* $\mathbf{w}$ *bezüglich* $\mathbf{a}$.

Definition 1.2. Unter den *Orthanten* des $\mathbf{R}^n$ verstehen wir die abgeschlossenen Mengen

$$\mathscr{O}_\varepsilon := \left\{ \mathbf{x} \in \mathbf{R}^n : \varepsilon_i x_i \ge 0 \text{ für } i \in \{1, \ldots, n\} \right\}$$

für $\varepsilon \in \{\pm 1\}^n$. Wir sagen, zwei Punkte $\mathbf{x}$, $\mathbf{y} \in \mathbf{R}^n$ *liegen im gleichen Orthanten*, wenn es ein $\varepsilon \in \{\pm 1\}^n$ mit $\mathbf{x}$, $\mathbf{y} \in \mathscr{O}_\varepsilon$ gibt; anderenfalls sagen wir, $\mathbf{x}$, $\mathbf{y} \in \mathbf{R}^n$ *liegen in verschiedenen Orthanten*.

Bemerkung 1.3.

- **a.** Es gilt $\mathbf{v} \preceq \mathbf{w}$ genau dann, wenn $\mathbf{v}$ und $\mathbf{w}$ im gleichen Orthanten liegen und $\mathbf{v}$ komponentenweise betragsmäßig kleiner als $\mathbf{w}$ ist.
- **b.** Es gilt $\mathbf{v} \preceq \mathbf{w}$ genau dann, wenn $\mathbf{v}^+ \le \mathbf{w}^+$ und $\mathbf{v}^- \le \mathbf{w}^-$ gelten.
- **c.** Derartige Aussagen über Reduktionsrelationen *bezüglich einer Matrix* $\mathbf{A}$ erhält man durch den Übergang zu der Einbettung der Tupel in den Graphenraum bezüglich $\mathbf{A}$.

Definition 1.4. Ein Tupel $\mathbf{w} \in \mathbf{Z}^n$ heißt *irreduzibel bezüglich der Matrix* $\mathbf{A} \in \mathbf{R}^{m \times n}$, wenn $\mathbf{w} \ne \mathbf{0}$ ist und es kein Tupel $\mathbf{v} \in \mathbf{Z}^n$ mit $\mathbf{v} \ne \mathbf{0}$, $\mathbf{w}$ und $\mathbf{v} \preceq_{\mathbf{A}} \mathbf{w}$ gibt. Anderenfalls heißt $\mathbf{w}$ *reduzibel bezüglich* $\mathbf{A}$.

Bemerkung 1.5. Für jedes Tupel $\mathbf{w}$ gibt es eine Zahl $k \in \mathbf{N}_0$ und Tupel $\mathbf{v}^1, \ldots, \mathbf{v}^k \preceq_{\mathbf{A}} \mathbf{w}$, welche irreduzibel bezüglich $\mathbf{A}$ sind, so daß $\mathbf{w} = \mathbf{v}^1 + \cdots + \mathbf{v}^k$ gilt.

Definition 1.6. Sei $\mathscr{C} \subset \mathbf{R}^n$ ein spitzer Kegel. Ein Vektor $\mathbf{w} \in \mathbf{Z}^n \cap \mathscr{C}$ heißt *irreduzibel bezüglich* $\mathscr{C}$, wenn $\mathbf{w} \ne \mathbf{0}$ ist und es keine Vektoren $\mathbf{u}$, $\mathbf{v} \in \mathbf{Z}^n \cap \mathscr{C}$ mit $\mathbf{u}$, $\mathbf{v} \ne \mathbf{0}$ und $\mathbf{w} = \mathbf{u} + \mathbf{v}$ gibt. Anderenfalls heißt $\mathbf{w}$ *reduzibel bezüglich* $\mathscr{C}$.

Bemerkung 1.7. Seien $\mathbf{A} \in \mathbf{R}^{m\times n}$ eine Matrix, $\mathbf{w} \in \mathbf{Z}^n \cap \ker \mathbf{A}$ und $\varepsilon \in \{\pm 1\}^n$ so gewählt, daß $\mathbf{w} \in \mathscr{O}_\varepsilon$ gilt. Genau dann ist $\mathbf{w}$ irreduzibel bezüglich $\mathbf{A}$, wenn $\mathbf{w}$ irreduzibel bezüglich des spitzen Kegels $\mathscr{O}_\varepsilon \cap \ker \mathbf{A}$ ist.

Die irreduziblen Vektoren können mit einem einfachen kombinatorischen Algorithmus berechnet werden.

Algorithmus 1.8 (Hilbertbasis-Algorithmus). Sei $\hat{\mathbf{A}} \in \mathbf{Q}^{\hat{m}\times n}$ eine beliebige Matrix; sei $\mathbf{A} \in \mathbf{Q}^{m\times n}$ eine Matrix, die von m Zeilen von $\hat{\mathbf{A}}$ gebildet wird, $0 \le m < \hat{m}$.

input G; (∗ Menge der bzgl. $\mathbf{A}$ irreduziblen Vektoren aus $\mathbf{Z}^n \cap \ker \mathbf{A}$ ∗)
output $\hat{G}$; (∗ Menge der bzgl. $\hat{\mathbf{A}}$ irreduziblen Vektoren aus $\mathbf{Z}^n \cap \ker \hat{\mathbf{A}}$ ∗)
$G_{\text{old}} \leftarrow \varnothing$;
while $G_{\text{old}} \neq G$ **do**
    $G_{\text{old}} \leftarrow G$;
    for all $\mathbf{v} \in G_{\text{old}}$ **do**
        for all $\mathbf{w} \in G_{\text{old}}$ mit $(\mathbf{w}, \mathbf{A}\mathbf{w})$ im gleichen Orthanten wie $(\mathbf{v}, \mathbf{A}\mathbf{v})$ **do**
            if $\hat{\mathbf{A}}\mathbf{v}$ und $\hat{\mathbf{A}}\mathbf{w}$ liegen in verschiedenen Orthanten **then**
                $\mathbf{z} \leftarrow \mathbf{v} + \mathbf{w}$;
                if es gibt kein $\tilde{\mathbf{z}} \in G$ mit $\tilde{\mathbf{z}} \preceq_{\hat{\mathbf{A}}} \mathbf{z}$ **then**
                    $G \leftarrow G \cup \{\mathbf{z}\}$
                fi
            fi
        od
    od
od;
$\hat{G} \leftarrow \varnothing$;
for all $\mathbf{z} \in G$ **do**
    if $\hat{\mathbf{A}}\mathbf{z} = \mathbf{0}$, und es gibt kein $\tilde{\mathbf{z}} \in G$ mit $\tilde{\mathbf{z}} \preceq_{\hat{\mathbf{A}}} \mathbf{z}$ und $\tilde{\mathbf{z}} \neq \mathbf{z}$ **then**
        $\hat{G} \leftarrow \hat{G} \cup \{\mathbf{z}\}$
    fi
od.

Bemerkung 1.9. **a.** Für $m = 0$, also $\mathbf{A} = \mathbf{0} \in \mathbf{Q}^{0\times n}$, sind die irreduziblen Vektoren aus $\mathbf{Z}^n \cap \ker \mathbf{A}$ genau die $\pm \mathbf{e}_i$ für $i \in \{1, \dots, n\}$.

b. Durch Anwendung von Algorithmus 1.8, ausgehend von dieser Menge, kann die Menge der bezüglich $\hat{\mathbf{A}}$ irreduziblen Vektoren aus $\mathbf{Z}^n \cap \ker \hat{\mathbf{A}}$ berechnet werden.

c. Es ist auch möglich, Algorithmus 1.8 als „Zeilenkompositionsverfahren“ [Urb98] einzusetzen. In diesem Fall fügt man, ausgehend von der leeren Matrix $\mathbf{0} \in \mathbf{Q}^{0\times n}$, Zeilen von $\hat{\mathbf{A}}$ sukzessive hinzu und wendet in jedem Schritt den Algorithmus an.

d. Die Laufzeit einer Implementierung von Algorithmus 1.8 sollte ausgabesensitiv gemessen werden, da $\#\hat{G}$ „exponentiell groß“ werden kann.

Satz 1.10. *Algorithmus 1.8 berechnet die Menge der bezüglich $\hat{\mathbf{A}}$ irreduziblen Vektoren aus $\mathbf{Z}^n \cap \ker \hat{\mathbf{A}}$. Insbesondere ist diese Menge endlich.*

Siehe z. B. [Urb98, HMW]. □

Bemerkung 1.11. **a.** Im Beweis von Satz 1.10 wird gewöhnlich die folgende Eigenschaft gezeigt: Ist $\mathbf{t} \in \mathbf{Z}^n \cap \ker \hat{\mathbf{A}}$ irreduzibel bezüglich $\hat{\mathbf{A}}$, so gibt es einen *binären Additions-Wurzelbaum* für $\mathbf{t}$, dessen Blätter mit bezüglich $\mathbf{A}$ irreduziblen Vektoren $\mathbf{z}^i \in \mathbf{Z}^n \cap \ker \mathbf{A}$ mit $\mathbf{z}^i \preceq \mathbf{t}$ bewertet sind und für den die Partialsumme in jedem inneren Knoten ein bezüglich $\hat{\mathbf{A}}$ irreduzibler Vektor ist. Algorithmus 1.8 berechnet per Breitensuche alle möglichen Partialsummen und somit auch alle bezüglich $\hat{\mathbf{A}}$ irreduziblen Vektoren.

b. Im allgemeinen gibt es jedoch *keine Folge* von bezüglich $\mathbf{A}$ irreduziblen Vektoren $\mathbf{z}^1, \dots, \mathbf{z}^l \in \mathbf{Z}^n \cap \ker \mathbf{A}$ mit $\mathbf{z}^1 + \dots + \mathbf{z}^l = \mathbf{t}$ und $\mathbf{z}^1, \dots, \mathbf{z}^l \preceq \mathbf{t}$, deren sämtliche Partialsummen, $\sum_{i=1}^{j} \mathbf{z}^i$ für $j < l$, irreduzibel bezüglich $\hat{\mathbf{A}}$ sind.

Beispiel. Sei $\mathbf{A} = \mathbf{0} \in \mathbf{Q}^{0\times 3}$ und $\hat{\mathbf{A}} = (1,2,3) \in \mathbf{Q}^{1\times 3}$. Die bezüglich $\mathbf{A}$ irreduziblen Vektoren aus $\mathbf{Z}^3 \cap \ker \mathbf{A}$ sind $\pm\mathbf{e}_i$ für $i \in \{1,2,3\}$. Der Vektor $\mathbf{t} := 2\mathbf{e}_3 - 3\mathbf{e}_2 \in \mathbf{Z}^3 \cap \ker \hat{\mathbf{A}}$ ist irreduzibel bezüglich $\hat{\mathbf{A}}$. Sei $\mathbf{z}^1, \dots, \mathbf{z}^l$ eine Folge wie in Bemerkung 1.11 b. Dann ist $l = 5$, und unter den $\mathbf{z}^i$ sind zwei Vektoren $\mathbf{e}_3$ und drei Vektoren $-\mathbf{e}_2$.

Es gilt $\{\mathbf{z}^1, \mathbf{z}^2\} = \{\mathbf{e}_3, -\mathbf{e}_2\}$, denn sonst wäre $\mathbf{z}^1 \preceq_{\hat{\mathbf{A}}} \mathbf{z}^1 + \mathbf{z}^2 = 2\mathbf{z}^1$. Weiterhin gilt $\mathbf{z}^3 = -\mathbf{e}_2$, da anderenfalls $\mathbf{e}_3 \preceq_{\hat{\mathbf{A}}} \mathbf{z}^1 + \mathbf{z}^2 + \mathbf{z}^3 = 2\mathbf{e}_3 - \mathbf{e}_2$ wäre. Gilt nun $\mathbf{z}^4 = \mathbf{e}_3$, so ist $\mathbf{e}_3 - \mathbf{e}_2 \preceq_{\hat{\mathbf{A}}} \mathbf{z}^1 + \dots + \mathbf{z}^4 = 2\mathbf{e}_3 - 2\mathbf{e}_2$; ist aber $\mathbf{z}^4 = -\mathbf{e}_2$, so ist $-\mathbf{e}_2 \preceq_{\hat{\mathbf{A}}} \mathbf{z}^1 + \dots + \mathbf{z}^4 = \mathbf{e}_3 - 3\mathbf{e}_2$.

Damit ist gezeigt, daß es in diesem Beispiel keine Folge mit den gewünschten Eigenschaften gibt. Hingegen gehört zu einer Darstellung von $\mathbf{t}$ als vollständig geklammerte Summe,

$$\mathbf{t} = \big((\mathbf{e}_3 - \mathbf{e}_2) - \mathbf{e}_2\big) + (\mathbf{e}_3 - \mathbf{e}_2),$$

ein binärer Additions-Wurzelbaum gemäß Bemerkung 1.11 a. □

1.2 Testmengen und erzeugende Mengen für ganzzahlige Programme

Definition 1.12. Sei ein zulässiger Bereich $P := \big\{\, \mathbf{x} \in \mathbf{Z}^n : \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \,\big\}$ gegeben. Wir betrachten die folgenden Probleme:

a. *Optimierungsproblem* (OPT). Gegeben $\mathbf{c}^\top \in \mathbf{Z}^{(n)}$ und ein zulässiger Punkt $\mathbf{x}$, finde die optimale Lösung $\mathbf{x}^*$ des ganzzahligen Programms $\max \big\{\, \mathbf{c}^\top \mathbf{x} : \mathbf{x} \in P \,\big\}$.

b. *Augmentierungsproblem* (AUG). Gegeben $\mathbf{c}^\top \in \mathbf{Z}^{(n)}$ und ein zulässiger Punkt $\mathbf{x}$, finde einen Vektor $\mathbf{t}$, so daß $\mathbf{x}+\mathbf{t}$ zulässig ist und $\mathbf{c}^\top\mathbf{t} > 0$ gilt, oder versichere, daß ein solcher nicht existiert.

c. *Gerichtetes Augmentierungsproblem* (DIR-AUG). Gegeben $\mathbf{c}^\top$, $\bar{\mathbf{c}}^\top \in \mathbf{Z}^{(n)}$ und ein zulässiger Punkt $\mathbf{x} \in \mathbf{Z}^n$, finde einen Vektor $\mathbf{t} \in \mathbf{Z}^n$, so daß $\mathbf{x}+\mathbf{t}$ zulässig ist und $\mathbf{c}^\top\mathbf{t}^+ - \bar{\mathbf{c}}^\top\mathbf{t}^- > 0$ gilt, oder versichere, daß ein solcher nicht existiert.

Nach einem Satz von Schulz, Weismantel und Ziegler [SWZ95] sind (AUG) und (OPT) für *0-1-Programme*, d. h. im Fall $P \subseteq \{0,1\}^n$, polynomiell äquivalente[1] Probleme. Für allgemeine ganzzahlige Programme ist (AUG) nicht stark genug; jedoch läßt sich (OPT) polynomiell auf (DIR-AUG) reduzieren [HMW]. – Diese Ergebnisse rechtfertigen aus komplexitätstheoretischer Sicht das Studium von Verfahren zur Lösung von Augmentierungsproblemen.

Definition 1.13. Sei $\mathscr{F}$ eine Familie von ganzzahligen Programmen der Gestalt

$$\text{IP}: \ \max\left\{ \mathbf{c}^\top\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{l} \le \mathbf{x} \le \mathbf{u},\ \mathbf{x} \in \mathbf{Z}^n \right\} \tag{1.1}$$

mit festem $n \in \mathbf{N}$.

a. Eine endliche Menge $G \subset \mathbf{Z}^n$ heißt eine *erzeugende Menge* für $\mathscr{F}$, wenn für jedes Programm IP $\in \mathscr{F}$ und je zwei für IP zulässige Punkte $\mathbf{x}, \mathbf{y} \in \mathbf{Z}^n$ gilt:

> Es gibt eine Zahl $l \in \mathbf{N}_0$ und Vektoren $\mathbf{t}^1,\dots,\mathbf{t}^l \in G$ mit $\mathbf{x}-\mathbf{y} = \sum_{i=1}^l \mathbf{t}^i$ und $\mathbf{t}^1,\dots,\mathbf{t}^l \preceq_{\mathbf{A}} \mathbf{x}-\mathbf{y}$.

b. Eine endliche Menge $T \subset \mathbf{Z}^n$ heißt eine *universelle Testmenge* für $\mathscr{F}$, wenn für jedes Programm IP $\in \mathscr{F}$, welches eine optimale Lösung besitzt, und jeden für IP zulässigen, bezüglich $\mathbf{c}^\top$ nicht-optimalen Punkt $\mathbf{x}$ ein Vektor $\mathbf{t} \in T$ mit den folgenden Eigenschaften existiert:

i. Der Punkt $\mathbf{x}+\mathbf{t}$ ist zulässig für IP.

ii. Es gilt $\mathbf{c}^\top\mathbf{t} > 0$, oder es gelten $\mathbf{c}^\top\mathbf{t} = 0$ und $\mathbf{t} \succ_{\text{lex}} \mathbf{0}$.

Hierbei ist $\succeq_{\text{lex}}$ die lexikographische Ordnung auf $\mathbf{Z}^n$.

Ist eine universelle Testmenge bekannt, so läßt sich das Problem (AUG) lösen: Ist ein zulässiger Punkt $\mathbf{x}$ gegeben, so findet man für ihn entweder eine verbessernde Richtung $\mathbf{t}$ in der Testmenge, oder $\mathbf{x}$ ist bereits optimal; im letzteren Fall bildet die Testmenge ein Zertifikat für die Optimalität von $\mathbf{x}$.

[1] Ein Problem Π heißt *polynomiell reduzibel* durch ein Problem Π', wenn Π auf einer Turingmaschine mit einem Orakel für Π' in polynomieller Zeit gelöst werden kann [GLS93]. Sind sowohl Π polynomiell reduzibel durch Π' als auch Π' polynomiell reduzibel durch Π, so heißen Π und Π' *polynomiell äquivalent*.

Definition 1.14. Das Symbol $\mathrm{IP}_{\mathbf{A}}$ bezeichne die Familie der ganzzahligen Programme der Gestalt (1.1) mit fester Matrix $\mathbf{A}$. Die *Gravermenge* für $\mathrm{IP}_{\mathbf{A}}$ ist die Menge aller bezüglich $\mathbf{A}$ irreduziblen Vektoren aus $\mathbf{Z}^n \cap \ker \mathbf{A}$.

Satz 1.15. *Die Gravermenge H für $\mathrm{IP}_{\mathbf{A}}$ ist die eindeutig bestimmte inklusionsminimale erzeugende Menge für die Familie $\mathrm{IP}_{\mathbf{A}}$. Gleichzeitig ist H eine universelle Testmenge für die Familie $\mathrm{IP}_{\mathbf{A}}$.*

Siehe z. B. [Urb98, HMW]. □

Der Begriff der erzeugenden Menge wurde in [Urb98] eingeführt, um das den Gravermengen zugrundeliegende Prinzip zu verallgemeinern. Zum Beweis der Eigenschaft, daß H eine universelle Testmenge ist, muß nämlich nur gezeigt werden, daß H eine erzeugende Menge ist; den Rest liefert das folgende Lemma [Urb98]:

Lemma 1.16. *Sei $\mathscr{F}$ eine Familie von ganzzahligen Programmen wie in Definition 1.13. Dann ist jede erzeugende Menge für $\mathscr{F}$ eine universelle Testmenge für $\mathscr{F}$.*

Beweis. Sei G eine erzeugende Menge für $\mathscr{F}$. Sei $\mathrm{IP} \in \mathscr{F}$ ein Programm mit optimaler Lösung $\mathbf{x}^*$, und sei $\mathbf{x}$ ein für IP zulässiger, bezüglich $\mathbf{c}^\top$ nicht-optimaler Punkt. Es gibt nun eine Zahl $l \in \mathbf{N}$ und Vektoren $\mathbf{t}^1, \dots, \mathbf{t}^l \in G$ mit $\mathbf{x}^* - \mathbf{x} = \sum_{i=1}^l \mathbf{t}^i$ und $\mathbf{t}^1, \dots, \mathbf{t}^l \preceq_{\mathbf{A}} \mathbf{x}^* - \mathbf{x}$. Wegen $\mathbf{c}^\top \mathbf{x}^* > \mathbf{c}^\top \mathbf{x}$ gibt es einen Index $i \in \{1, \dots, l\}$ mit $\mathbf{c}^\top \mathbf{t}^i > 0$.

Es gilt $\mathbf{A}\mathbf{t}^i \preceq \mathbf{A}(\mathbf{x}^* - \mathbf{x}) = \mathbf{0}$, und daher ist $\mathbf{A}(\mathbf{x} + \mathbf{t}^i) = \mathbf{A}\mathbf{x} = \mathbf{b}$. Wegen $\mathbf{t}^i \preceq \mathbf{x}^* - \mathbf{x}$ gilt außerdem

$$l_j \leq \min\left\{x_j^*, x_j\right\} \leq x_j + t_j^i \leq \max\left\{x_j^*, x_j\right\} \leq u_j$$

für $j \in \{1, \dots, n\}$. Somit ist $\mathbf{x} + \mathbf{t}^i$ zulässig für IP. □

Bemerkung 1.17. Ist eine erzeugende Menge G für eine Familie von ganzzahligen Programmen bekannt, so kann das Problem (DIR-AUG) für die Programme aus dieser Familie gelöst werden [HMW]: Sei vorausgesetzt, daß es einen Vektor $\mathbf{t} \in \mathbf{Z}^n$ gibt, so daß $\mathbf{x} + \mathbf{t}$ zulässig ist und $\mathbf{c}^\top \mathbf{t}^+ - \bar{\mathbf{c}}^\top \mathbf{t}^- > 0$ gilt. Dann besitzt $\mathbf{t}$ eine Zerlegung in Vektoren $\mathbf{t}^1, \dots, \mathbf{t}^l \in G$ mit $\mathbf{t}^i \preceq_{\mathbf{A}} \mathbf{t}$ für $i \in \{1, \dots, l\}$,

$$\mathbf{t} = \mathbf{t}^1 + \dots + \mathbf{t}^l.$$

Insbesondere gelten $\mathbf{t}^+ = \left(\mathbf{t}^1\right)^+ + \dots + \left(\mathbf{t}^l\right)^+$ und $\mathbf{t}^- = \left(\mathbf{t}^1\right)^- + \dots + \left(\mathbf{t}^l\right)^-$, also auch

$$\mathbf{c}^\top \mathbf{t}^+ - \bar{\mathbf{c}}^\top \mathbf{t}^- = \left(\mathbf{c}^\top \left(\mathbf{t}^1\right)^+ - \bar{\mathbf{c}}^\top \left(\mathbf{t}^1\right)^-\right) + \dots + \left(\mathbf{c}^\top \left(\mathbf{t}^l\right)^+ - \bar{\mathbf{c}}^\top \left(\mathbf{t}^l\right)^-\right) > 0.$$

Daher gibt es einen Index i mit $\mathbf{c}^\top \left(\mathbf{t}^i\right)^+ - \bar{\mathbf{c}}^\top \left(\mathbf{t}^i\right)^- > 0$. Wie im Beweis von Lemma 1.16 gezeigt wurde, ist $\mathbf{x} + \mathbf{t}^i$ (für jeden Index i) ein zulässiger Punkt. Somit ist $\mathbf{t}^i$ eine Lösung des Problems (DIR-AUG).

2 Primitive Partitionsidentitäten

In der Einführung der Partitionsidentitäten folgen wir [Urb98] und [DGS96].

Definition 2.1. Eine Identität der Form

$$p_1 + \cdots + p_k = q_1 + \cdots + q_l \tag{2.1}$$

mit Zahlen $p_1, \ldots, p_k; q_1, \ldots, q_l \in \mathbf{N}$ heißt *Partitionsidentität*. Die in ihr auftretenden Zahlen $p_1, \ldots, p_k; q_1, \ldots, q_l$ heißen die *Teile* der Identität, und ihr Maximum heißt das *größte Teil* der Identität.

Die Identität (2.1) heißt *primitiv*, wenn keine echte Teilsumme auf der linken Seite gleich einer echten Teilsumme auf der rechten Seite ist.

Wir wollen stets annehmen, daß $p_i \neq q_j$ für alle $i \in \{1, \ldots, k\}$ und $j \in \{1, \ldots, l\}$ gelte, das heißt, (2.1) enthalte keine trivialen Teilidentitäten der Form $a = a$.

Einer Partitionsidentität (2.1), deren größtes Teil höchstens n ist, ordnen wir ein Tupel $\mathbf{x} \in \mathbf{Z}^n$ zu vermöge

$$x_i := \#\big\{\, j \in \{1, \ldots, k\} : p_j = i \,\big\} - \#\big\{\, j \in \{1, \ldots, l\} : q_j = i \,\big\}$$

für $i \in \{1, \ldots, n\}$. Mit $\mathbf{n} := (1, \ldots, n)^\top$ gilt dann $\mathbf{n}^\top \mathbf{x} = 0$. Umgekehrt bestimmt jedes Tupel $\mathbf{x} \in \mathbf{Z}^n$ mit $\mathbf{n}^\top \mathbf{x} = 0$ eine Partitionsidentität, und zwar eindeutig bis auf die Reihenfolge der Teile.

Eine Partitionsidentität ist offenbar genau dann primitiv, wenn das ihr zugeordnete Tupel irreduzibel bezüglich $\mathbf{n}^\top$ ist; vergleiche [HMW] oder [Urb98]. Damit bilden die den primitiven Partitionsidentitäten zugeordneten Tupel genau die Gravermenge für die Familie der *Rucksackprobleme*

$$\max\big\{\, \mathbf{c}^\top \mathbf{x} : \mathbf{n}^\top \mathbf{x} = b,\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\ \mathbf{x} \in \mathbf{Z}^n \,\big\} \tag{2.2}$$

mit beliebigen Zielfunktionalen $\mathbf{c}^\top \in \mathbf{R}^{(n)}$, Schranken $\mathbf{l}, \mathbf{u} \in \mathbf{Z}^n$ und rechten Seiten $b \in \mathbf{Z}$.

2.1 Das Verfahren von Urbaniak

R. Urbaniak [Urb98] schlägt ein iteratives Verfahren zur Berechnung der primitiven Partitionsidentitäten vor. Die von ihr formulierten Algorithmen 4.3.8 und 4.3.11 bestimmen aus der Menge $\mathscr{P}(n)$ der primitiven Partitionsidentitäten mit größtem Teil höchstens n die Menge $\mathscr{P}(n+1)$ derer mit größtem Teil höchstens $n+1$.

Wir geben das Verfahren mit Tupeln anstelle von Identitäten wieder, um eine eindeutige und zugleich einfachere Notation zu erhalten.

Sei P_n die Menge der bezüglich $\mathbf{n}^\top$ irreduziblen Tupel $\mathbf{x} \in \mathbf{Z}^n \cap \ker \mathbf{n}^\top$, welche die Bedingung $x_{\max \operatorname{supp} \mathbf{x}} > 0$ erfüllen. Diese Bedingung fordert, daß das größte Teil einer Identität auf der linken Seite auftreten möge; auf diese Weise ziehen wir uns auf die Betrachtung eines Vertreters für ein Paar von entgegengesetzten Identitäten zurück.

Für $t \in \mathbf{N}_0$ sei P_n^t die Menge der Tupel $\mathbf{x} \in P_n$ mit $x_n = t$. Seien ferner $\iota_{n+1} : \mathbf{R}^n \hookrightarrow \mathbf{R}^{n+1}$, $\mathbf{x} \mapsto (\mathbf{x}, 0)$, die Einbettung des $\mathbf{R}^n$ in den $\mathbf{R}^{n+1}$ und $\hat{\mathbf{n}} := (1, \ldots, n+1)^\top$.

Algorithmus 2.2 (Urbaniak).

input P_n;
output P_{n+1};
$P_{n+1}^0 \leftarrow \iota_{n+1}(P_n)$;
$(*\ P_{n+1}^0$ ist berechnet $*)$
$P_{n+1} \leftarrow P_{n+1}^0$;
$P_{n+1}^0 \leftarrow P_{n+1}^0 \cup \{\mathbf{0}\}$;
for $t \leftarrow 0$ **to** $n-1$ **do**
  $P_{n+1}^{t+1} \leftarrow \emptyset$;
  for all $\mathbf{v} \in P_{n+1}^t$ **do**
    for $i \leftarrow 1$ **to** $\lfloor \frac{n+1}{2} \rfloor$ **do**
      $\mathbf{w} \leftarrow \mathbf{v} + \mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i}$;
      $\langle$Reduziere $\mathbf{w}$ mit P_{n+1} und füge in P_{n+1}^{t+1} ein$\rangle$;
      if $t = 0$ **then**
        $\mathbf{w} \leftarrow -\mathbf{v} + \mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i}$;
        $\langle$Reduziere $\mathbf{w}$ mit P_{n+1} und füge in P_{n+1}^{t+1} ein$\rangle$
      fi
    od
  od;
  $(*\ P_{n+1}^{t+1}$ ist berechnet $*)$
  $P_{n+1} \leftarrow P_{n+1} \cup P_{n+1}^{t+1}$
od.

$\langle$Reduziere $\mathbf{w}$ mit R und füge in P ein$\rangle \equiv$
  while $\exists \mathbf{u} \in R \cup -R$ mit $\mathbf{u} \preceq \mathbf{w}$ **do**
    $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{u}$
  od;
  if $\mathbf{w} \neq \mathbf{0}$ **then**
    $P \leftarrow P \cup \{\mathbf{w}\}$
  fi.

Zum Beweis der Korrektheit dieses Verfahrens verwenden wir eine Abschätzung für die ℓ_1-Norm von Vektoren aus P_n.

Lemma 2.3 (Diaconis–Graham–Sturmfels). *Für* $\mathbf{x} \in P_n$ *gilt*

$$\|\mathbf{x}\|_1 \leq \max\operatorname{supp}\mathbf{x}^+ + \max\operatorname{supp}\mathbf{x}^- \leq 2n-1.$$

Dies ist die Aussage von Korollar 1 in [DGS96]. $\square$

Folgerung 2.4. *Für* $\mathbf{x} \in P_n$ *gilt* $x_n \leq n-1$.

Beweis. Angenommen, es ist $x_n \geq n$. Dann gilt $(n-1)\,\|\mathbf{x}^-\|_1 \geq \mathbf{n}^\top\mathbf{x}^- = \mathbf{n}^\top\mathbf{x}^+ \geq n^2$, also

$$\left\|\mathbf{x}^-\right\|_1 \geq \frac{n^2}{n-1} > n \qquad \text{und} \qquad \left\|\mathbf{x}^+\right\|_1 \geq n.$$

Somit gilt $\|\mathbf{x}\|_1 = \|\mathbf{x}^+\|_1 + \|\mathbf{x}^-\|_1 > 2n$; das ist ein Widerspruch zu Lemma 2.3. $\square$

Satz 2.5 (Urbaniak). *Algorithmus 2.2 ist korrekt.*

Beweis. In jedem Durchlauf der Schleife 23–25 wird $\|\mathbf{w}\|_1 \in \mathbf{N}_0$ verkleinert; daher bricht die Schleife nach endlich vielen Schritten ab. Damit ist das ganze Verfahren endlich.

Die Bedingung in Zeile 4 wird offensichtlich erfüllt. Wir zeigen die Gültigkeit der Bedingung in Zeile 19 durch Induktion nach t.

Sei dazu $t \in \{0,\dots,n-1\}$ beliebig, und sei P_n^i für $i \in \{0,\dots,t\}$ bereits berechnet. Sei $\mathbf{w} \in P_{n+1}^{t+1}$. Da $\mathbf{w} \neq \mathbf{0}$ ist und $\hat{\mathbf{n}}^\top\mathbf{w} = 0$ gilt, ist $\operatorname{supp}\mathbf{w}^- \neq \varnothing$. Sei $i \in \operatorname{supp}\mathbf{w}^-$ und $\mathbf{v} := \mathbf{w} + \mathbf{e}_i + \mathbf{e}_{(n+1)-i} - \mathbf{e}_{n+1}$. Ist $\mathbf{v} = \mathbf{0}$, so wird $\mathbf{w}$ wegen Zeile 6 im Durchlauf der äußeren Schleife für $t = 0$ erzeugt. Sei also $\mathbf{v} \neq \mathbf{0}$. Dann kann i so gewählt werden, daß der Fall $w_i = -1$ mit $i = (n+1)-i$ nicht eintritt. Wäre nun $(n+1)-i \in \operatorname{supp}\mathbf{w}^-$, so wäre $\mathbf{v} \preceq_{\hat{\mathbf{n}}^\top} \mathbf{w}$, also $\mathbf{w}$ reduzibel bezüglich $\hat{\mathbf{n}}^\top$. Somit gilt $w_{(n+1)-i} \geq 0$. Angenommen, $\mathbf{v} \notin \pm P_{n+1}^t$. Dann gibt es $\mathbf{v}^1, \mathbf{v}^2 \in \mathbf{Z}^n \setminus \{\mathbf{0}\}$ mit $\mathbf{v}^1, \mathbf{v}^2 \preceq_{\hat{\mathbf{n}}^\top} \mathbf{v}$ und $\mathbf{v} = \mathbf{v}^1 + \mathbf{v}^2$. Wenigstens einer der beiden Summanden ist in der $\big((n+1)-i\big)$-ten Komponente kleiner als die Summe; o. B. d. A. sei dies $\mathbf{v}^1$. Dann gilt

$$0 \leq v^1_{(n+1)-i} \leq w_{(n+1)-i} < v_{(n+1)-i} = w_{(n+1)-i} + 1.$$

Daher ist $\mathbf{v}^1 \preceq_{\hat{\mathbf{n}}^\top} \mathbf{w}$, also $\mathbf{w}$ reduzibel bezüglich $\hat{\mathbf{n}}^\top$. – Somit ist $\mathbf{v} \in \pm P_{n+1}^t$, und $\mathbf{w}$ wird im t-ten Durchlauf der äußeren Schleife aus $\mathbf{v}$ erzeugt.

Andererseits werden während des Verfahrens nur Vektoren aus $\mathbf{Z}^{n+1} \cap \ker\hat{\mathbf{n}}^\top$ erzeugt. Ist fernerhin ein Vektor $\mathbf{w} \in P_{n+1}^{t+1}$ mit $w_{n+1} > 0$ reduzibel bezüglich $\hat{\mathbf{n}}^\top$, so gibt es einen Reduzierer $\mathbf{u} \neq \mathbf{0}$ mit $u_{n+1} < w_{n+1}$. Da bei der Berechnung von P_{n+1}^{t+1} bereits alle Elemente von P_{n+1}^i für alle $i \in \{0,\dots,t\}$ bekannt sind, sichert die Prozedur in den Zeilen 23–28, daß nur irreduzible Vektoren, also Elemente von P_{n+1}^{t+1}, gespeichert werden.

Wegen Folgerung 2.4 gilt $P_{n+1} = \bigcup_{t=0}^n P_{n+1}^t$. Somit berechnet der Algorithmus alle Vektoren von P_{n+1}. $\square$

Auswertung. R. Urbaniak konnte mit einer Implementation von Algorithmus 2.2 die primitiven Partitionsidentitäten bis $n = 22$ berechnen; siehe Tafel 2.1.

Zur Berechnung der primitiven Partitionsidentitäten läßt sich auch der allgemeine Hilbertbasis-Algorithmus 1.8 einsetzen. Zum Vergleich der Leistung der Algorithmen 1.8 und 2.2 wurde Algorithmus 1.8 unter Verwendung von Techniken aus Abschnitt 2.3 implementiert; siehe wiederum Tafel 2.1.

Der Grund für das bessere Laufzeitverhalten des Verfahrens von R. Urbaniak ist, daß in ihm sehr wenige Vektoren, nämlich

$$\mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i} \quad \text{für } i \in \left\{1, \ldots, \left\lfloor \tfrac{n}{2} \right\rfloor\right\},$$

ausreichen, um aus den bereits bekannten Vektoren die verbleibenden zu erzeugen. In Algorithmus 1.8 werden hingegen alle *Paare* von bekannten Vektoren, die eine bestimmte Orthantenbedingung erfüllen, kombiniert, um die fehlenden Vektoren zu erzeugen.

In diesem Zusammenhang verdient Bemerkung 1.11 b erneut Beachtung. Ihre Aussage ist, daß die Vektoren der Menge G in Algorithmus 1.8 i. a. nicht ausreichen, um aus den bekannten Vektoren die verbleibenden zu erzeugen; dazu ist eine größere Menge nötig, die *a priori* nicht bekannt ist.

In Abschnitt 2.4 werden wir analysieren, welche Eigenschaft es uns im Fall der primitiven Partitionsidentitäten erlaubt, mit einer kleinen und *a priori* bekannten Menge von Vektoren zu arbeiten. Mit dieser Kenntnis werden wir dann ein Verfahren zur Berechnung minimaler erzeugender Mengen für Familien von Rucksackproblemen formulieren können, welches für „genügend dichte" Gewichtmengen schneller als Algorithmus 1.8 ist.

2.2 Verbesserungen des Verfahrens

Bedingungen für Irreduzibilität. Unser Beweis von Satz 2.5 zeigte bereits, daß die Konstruktion eines neuen Vektors $\mathbf{w}$ aus einem bereits bekannten Vektor $\mathbf{v}$ durch Addition von Vektoren der Gestalt $\mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i}$ nur unter bestimmten Vorzeichenbedingungen an die Komponenten erforderlich ist. Diese Bedingungen werden wir im folgenden genauer diskutieren.

Lemma 2.6. *Sei* $\mathbf{v} \in P^t_{n+1}$ *und* $\mathbf{w} := \mathbf{v} + \mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i}$.

a. *Sind* v_i, $v_{(n+1)-i} > 0$, *so ist* $\mathbf{w} \in P^{t+1}_{n+1}$.

b. *Sind* v_i, $v_{(n+1)-i} \leq 0$, *so ist* $\mathbf{w} \notin P^{t+1}_{n+1}$.

Beweis. Seien v_i, $v_{(n+1)-i} > 0$. Angenommen, es ist $\mathbf{w} = \mathbf{w}^1 + \mathbf{w}^2$ mit $\mathbf{0} \neq \mathbf{w}^1, \mathbf{w}^2 \preceq_{\hat{\mathbf{n}}^\top} \mathbf{w}$. Sei o. B. d. A. $w^1_{n+1} < w_{n+1}$. Dann gelten $w^1_{n+1} \leq v_{n+1}$ und $w^1_i < v_i$. Somit ist $\mathbf{w}^1 \preceq_{\hat{\mathbf{n}}^\top} \mathbf{v}$ und $\mathbf{w}^1 \neq \mathbf{v}$, d. h., $\mathbf{v}$ ist im Widerspruch zur Voraussetzung reduzibel bezüglich $\hat{\mathbf{n}}^\top$.

Seien v_i, $v_{(n+1)-i} \leq 0$. Dann gelten $\mathbf{0} \neq \mathbf{v} \neq \mathbf{w}$ und $\mathbf{v} \preceq_{\hat{\mathbf{n}}^\top} \mathbf{w}$. Somit ist $\mathbf{w}$ reduzibel bezüglich $\hat{\mathbf{n}}^\top$. □

n	$\#P_n$	Laufzeit		
		Alg. 1.8[a]	Alg. 2.2[b,c]	Alg. 2.12[a,c]
2	1			
3	5			
4	15			
5	47			
6	102			
7	276			
8	578			
9	1 261	1 s		
10	2 465	5 s		
11	5 362	23 s	2 s	
12	9 285	1 min 24 s	5 s	
13	18 900	6 min 18 s	23 s	2 s
14	33 269	22 min 32 s	1 min 08 s	5 s
15	58 171	1 h 19 min	3 min 39 s	12 s
16	99 328	4 h 23 min	12 min 06 s	24 s
17	181 514	15 h 25 min	49 min 23 s	51 s
18	287 239		2 h 03 min	1 min 45 s
19	502 116		7 h 01 min	3 min 31 s
20	775 710		16 h 03 min	7 min 00 s
21	1 239 710		42 h 44 min	13 min 27 s
22	1 956 334		101 h 25 min	25 min 18 s
23	3 210 736			49 min 48 s
24	4 660 786			1 h 40 min
25	7 297 823			3 h 13 min
26	10 997 235			6 h 16 min
27	16 536 803			12 h 06 min

[a] Auf einer Sun Enterprise 450 Server.
[b] Auf einer Sun Ultra Enterprise 3000; Angaben aus [Urb98].
[c] Angegeben sind die Gesamtlaufzeiten für die Berechnung von P_n, nicht die Laufzeiten für die Berechnung von P_n aus P_{n-1}.

Tafel 2.1: Berechnung der primitiven Partitionsidentitäten

Definition 2.7. In der Situation von Lemma 2.6 a sagen wir, $\mathbf{w}$ *entstehe aus* $\mathbf{v}$ *durch leichtes Heben.*

Satz 2.8. *Sei* $\mathbf{v} \in P^t_{n+1}$ *mit* $v_i \leq 0$ *und* $v_{(n+1)-i} > 0$. *Sei* $\mathbf{w} := \mathbf{v} + \mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i}$. *Dann ist entweder* $\mathbf{w}$ *irreduzibel bezüglich* $\hat{\mathbf{n}}^\top$, *oder* $\mathbf{w}$ *zerfällt in zwei bezüglich* $\hat{\mathbf{n}}^\top$ *irreduzible Vektoren* $\mathbf{w}^1$, $\mathbf{w}^2$ *mit*

$$w^1_i = w_i, \qquad w^2_i = 0, \tag{2.3a}$$

$$w^1_{n+1} = 0, \qquad w^2_{n+1} = w_{n+1}, \tag{2.3b}$$

wobei $\mathbf{w}^2$ *aus einem Vektor* $\mathbf{u} \in P^t_{n+1}$ *durch leichtes Heben entsteht.*

Beweis. Sei $\mathbf{w} = \mathbf{w}^1 + \cdots + \mathbf{w}^l$ mit einer geeigneten Zahl $l \in \mathbf{N}$ und bezüglich $\hat{\mathbf{n}}^\top$ irreduziblen Vektoren $\mathbf{w}^1, \ldots, \mathbf{w}^l \preceq_{\hat{\mathbf{n}}^\top} \mathbf{w}$. Wegen

$$0 \leq w^k_{n+1-i} \leq w_{n+1-i} < v_{n+1-i} \tag{2.4}$$

gilt $\mathbf{v} \neq \mathbf{w}^k$ für $k \in \{1, \ldots, l\}$. Deshalb folgt aus der Irreduzibilität von $\mathbf{w}^k$ insbesondere $\mathbf{v} \not\preceq_{\hat{\mathbf{n}}^\top} \mathbf{w}^k$, und daher gibt es für jedes $k \in \{1, \ldots, l\}$ einen Index $j(k) \in \{1, \ldots, n+1\}$, für den $v^k_{j(k)} \not\preceq w^k_{j(k)}$ gilt. Wegen $w^k_j \preceq w_j \preceq v_j$ für $j \notin \{i,\ n+1\}$ gilt $j(k) \in \{i,\ n+1\}$ für alle $k \in \{1, \ldots, l\}$. Wegen $w_i = v_i - 1$ und $w_{n+1} = v_{n+1} + 1$ folgt nun, daß $l \leq 2$ ist.

Ist $l = 1$, so ist $\mathbf{w}$ irreduzibel bezüglich $\hat{\mathbf{n}}^\top$. Ist $l = 2$, so erfüllen $\mathbf{w}^1$ und $\mathbf{w}^2$ nach eventueller Vertauschung die Bedingungen (2.3).

Sei schließlich $\mathbf{u} := \mathbf{w}^2 - \mathbf{e}_{n+1} + \mathbf{e}_i + \mathbf{e}_{(n+1)-i}$. Angenommen, $\mathbf{u}$ ist reduzibel bezüglich $\hat{\mathbf{n}}^\top$. Da $u_i = 1$ ist, gibt es einen Reduzierer $\mathbf{u}^2 \preceq_{\hat{\mathbf{n}}^\top} \mathbf{u}$ mit $u^2_i = 0$ und $\mathbf{u}^2 \neq \mathbf{0}$. Dieser erfüllt $\mathbf{u}^2 \preceq_{\hat{\mathbf{n}}^\top} \mathbf{v}$. Das ist ein Widerspruch zur Irreduzibilität von $\mathbf{v}$. – Nach (2.4) ist $w^2_{n+1-i} \geq 0$, und nach (2.3a) ist $w^2_i = 0$; somit liegt für $\mathbf{u}$ und $\mathbf{w}^2$ die Situation aus Lemma 2.6 vor. □

Bemerkung 2.9. **a.** Satz 2.8 sichert die Existenz eines Reduzierers $\mathbf{w}^1$ für $\mathbf{w}$ mit verschwindender letzter Komponente, sobald $\mathbf{w}$ überhaupt reduzibel ist; außerdem ist dann $\mathbf{w} - \mathbf{w}^1$ $(= \mathbf{w}^2)$ irreduzibel. Um also ein Tupel $\mathbf{w}$, das in Algorithmus 2.2 erzeugt wird, so weit wie möglich zu reduzieren, genügt es, einen Reduzierer $\mathbf{w}^1$ mit $w^1_{n+1} = 0$ zu finden. Daher können wir im Algorithmus eine *statische* Datenstruktur einsetzen, welche die Menge $\iota_{n+1}(P_n)$ speichert und welche die effiziente Suche nach einem Reduzierer für $\mathbf{w}$ in $\pm\iota_{n+1}(P_n)$ gestattet. Für diese Aufgabe geeignete Datenstrukturen werden in Abschnitt 2.3 vorgestellt.

b. Unser Beweis von Satz 2.5 zeigt, daß es für jedes Tupel $\mathbf{w} \in P^{t+1}_{n+1}$ ein Tupel $\mathbf{v} \in \pm P^t_{n+1}$ und eine Zahl $i \in \left\{1, \ldots, \left\lfloor \frac{n}{2} \right\rfloor\right\}$ gibt mit $\mathbf{w} = \mathbf{v} + \mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i}$. Ist daher ein von Algorithmus 2.2 erzeugtes Tupel $\mathbf{w}$ reduzibel, so kann es sofort verworfen werden; die Reduktion von $\mathbf{w}$ muß nicht tatsächlich ausgeführt werden.

c. Schließlich können wir der Aussage von Satz 2.8 entnehmen, daß es keinen Sinn machen wird, in einer Implementierung des Verfahrens die Irreduzibilität von $\mathbf{w}^2$ ausnutzen zu wollen, da $\mathbf{w}^2$ bereits durch leichtes Heben entsteht.

„Austrocknen von Quellen“. Während der Ausführung von Algorithmus 2.2 werden viele (reduzible und irreduzible) Tupel mehrfach erzeugt und damit auch mehrfach auf Irreduzibilität geprüft. Wir diskutieren nun Möglichkeiten, um diesen Aufwand zu vermeiden.

Eine wesentliche Beobachtung hierzu ist, daß das *Auffinden eines Elementes* in einer Menge P leichter ist als das *Finden eines Reduzierers* in P. Tatsächlich läßt sich für das erstere Problem z. B. eine dynamisch wachsende Hashtabelle mit Verkettung einsetzen; unter gewissen Voraussetzungen erlaubt eine solche das Auffinden eines Elementes in $\mathrm{O}(1)$ erwarteter Zeit bzw. $\Theta(\log \#P / \log\log \#P)$ erwarteter *Worst-case*-Zeit sowie Einfüge- und Löschoperationen in $\mathrm{O}(1)$ erwarteter amortisierter Zeit und benötigt nur $\mathrm{O}(\#P)$ Speicherplatz [Smi97, Meh84].[1] Datenstrukturen zur Lösung des letzteren Problems werden in Abschnitt 2.3 studiert; keine der dort angegebenen Strukturen gestattet das Finden eines Reduzierers in einer besseren Zeitkomplexität als $\mathrm{O}(\log^{n-2} \#P)$.

Man könnte nun versuchen, auch alle im Verfahren bisher erzeugten *reduziblen* Tupel abzuspeichern, um vor der Überprüfung eines Tupels $\mathbf{w}$ auf Irreduzibilität nachsehen zu können, ob $\mathbf{w}$ schon früher erzeugt worden ist. Es zeigt sich aber, daß es in der Praxis aufgrund der Beschränkung des Speicherplatzes nicht möglich ist, die erzeugten reduziblen Tupel zu speichern.

Statt dessen speichern wir die Menge P^t_{n+1} der irreduziblen „Quelltupel“ in einer Hashtabelle und assoziieren mit jedem Quelltupel $\mathbf{v} \in P^t_{n+1}$ eine gewisse Indexmenge $I_\mathbf{v} \subseteq \left\{1, \dots, \left\lfloor \frac{n}{2} \right\rfloor\right\}$; im Fall $t = 0$ berücksichtigen wir zusätzlich die Tupel $\mathbf{v} \in -P^0_{n+1}$. Hierbei zeigt $i \in I_\mathbf{v}$ an, daß aus $\mathbf{v}$ noch $\mathbf{w} := \mathbf{v} + \mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i}$ erzeugt werden muß. Während des Aufbaus der Hashtabelle für P^t_{n+1} initialisieren wir $I_\mathbf{v}$ für alle $\mathbf{v} \in P^t_{n+1}$ unter Beachtung der Bedingungen aus Lemma 2.6. Für jedes Tupel $\mathbf{w}$, das während des Verfahrens erzeugt wird, ermitteln wir die möglichen Quelltupel $\mathbf{u}$ und modifizieren deren Indexmengen $I_\mathbf{u}$:

Algorithmus 2.10 (Austrocknen der Quellen).

```
input w ∈ Z^{n+1}, P^t_{n+1};
for i ← 1 to ⌊n/2⌋ do
    u ← w − e_{n+1} + e_i + e_{(n+1)−i};
    Suche u in P^t_{n+1};
    if u ∈ P^t_{n+1} then
        I_u ← I_u \ {i}
    fi
od.
```

[1] Diese Komplexitäten gelten unter der Annahme, daß der Hashwert eines Elementes in $\mathrm{O}(1)$ Zeit berechnet werden kann. In unserem Fall ist jedes Element aber ein Tupel der Länge n, so daß für jede geeignete Hashfunktion die Berechnung des Hashwertes $\Omega(n)$ Zeit erfordert. Setzen wir eine Hashfunktion ein, die in $\Theta(n)$ Zeit berechnet werden kann, so erlaubt die Datenstruktur das Auffinden in $\Theta(n)$ erwarteter Zeit bzw. $\Theta(n + \log \#P / \log\log \#P)$ erwarteter *Worst-case*-Zeit sowie Einfüge- und Löschoperationen in $\Theta(n)$ erwarteter amortisierter Zeit und benötigt $\mathrm{O}(n \#P)$ Speicherplatz.

Bemerkung 2.11. Falls $\mathbf{w} \in P_{n+1}^{t+1}$ ist und $u_i < 0$ oder $u_{(n+1)-i} < 0$ gilt, so wird die Bedingung in Zeile 5 stets erfüllt sein. Dies folgt aus dem Beweis von Satz 2.5.

Durch die Verwendung von Algorithmus 2.10 können wir erreichen, daß kein Vektor mehr als einmal erzeugt wird. – In der Implementierung zeigte sich jedoch, daß es besser ist, nur dann Algorithmus 2.10 einzusetzen bzw. die Schritte in Zeilen 4–7 auszuführen, wenn die in Bemerkung 2.11 genannten Bedingungen erfüllt sind, also im voraus gesichert ist, daß der betriebene Aufwand tatsächlich zum Austrocknen einer Quelle führen wird. Ferner erwies es sich als vorteilhaft, zuerst alle durch leichtes Heben nach Lemma 2.6 a entstehenden Vektoren zu erzeugen und ihre Quellen auszutrocknen. Auf diese Weise müssen weniger Vektoren auf Irreduzibilität überprüft werden.

Zusammenfassung. Wir sind nun in der Lage, eine Variante von Algorithmus 2.2 zu formulieren, welche die bisher diskutierten Techniken einsetzt.

Algorithmus 2.12 (Modifizierter Urbaniak-Algorithmus).

input P_n;
output P_{n+1};
Konstruiere Datenstruktur $\mathscr{D}$ für Suche nach Reduzierern in $\pm\iota_{n+1}(P_n)$;
$P_{n+1}^0 \leftarrow \iota_{n+1}(P_n)$;
for all $\mathbf{v} \in \pm P_{n+1}^0$ **do**
    $I_\mathbf{v} \leftarrow \left\{ i \in \left\{1,\dots,\left\lfloor \frac{n}{2} \right\rfloor\right\} : v_i > 0 \text{ oder } v_{(n+1)-i} > 0 \right\}$;
od;
$P_{n+1}^1 \leftarrow \varnothing$;
for $i \leftarrow 1$ **to** $\left\lfloor \frac{n+1}{2} \right\rfloor$ **do**
    $\mathbf{v} \leftarrow \mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i}$;
    $P_{n+1}^1 \leftarrow P_{n+1}^1 \cup \{\mathbf{v}\}$;
    $I_\mathbf{v} \leftarrow \varnothing$
od;
$\langle$Berechne die Menge P_{n+1}^1 der Vektoren der nächsten Stufe aus $\pm P_{n+1}^0\rangle$;
for $t \leftarrow 1$ **to** $n-1$ **do**
    $P_{n+1}^{t+1} \leftarrow \varnothing$;
    $\langle$Berechne die Menge P_{n+1}^{t+1} der Vektoren der nächsten Stufe aus $P_{n+1}^t\rangle$;
od;
$P_{n+1} \leftarrow \bigcup_{t=0}^n P_{n+1}^t$.

$\langle$Berechne die Menge Q der Vektoren der nächsten Stufe aus $P\rangle \equiv$
    for all $\mathbf{v} \in P$ **do**
        for $i \leftarrow 1$ **to** $\left\lfloor \frac{n}{2} \right\rfloor$ **do**
            if $v_i > 0$ und $v_{(n+1)-i} > 0$ **then**
                $\mathbf{w} \leftarrow \mathbf{v} + \mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i}$;   (∗ irreduzibel nach Lemma 2.6 a ∗)
                $\langle$Füge $\mathbf{w}$ in Q ein und trockne die Quellen von $\mathbf{w}$ in P aus$\rangle$
            fi

    od
  od;
  for all $\mathbf{v} \in P$ **do**
    for all $i \in I_{\mathbf{v}}$ **do**
      $\mathbf{w} \leftarrow \mathbf{v} + \mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i}$;
      if $\nexists \mathbf{u} \in \mathscr{D}$ mit $\mathbf{u} \preceq_{\hat{\mathbf{n}}^\top} \mathbf{w}$ **then**
        ⟨Füge $\mathbf{w}$ in Q ein und trockne die Quellen von $\mathbf{w}$ in P aus⟩
      fi
    od
  od.

⟨Füge $\mathbf{w}$ in Q ein und trockne die Quellen von $\mathbf{w}$ in P aus⟩ $\equiv$
  if $\mathbf{w} \notin Q$ **then**
    $Q \leftarrow Q \cup \{\mathbf{w}\}$;
    Wende Algorithmus 2.10 auf $\mathbf{w}$ und P an
  fi.

Die Korrektheit dieser Variante folgt aus der vorangegangenen Diskussion. □

Auswertung. Mit einer Implementierung von Algorithmus 2.12 konnten die primitiven Partitionsidentitäten bis $n = 27$ berechnet werden. Das Laufzeitverhalten gegenüber der Implementierung von Algorithmus 2.2 in [Urb98] wurde entscheidend verbessert; siehe Tafel 2.1 auf Seite 11.

Für $n = 27$ ist bereits $\#P_n \approx 16{,}5 \cdot 10^6$, und damit benötigt man allein für die Speicherung der Tupel etwa

$$\#P_n \cdot n \cdot \lceil \log_2 2n \rceil \text{ bit} \approx 16{,}5 \cdot 10^6 \cdot 27 \cdot 6 \,\text{bit} \approx 334\,\text{MB}.$$

Auf dem verwendeten Rechnersystem konnten für die Berechnung von P_n für $n = 28$ nicht mehr alle Daten intern gespeichert werden. Auf die Berechnung der P_n für $n \geq 28$ wurde daher verzichtet.

2.3 Effiziente Datenstrukturen zur Speicherung der irreduziblen Tupel

In der Einführung folgen wir [Cha90].

Definition 2.13. *Problem der orthogonalen Bereichssuche im statischen Fall.* Gegeben sei eine endliche Menge P von Punkten des $\mathbf{Q}^d$. Seien $(\mathscr{S}, +)$ eine kommutative Halbgruppe und $w\colon P \to \mathscr{S}$ eine Abbildung, die *Gewichtsfunktion*. Konstruiere eine Datenstruktur $\mathscr{D}(P)$ aus P, welche Anfragen der folgenden Gestalt erlaubt:

Gegeben ein achsenparalleler Quader $Q = [a_1, b_1] \times \cdots \times [a_d, b_d]$, ermittle

$$\sum_{p \in P \cap Q} w(p).$$

Beispiel 2.14. **a.** Seien $(\mathscr{S}, +) = (\mathbf{Z}, +)$ und $w(p) = 1$ für $p \in P$. Dann liefert jede Anfrage die Anzahl der Punkte im vorgegebenen Quader.

b. In einigen Fällen genügt es uns zu entscheiden, ob im vorgegebenen Quader ein Punkt aus P liegt. Wir wählen dazu $(\mathscr{S}, +) = (\{0, 1\}, \max)$ und $w(p) = 1$ für $p \in P$.

c. Seien $(\mathscr{S}, +) = (2^P, \cup)$ und $w(p) = \{p\}$ für $p \in P$. Dann erhält man auf jede Anfrage die Menge der Punkte im vorgegebenen Quader.

In diesem Fall ist eine *ausgabesensitive Analyse* der Zeitkomplexität einer Anfrage sinnvoll, d. h., diese wird eine Funktion von $N := \#P$ und $K := \#(P \cap Q)$ sein.

Das Problem der orthogonalen Bereichssuche tritt häufig in den Bereichen des Datenbankentwurfs, der Computergrafik und der *computational geometry* auf [dB+97]. Für die in den beiden letzteren Bereichen auftretenden niedrigdimensionalen Räume, d. h. für „kleine“ d, ist das Problem sehr weitgehend untersucht worden; eine umfassende Bibliographie ist in [dB+97] zu finden.

Die Suche nach Reduzierern als orthogonale Bereichssuche. Seien $\mathbf{A} \in \mathbf{Q}^{m \times n}$ eine Matrix und $\tilde{P} \subset \mathbf{Z}^n$ eine endliche Menge von ganzzahligen Tupeln. Die Suche nach Reduzierern bezüglich $\mathbf{A}$ ist die folgende Aufgabe:

Gegeben ein Tupel $\mathbf{x} \in \mathbf{Z}^n$, ermittle die Menge der Tupel $\tilde{\mathbf{x}} \in \tilde{P}$ mit $\tilde{\mathbf{x}} \preceq_{\mathbf{A}} \mathbf{x}$.

Zur Beantwortung von Anfragen dieser Gestalt konstruieren wir die Einbettung von $\tilde{P}$ in den Graphenraum bezüglich $\mathbf{A}$,

$$P := \left\{ (\tilde{\mathbf{x}}, \mathbf{A}\tilde{\mathbf{x}}) : \tilde{\mathbf{x}} \in \tilde{P} \right\},$$

und wählen $(\mathscr{S}, +) = (2^P, \cup)$ und $w((\tilde{\mathbf{x}}, \mathbf{A}\tilde{\mathbf{x}})) := \{\tilde{\mathbf{x}}\}$ für $\tilde{\mathbf{x}} \in \tilde{P}$; siehe Beispiel 2.14 c. Ist nun ein Tupel $\mathbf{x} \in \mathbf{Z}^n$ gegeben, so setzen wir

$$[a_i, b_i] := \begin{cases} [0, x_i] & \text{für } x_i \geq 0, \\ [x_i, 0] & \text{für } x_i \leq 0 \end{cases} \qquad \text{für } i \in \{1, \ldots, d\}$$

und $Q := [a_1, b_1] \times \cdots \times [a_d, b_d]$. Dann liefert $\sum_{p \in P \cap Q} w(p)$ die gesuchte Menge der Tupel $\tilde{\mathbf{x}} \in \tilde{P}$ mit $\tilde{\mathbf{x}} \preceq_{\mathbf{A}} \mathbf{x}$.

Wie in Abschnitt 2.2 gezeigt wurde, genügt uns tatsächlich eine statische Datenstruktur, d. h., nach ihrer Konstruktion werden Anfragen gestellt, aber keine Einfüge- oder Löschoperationen ausgeführt.

Wir geben im folgenden einen Überblick über geeignete Datenstrukturen. Die Zeitkomplexität für eine Anfrage ist für den Fall von Beispiel 2.14 c angegeben.

Orthogonale Bereichssuche auf allgemeinen Schlüsselräumen. Zunächst werden Strukturen vorgestellt, die auf *Vergleichen* zwischen den Schlüsseln (also den Komponenten der Punkte) basieren und somit für beliebige endliche Punktmengen des $\mathbf{Q}^d$ einsetzbar sind.

Kd-trees. Eine klassische Datenstruktur für orthogonale Bereichssuche ist der *kd-tree*, d. i. ein balancierter Blattsuchbaum (*trie*), in dem der Schlüsselraum zyklisch nach den d Komponenten der Schlüssel unterteilt wird [dB+97]. Ein *kd-tree* läßt sich in $\mathrm{O}(dN \log N)$ Zeit konstruieren und benötigt $\mathrm{O}(dN)$ Speicherplatz. Eine Suchanfrage wird wie eine gewöhnliche Intervallsuche in Blattsuchbäumen bearbeitet und kostet $\mathrm{O}(N^{1-1/d} + K)$ Zeit.

Range trees. Die primäre Struktur eines *range tree* ist ein balancierter Blattsuchbaum (*trie*) bezüglich der letzten Schlüsselkomponente, in welchem die Elemente von P gespeichert sind. Für $d > 1$ ist jedem inneren Knoten v eine sekundäre Struktur zugeordnet, nämlich wiederum ein *range tree* bezüglich der um die letzte Komponente verkürzten Schlüssel, welcher die Elemente im Teilbaum mit Wurzel v speichert [dB+97, Smi94].

Im statischen Fall können alle *tries* perfekt balanciert konstruiert werden. Im dynamischen Fall können Strukturen, die wie Rot-Schwarz- oder AVL-Bäume Rotationen zur Sicherung der Balanciertheit verwenden, nicht eingesetzt werden, da die Korrektur der sekundären Datenstruktur zuviel Aufwand erfordert. Statt ihrer können z. B. $\mathrm{BB}[\alpha]$-Bäume verwendet werden; sie bewahren die Balanciertheit durch partiellen Neuaufbau der Struktur, welcher sich über eine Folge von Einfüge- und Löschoperationen amortisiert [Smi94]. – In beiden Fällen läßt sich der *range tree* in $\mathrm{O}(N \log^{d-1} N)$ Zeit konstruieren und benötigt $\mathrm{O}(N \log^{d-1} N)$ Speicherplatz.

Um in einem *range tree* eine Suchanfrage zu beantworten, wird in der primären Struktur eine Intervallsuche durchgeführt; dies führt auf $\mathrm{O}(\log N)$ viele Unterbäume, deren Blätter in dem vorgegebenen Intervall liegen. In den sekundären Strukturen, die den Wurzeln dieser Unterbäume zugeordnet sind, wird dann eine orthogonale Bereichssuche bezüglich der verbleibenden Komponenten ausgeführt. Eine Anfrage läßt sich auf diese Weise in $\mathrm{O}(\log^d N + K)$ Zeit beantworten.

Es sind Techniken bekannt, die die Anfragezeit um einen $(\log N)$-Faktor verringern (*fractional cascading* [MN90, dB+97], *layering* [Smi94]); gleichzeitig läßt sich für jedes $\varepsilon > 0$ der Speicherplatzbedarf auf $\mathrm{O}(N \log^{d-2+\varepsilon} N)$ senken [Cha90]. Diese Verbesserungen sind jedoch nur für niedrigdimensionale Schlüsselräume relevant.

Die M-Strukturen von Scholten–Overmars. Die Anwendung eines allgemeinen Verfahrens von Scholten und Overmars [SO89] auf das Problem der orthogonalen Bereichssuche führt zu einer Klasse von sogenannten *M-Strukturen*, welche verschiedene Kompromisse zwischen Speicherplatzbedarf und Anfragezeit erlaubt.

Dazu wird eine Funktion $F\colon \mathbf{N} \to \mathbf{N}$ mit $F(i) \leq i$ fixiert. M-Strukturen sind nun wie *range trees* aufgebaut, aber benutzen keine gewöhnlichen Blattsuchbäume (der Ordnung 2), sondern B-Bäume [Knu98] der Ordnung $F(M)$, wobei M die Anzahl der zu speichernden Elemente ist. Jeder innere Knoten des B-Baums legt hierbei seine Kinder in einem balancierten Binärbaum ab; dies erlaubt eine schnelle Intervallsuche.

Wird $F(M)$ vergrößert, so sinkt offensichtlich die Anzahl der sekundären Datenstrukturen und somit der benötigte Speicherplatz, aber die Anfragezeit steigt. Zum Beispiel läßt sich für jede Zahl $\varepsilon > 0$ eine Anfragezeit von $O(N^{\varepsilon})$ bei Verwendung von $O(N)$ Speicherplatz erzielen, indem $F(M) = M^{\delta}$ für ein geeignetes $\delta > 0$ gewählt wird.

Orthogonale Bereichssuche auf Rastern. Für die Komplexität der orthogonalen Bereichssuche sind untere Schranken bekannt [Cha90]. In unserer Anwendung, der Suche nach Reduzierern, besitzt der Schlüsselraum jedoch eine spezielle Gestalt; es treten nur „kurze“, ganzzahlige Tupel auf. Im Fall der primitiven Partitionsidentitäten etwa liefert Lemma 2.3, daß alle zu speichernden Tupel $\mathbf{x}$ der Abschätzung $\|\mathbf{x}\|_1 \leq 2n-1$ genügen. Die Bereichssuche erfolgt somit auf einem *$U \times \cdots \times U$-Raster* mit $U := 4n-1$. Für eine derart eingeschränkte Aufgabe sind die unteren Schranken nicht mehr beweisbar. Tatsächlich gibt es Strukturen mit besseren Komplexitäten; siehe [Wil83, Wil84, Ove88].

Digital tries. Da in unserer Anwendung die Zahl U sehr klein ist, lassen sich die Tupel in einem *digital trie* [Knu98] der Ordnung U speichern. Auf der i-ten Ebene des *trie* erfolgt die U-Wege-Entscheidung hierbei nach der i-ten Koordinate des Tupels. In jedem inneren Knoten werden die Kinder zur direkten Adressierung in einem Feld der Größe höchstens U abgelegt und zusätzlich vorwärtsverkettet; dies ermöglicht eine schnelle Bereichssuche.

Auswertung. In unserem Fall ist der Speicherplatz die für die Wahl der Datenstruktur ausschlaggebende Ressource; siehe Tafel 2.1. In der Implementierung zeigte sich, daß Strukturen wie die *range trees* wegen ihres Speicherplatzbedarfes von $O(N \log^{d-1} N)$ nicht einsetzbar sind. Die besten Ergebnisse wurden mit den *digital tries* erzielt.

Speicherung der feinsten irreduziblen Tupel. Die Menge der irreduziblen Tupel ist keine beliebige Punktmenge; vielmehr besitzt sie eine innere Struktur, die ausgenutzt werden kann, um den Speicheraufwand zu verringern. – In Verallgemeinerung der Situation aus Lemma 2.6 a (leichtes Heben) definieren wir:

Definition 2.15. Seien $\mathbf{v}, \mathbf{w} \in \mathbf{Z}^n \cap \ker \mathbf{n}^\top$. Das Tupel $\mathbf{w}$ heißt eine *elementare Vergröberung* von $\mathbf{v}$, in Zeichen $\mathbf{v} \ll \mathbf{w}$, wenn es eine Indexmenge $I \subseteq \{1,\dots,n\}$ und einen Index $k \in \{1,\dots,n\} \setminus I$ gibt, so daß

$$\begin{aligned} 0 \leq v_i &< w_i && \text{für } i \in I, & \qquad\qquad & w_i < v_i \leq 0 && \text{für } i \in I,\\ 0 \leq w_k &< v_k, && & \text{oder} \qquad & v_k < w_k \leq 0, &&\\ v_i &= w_i && \text{für } i \notin I \cup \{k\} & & v_i = w_i && \text{für } i \notin I \cup \{k\} \end{aligned}$$

gelten. Dadurch wird eine antisymmetrische Relation $\ll$ auf $\mathbf{Z}^n \cap \ker \mathbf{n}^\top$ definiert, deren reflexiv-transitiver Abschluß (eine Halbordnung) mit $\ll^*$ bezeichnet werde.

Lemma 2.16. *Ist* $\mathbf{v} \in \pm P_n$ *und gilt* $\mathbf{v} \ll^* \mathbf{w}$, *so ist auch* $\mathbf{w} \in \pm P_n$.

Beweis. Es genügt, diese Aussage im Fall $\mathbf{v} \ll \mathbf{w}$ zu zeigen. Seien I und k wie in Definition 2.15; o. B. d. A. möge der Fall mit $0 \leq w_k < v_k$ eintreten. Wegen $\mathbf{v}, \mathbf{w} \in \ker \mathbf{n}^\top$ ist $I \neq \varnothing$. Angenommen, es ist $\mathbf{w} = \mathbf{w}^1 + \mathbf{w}^2$ mit $\mathbf{0} \neq \mathbf{w}^1, \mathbf{w}^2 \preceq_{\mathbf{n}^\top} \mathbf{w}$. Sei o. B. d. A. $w_k^1 < w_k$. Dann gelten $w_k^1 \leq v_k$ und $w_i^1 < v_i$ für $i \in I$. Somit ist $\mathbf{w}^1 \preceq_{\mathbf{n}^\top} \mathbf{v}$ und $\mathbf{w}^1 \neq \mathbf{v}$, d. h., $\mathbf{v}$ ist im Widerspruch zur Voraussetzung reduzibel bezüglich $\mathbf{n}^\top$. □

Die Idee ist es nun, nur die bezüglich $\ll^*$ minimalen Vektoren aus P_n, d. h. die „feinsten“ irreduziblen Vektoren, zu speichern; diese Menge wird mit $P_n^{\ll^*}$ bezeichnet.

n	$\#P_n$	$\#P_n^{\ll^*}$	n	$\#P_n$	$\#P_n^{\ll^*}$
3	5	4	12	9 285	2 377
4	15	10	13	18 900	4 235
5	47	26	14	33 269	6 840
6	102	51	15	58 171	11 599
7	276	117	16	99 328	18 514
8	578	216	17	181 514	30 273
9	1 261	427	18	287 239	45 782
10	2 465	759	19	502 116	72 575
11	5 362	1 428	20	775 710	109 848

Tafel 2.2: Speicherung der feinsten irreduziblen Vektoren

Die Suche nach einem Reduzierer für ein Tupel $\mathbf{u}$ in der auf diese Weise repräsentierten halbgeordneten Menge P_n erfolgt dann durch *mehrere* Anfragen zur orthogonalen Bereichssuche in $P_n^{\ll^*}$; dazu werden jene Vergröberungen aufgezählt, die zu einem Reduzierer von $\mathbf{u}$ führen können. – Durch dieses Verfahren wird Speicherplatz auf Kosten einer erhöhten Laufzeit gespart.

2.4 Berechnung von erzeugenden Mengen für Rucksackprobleme

Wir wollen nun für einen beliebigen festen Vektor $\mathbf{a} \in \mathbf{Z}^n$ die minimale erzeugende Menge (Gravermenge) für die Familie $\mathscr{F}_\mathbf{a}$ der Probleme

$$\max\left\{\, \mathbf{c}^\top \mathbf{x} : \mathbf{a}^\top \mathbf{x} = b,\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\ \mathbf{x} \in \mathbf{Z}^n \,\right\} \tag{2.5}$$

mit beliebigen Zielfunktionalen $\mathbf{c}^\top \in \mathbf{R}^{(n)}$, rechten Seiten $b \in \mathbf{Z}$ sowie unteren und oberen Schranken $\mathbf{l} \in (\mathbf{Z} \cup \{-\infty\})^n$, $\mathbf{u} \in (\mathbf{Z} \cup \{+\infty\})^n$ berechnen.

Die bisher behandelte Berechnung der primitiven Partitionsidentitäten stellt einen Spezialfall dieser Aufgabe für $\mathbf{a} = \mathbf{n}$ dar, wie bereits zu Beginn dieses Kapitels festgestellt wurde. Durch Anwendung verschiedener Techniken können wir jedoch die in den vorangegangenen Abschnitten vorgestellten Verfahren zur Lösung der allgemeinen Problemstellung benutzen.

Aggregierung von Variablen. Kennen wir die minimale erzeugende Menge $G_{\bar{\mathbf{a}}}$ für die Familie $\mathscr{F}_{\bar{\mathbf{a}}}$, so können wir für jeden Vektor $\mathbf{a}$, der durch Wiederholung von Einträgen aus $\bar{\mathbf{a}}$ entsteht, die minimale erzeugende Menge $G_{\mathbf{a}}$ für die Familie $\mathscr{F}_{\mathbf{a}}$ berechnen.

Für einen beliebigen Vektor $\mathbf{a} \in \mathbf{Z}^n$ seien

$$I(\mathbf{a}) := \big\{\, |a_i| : i \in \{1,\dots,n\} \,\big\} \setminus \{0\},$$

$\bar{n} := \#I(\mathbf{a})$ und $\bar{\mathbf{a}} \in \mathbf{N}^{\bar{n}}$ der eindeutig bestimmte Vektor mit $I(\bar{\mathbf{a}}) = I(\mathbf{a})$ und $\bar{a}_i < \bar{a}_j$ für $i < j$. Der so definierte Vektor $\bar{\mathbf{a}}$ enthält somit die im Vektor $\mathbf{a}$ auftretenden Beträge von Komponenten. – Weiterhin sei

$$\begin{aligned} M(\mathbf{a}) := \big\{\, \mathbf{x} \in \mathbf{Z}^n : {} & x_i = 0 \text{ für } i \text{ mit } a_i = 0, \\ & x_i \operatorname{sgn} a_i \cdot x_j \operatorname{sgn} a_j \geq 0 \text{ für } i,j \text{ mit } |a_i| = |a_j| \,\big\}; \end{aligned}$$

dies ist die Menge aller Vektoren des $\mathbf{Z}^n$, deren Komponenten auf jeweils den Indexmengen, auf denen $\mathbf{a}$ betragsmäßig konstant ist, verträgliche Vorzeichen besitzen. Schließlich sei die Aggregierungsfunktion $\alpha\colon M(\mathbf{a}) \to \mathbf{Z}^{\bar{n}}$ definiert durch

$$\big(\alpha(\mathbf{x})\big)_j := \sum_{i\colon\, |a_i| = \bar{a}_j} x_i \operatorname{sgn} a_i \quad \text{für } j \in \{1,\dots,\bar{n}\}.$$

Satz 2.17 (Urbaniak). *Unter den Voraussetzungen wie oben gilt*

$$\begin{aligned} G_{\mathbf{a}} = \bigcup_{\bar{\mathbf{x}} \in G_{\bar{\mathbf{a}}}} \overset{-1}{\alpha}(\bar{\mathbf{x}}) \cup \{\, \pm \mathbf{e}_i : a_i = 0 \,\} \\ \cup \big\{\, \mathbf{e}_i \operatorname{sgn} a_i - \mathbf{e}_j \operatorname{sgn} a_j : |a_i| = |a_j| \neq 0 \textit{ mit } i \neq j \,\big\}. \end{aligned}$$

Dies ist eine leichte Verallgemeinerung von Satz 4.1.2 in [Urb98]. □

Löschung von Variablen. Ebenso können wir für jeden Vektor $\mathbf{a}$, der durch Entfernung von Einträgen aus $\bar{\mathbf{a}}$ entsteht, die minimale erzeugende Menge $G_{\mathbf{a}}$ für die Familie $\mathscr{F}_{\mathbf{a}}$ auf einfache Weise berechnen.

Seien $\bar{\mathbf{a}} \in \mathbf{Z}^{\bar{n}}$ und $\mathbf{a} \in \mathbf{Z}^n$, und sei $\pi\colon \{1,\dots,n\} \to \{1,\dots,\bar{n}\}$ eine injektive Abbildung mit $\mathbf{a} = (\bar{a}_{\pi(1)},\dots,\bar{a}_{\pi(n)})^\top$. Für $\mathbf{x} \in \mathbf{Z}^n$ sei $\pi^{-1}(\mathbf{x}) = \bar{\mathbf{x}} \in \mathbf{Z}^{\bar{n}}$ definiert durch

$$\bar{x}_j^k := \begin{cases} x_i^k & \text{für } j \in \operatorname{im}\pi \text{ mit } j = \pi(i), \\ 0 & \text{für } j \notin \operatorname{im}\pi, \end{cases}$$

und für $\bar{\mathbf{x}} \in \mathbf{Z}^{\bar{n}}$ mit $\bar{x}_j = 0$ für $j \notin \operatorname{im}\pi$ sei

$$\pi(\bar{\mathbf{x}}) := (\bar{x}_{\pi(1)},\dots,\bar{x}_{\pi(n)})^\top \in \mathbf{Z}^n.$$

Satz 2.18. *Unter den Voraussetzungen wie oben gilt*

$$G_{\mathbf{a}} = \left\{ \pi(\bar{\mathbf{x}}) : \bar{\mathbf{x}} \in G_{\bar{\mathbf{a}}} \textit{ mit } \bar{x}_j = 0 \textit{ für } j \notin \operatorname{im}\pi \right\}.$$

Beweis. Sei $\bar{\mathbf{x}} \in G_{\bar{\mathbf{a}}}$ mit $\bar{x}_j = 0$ für $j \notin \operatorname{im}\pi$. Sei $\mathbf{x} := \pi(\bar{\mathbf{x}})$. Dann gilt

$$\mathbf{a}^\top \mathbf{x} = \sum_{i=1}^{n} a_i \bar{x}_{\pi(i)} = \sum_{i=1}^{n} \bar{a}_{\pi(i)} \bar{x}_{\pi(i)} = \sum_{j=1}^{\bar{n}} \bar{a}_j \bar{x}_j = \bar{\mathbf{a}}^\top \bar{\mathbf{x}} = 0. \tag{2.6}$$

Angenommen, $\mathbf{x}$ ist reduzibel bezüglich $\mathbf{a}^\top$. Dann gibt es Vektoren $\mathbf{x}^1, \mathbf{x}^2 \in \mathbf{Z}^n \setminus \{\mathbf{0}\}$ mit $\mathbf{x} = \mathbf{x}^1 + \mathbf{x}^2$ und $\mathbf{x}^1, \mathbf{x}^2 \preceq_{\mathbf{a}^\top} \mathbf{x}$. Mit $\bar{\mathbf{x}}^{\mathbf{k}} := \pi^{-1}(\mathbf{x}^k)$ für $k \in \{1,2\}$ gilt dann $\bar{\mathbf{x}} = \bar{\mathbf{x}}^{\mathbf{1}} + \bar{\mathbf{x}}^{\mathbf{2}}$ mit $\mathbf{0} \neq \bar{\mathbf{x}}^1, \bar{\mathbf{x}}^2 \preceq_{\bar{\mathbf{a}}^\top} \bar{\mathbf{x}}$; das ist ein Widerspruch zur Irreduzibilität von $\bar{\mathbf{x}}$ bezüglich $\bar{\mathbf{a}}^\top$. – Somit ist $\mathbf{x} \in G_{\mathbf{a}}$.

Sei umgekehrt $\mathbf{x} \in G_{\mathbf{a}}$, also $\mathbf{x}$ irreduzibel bezüglich $\mathbf{a}^\top$ mit $\mathbf{a}^\top \mathbf{x} = 0$. Für $\bar{\mathbf{x}} := \pi^{-1}(\mathbf{x})$ gilt (2.6), und $\bar{\mathbf{x}}$ ist wiederum irreduzibel bezüglich $\bar{\mathbf{a}}^\top$; somit ist $\bar{\mathbf{x}} \in G_{\bar{\mathbf{a}}}$ mit $\bar{x}_j = 0$ für alle $j \notin \operatorname{im}\pi$. □

Ein rekursives Berechnungsschema. Eine Analyse des Beweises von Satz 2.5 zeigt, daß die Algorithmen 2.2 und 2.12 nach leichten Modifikationen in einer etwas allgemeineren Situation angewandt werden können.

Für eine Menge $I \subseteq \{1,\dots,n\}$ sei P^I die Menge der Tupel $\mathbf{x} \in P_n$ mit $x_i = 0$ für alle $i \notin I$. Ist I die Menge der Komponenten eines Vektors $\mathbf{a}$, so wird P^I nach Satz 2.18 im wesentlichen die Menge $G_{\mathbf{a}}$ sein. – Offensichtlich kann P^I berechnet werden, indem man P_n bestimmt und dann alle Tupel entfernt, die nicht der Bedingung genügen. Erfüllt jedoch I eine bestimmte Voraussetzung, so können wir auch den Urbaniak-Algorithmus direkt zur Berechnung einsetzen:

Satz 2.19. *Sei $n \in \mathbf{N}$. Sei $I \subseteq \{1,\dots,n\}$ eine Menge, die* bezüglich Differenzbildung zu $n+1$ abgeschlossen ist, *d. h., aus $i \in I$ folge $(n+1) - i \in I$. Ersetzt man in Algorithmus 2.12 die Zeile 9 durch*

for all $i \in I \cap \left\{1,\dots,\left\lfloor \frac{n+1}{2} \right\rfloor\right\}$ **do**,

so berechnet das Verfahren die Menge $P^{I \cup \{n+1\}}$, wenn es P^I als Eingabe erhält.

Beweis. Siehe den Beweis von Satz 2.5. □

Dieses Resultat führt zu dem folgenden Schema zur Berechnung der minimalen erzeugenden Menge $G_{\mathbf{a}}$.

Algorithmus 2.20 (Rekursive Berechnung der minimalen erzeugenden Menge).

1: **input** $\mathbf{a} \in \mathbf{Z}^n$;
2: **output** $G_{\mathbf{a}}$;
3: $I \leftarrow I(\mathbf{a}) = \left\{ |a_i| : i \in \{1,\dots,n\} \right\} \setminus \{0\}$;

4: $\langle$Berechne $P^I\rangle$;
5: $G_{\bar{\mathbf{a}}} \leftarrow P^I \cup -P^I$;
6: Konstruiere $G_{\mathbf{a}}$ aus $G_{\bar{\mathbf{a}}}$ gemäß Satz 2.17.

7: $\langle$Berechne $P^I\rangle \equiv$
8:   **if** $\#I = 2$ **then**
9:     Sei $I = \{i, j\}$ mit $i < j$;
10:     $d \leftarrow \gcd(i, j)$;
11:     $P^I \leftarrow \left\{-j/d\,\mathbf{e}_i + i/d\,\mathbf{e}_j\right\}$
12:   **else**
13:     $k \leftarrow \max I$;
14:     $\bar{I} \leftarrow \varnothing$;
15:     **for all** $i \in I \setminus \{k\}$ **do**
16:       $\bar{I} \leftarrow \bar{I} \cup \{i, k-i\}$
17:     **od**;
18:     $\langle$Berechne $P^{\bar{I}}\rangle$;
19:     Konstruiere $P^{\bar{I}\cup\{k\}}$ aus P^I mit Algorithmus 2.12 gemäß Satz 2.19;
20:     $P^I \leftarrow \varnothing$;
21:     **for all** $\mathbf{x} \in P^{\bar{I}\cup\{k\}}$ **do**
22:       **if** $x_i = 0$ für $i \in \bar{I} \setminus I$ **then**
23:         $P^I \leftarrow P^I \cup \{\mathbf{x}\}$
24:       **fi**
25:     **od**
26:   **fi**.

Satz 2.21. *Algorithmus 2.20 ist korrekt.*

Beweis. Das Verfahren zieht sich mittels Satz 2.17 auf die Berechnung der minimalen erzeugenden Menge $G_{\bar{\mathbf{a}}}$ zurück, wobei $\bar{\mathbf{a}}$ lauter verschiedene, positive Einträge besitzt. Diese Menge wird durch die rekursive Prozedur in den Zeilen 7–26 berechnet. Falls nur zwei Einträge vorhanden sind, kann die minimale erzeugende Menge direkt angegeben werden; dies geschieht in den Zeilen 9–11. Anderenfalls wird in den Zeilen 14–17 durch eventuelles Hinzufügen von Einträgen die Voraussetzung von Satz 2.19 gesichert; nach dem rekursiven Abstieg wird der gemäß Satz 2.19 modifizierte Algorithmus 2.12 angewendet und schließlich in den Zeilen 20–25 die nach Satz 2.18 relevante Teilmenge von Vektoren bestimmt. □

Zum Vergleich der Laufzeiten von Algorithmus 2.20 und Algorithmus 1.8 sei ein weiteres Mal auf Tafel 2.1 verwiesen. Enthält $\mathbf{a}$ bereits viele der Zahlen aus der Menge $\left\{1, \ldots, \max_{i=1}^{n} |a_i|\right\}$ als Einträge, so müssen im Verlauf des rekursiven Berechnungsschemas nur wenige Einträge hinzugefügt werden, und Algorithmus 2.20 wird wie im Fall $\mathbf{a} = (1, \ldots, n)^\top$ schneller als Algorithmus 1.8 sein. Müssen hingegen viele Einträge ergänzt werden, so benötigt Algorithmus 2.20 eine größere Laufzeit.

·	·	·	·	·	6	·	8	·	·	·	·	13	·	15	16	17	·	19	·	21	·	·	·	·	26	·	·	·	·	·	32
·	·	·	·	·	6	·	8	·	·	11	·	13	·	15	16	17	·	19	·	21	·	·	24	·	26						
·	2	·	·	5	6	7	8	9	10	11	·	13	·	15	16	17	18	19	20	21	·	·	24								
·	2	3	4	5	6	7	8	9	10	11	·	13	14	15	16	17	18	19	20	21	22										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19													
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17															
⋮																															
1	2	3	4	5	6																										
1	2	3	4	5																											
1	2	3	4																												
1	2	3																													
1	2																														

Abbildung 2.3: Arbeitsweise des rekursiven Berechnungsschemas

3 Gemischt-ganzzahlige Programme

Definition 3.1. Eine Optimierungsaufgabe der Gestalt

$$\text{MIP: } \max\left\{ \mathbf{c}^\top \mathbf{x} + \mathbf{d}^\top \mathbf{y} : \mathbf{A}\mathbf{x} + \mathbf{C}\mathbf{y} = \mathbf{b},\ \mathbf{l} \leq (\mathbf{x},\mathbf{y}) \leq \mathbf{u},\ \mathbf{x} \in \mathbf{Z}^n,\ \mathbf{y} \in \mathbf{R}^d \right\} \tag{3.1}$$

mit den rationalen Matrizen $\mathbf{A} \in \mathbf{Q}^{m\times n}$ und $\mathbf{C} \in \mathbf{Q}^{m\times d}$, der rechten Seite $\mathbf{b} \in \mathbf{Q}^m$, den linearen Zielfunktionalen $\mathbf{c}^\top \in \mathbf{R}^{(n)}$ und $\mathbf{d}^\top \in \mathbf{R}^{(d)}$ sowie den unteren und oberen Schranken $\mathbf{l}$, $\mathbf{u} \in \mathbf{Q}^{n+d}$ heißt *gemischt-ganzzahliges Programm* (mixed integer program).

Während duale (polyedrische) Methoden gleichermaßen für ganzzahlige und gemischt-ganzzahlige Programme verwendet werden können, ist bislang keine Theorie primaler Methoden verfügbar. Wir entwickeln in diesem Kapitel einen primalen Zugang zur Lösung gemischt-ganzzahliger Programme, welcher sich an Augmentierungsverfahren unter Verwendung von Gravermengen für ganzzahlige Programme anlehnt.

3.1 Reduktionsbegriffe

Für Differenzvektoren zweier für ein gemischt-ganzzahliges Programm zulässiger Punkte benötigen wir einen Begriff der „Irreduzibilität“.

Definition 3.2. Seien $\mathbf{A} \in \mathbf{Q}^{m\times n}$ und $\mathbf{C} \in \mathbf{Q}^{m\times d}$.

a. Ein Tupel $(\tilde{\mathbf{x}},\tilde{\mathbf{y}}) \in \mathbf{Z}^n \times \mathbf{R}^d$ *reduziere* ein Tupel $(\mathbf{x},\mathbf{y}) \in \mathbf{Z}^n \times \mathbf{R}^d$ *bezüglich* $\mathbf{A}$ *und* $\mathbf{C}$, in Zeichen $(\tilde{\mathbf{x}},\tilde{\mathbf{y}}) \preceq_{\mathbf{A}|\mathbf{C}} (\mathbf{x},\mathbf{y})$, wenn

$$(\tilde{\mathbf{x}},\tilde{\mathbf{y}},\mathbf{A}\tilde{\mathbf{x}},\mathbf{C}\tilde{\mathbf{y}},\mathbf{A}\tilde{\mathbf{x}}+\mathbf{C}\tilde{\mathbf{y}}) \preceq (\mathbf{x},\mathbf{y},\mathbf{A}\mathbf{x},\mathbf{C}\mathbf{y},\mathbf{A}\mathbf{x}+\mathbf{C}\mathbf{y}) \tag{3.2}$$

gilt.

b. Ein Tupel $(\mathbf{x},\mathbf{y}) \in \mathbf{Z}^n \times \mathbf{R}^d$ heißt *wesentlich reduzibel bezüglich* $\mathbf{A}$ *und* $\mathbf{C}$, wenn es zwei Tupel $(\mathbf{x}^1,\mathbf{y}^1) \in \mathbf{Z}^n \times \mathbf{R}^d$ und $(\mathbf{x}^2,\mathbf{y}^2) \in \mathbf{Z}^n \times \mathbf{R}^d$ gibt mit $(\mathbf{x}^i,\mathbf{y}^i) \preceq_{\mathbf{A}|\mathbf{C}} (\mathbf{x},\mathbf{y})$ und $(\mathbf{x}^i,\mathbf{y}^i) \notin \{\mathbf{0}\} \times \ker \mathbf{C}$ für $i \in \{1,2\}$, so daß $(\mathbf{x}^1,\mathbf{y}^1) + (\mathbf{x}^2,\mathbf{y}^2) = (\mathbf{x},\mathbf{y})$ gilt.

Bemerkung 3.3. Sei $d = 0$ und $\mathbf{C} = \mathbf{0} \in \mathbf{R}^{m\times 0}$. Dann stimmt der hier eingeführte Reduktionsbegriff bezüglich $\mathbf{A}$ und $\mathbf{0}$ mit dem gewöhnlichen Reduktionsbegriff bezüglich $\mathbf{A}$ aus Definition 1.1 überein. Ferner ist ein Tupel wesentlich reduzibel bezüglich $\mathbf{A}$ und $\mathbf{0}$ genau dann, wenn es reduzibel bezüglich $\mathbf{A}$ ist.

Definition 3.2 wurde so gewählt, um unter anderem die aus Bemerkung 1.5 bekannte Eigenschaft der Zerlegbarkeit eines jeden Vektors in *endlich viele* „irreduzible“ (nicht wesentlich reduzible) Vektoren zu erhalten. Allerdings läßt sich die Endlichkeit nur für Vektoren aus $\ker(\mathbf{A}|\mathbf{C})$ sichern.

Lemma 3.4. *Für jedes Tupel* $(\mathbf{x},\mathbf{y}) \in \mathbf{Z}^n \times \mathbf{R}^d$ *mit* $\mathbf{A}\mathbf{x}+\mathbf{C}\mathbf{y}=\mathbf{0}$ *gibt es eine Zahl* $k \in \mathbf{N}_0$ *und Tupel* $(\mathbf{x}^1,\mathbf{y}^1),\dots,(\mathbf{x}^k,\mathbf{y}^k) \in \mathbf{Z}^n \times \mathbf{R}^d$ *mit* $(\mathbf{x}^i,\mathbf{y}^i) \preceq_{\mathbf{A}|\mathbf{C}} (\mathbf{x},\mathbf{y})$ *für* $i \in \{1,\dots,k\}$*, welche nicht wesentlich reduzibel bezüglich* $\mathbf{A}$ *und* $\mathbf{C}$ *sind, so daß gilt:*

$$(\mathbf{x},\mathbf{y}) = (\mathbf{x}^1,\mathbf{y}^1)+\dots+(\mathbf{x}^k,\mathbf{y}^k).$$

Beweis. Er erfolgt durch Induktion über $\|\mathbf{x}\|_1 \in \mathbf{N}_0$.

Falls $\|\mathbf{x}\|_1 = 0$, also $\mathbf{x}=\mathbf{0}$, ist, so gilt $\mathbf{C}\mathbf{y} = \mathbf{A}\mathbf{x}+\mathbf{C}\mathbf{y}=\mathbf{0}$. Ist nun $(\tilde{\mathbf{x}},\tilde{\mathbf{y}}) \in \mathbf{Z}^n \times \mathbf{R}^d$ mit $(\tilde{\mathbf{x}},\tilde{\mathbf{y}}) \preceq_{\mathbf{A}|\mathbf{C}} (\mathbf{x},\mathbf{y})$, so gelten insbesondere $\tilde{\mathbf{x}} \preceq \mathbf{x} = \mathbf{0}$ und $\mathbf{C}\tilde{\mathbf{y}} \preceq \mathbf{C}\mathbf{y} = \mathbf{0}$, und somit ist $(\tilde{\mathbf{x}},\tilde{\mathbf{y}}) \in \{\mathbf{0}\} \times \ker \mathbf{C}$. Daher ist $(\mathbf{x},\mathbf{y})$ nicht wesentlich reduzibel.

Sei nun $\nu \in \mathbf{N}$ und die Aussage bereits für alle Tupel $(\mathbf{x},\mathbf{y}) \in \mathbf{Z}^n \times \mathbf{R}^d$ mit $\|\mathbf{x}\|_1 < \nu$ bewiesen. Sei $(\mathbf{x},\mathbf{y}) \in \mathbf{Z}^n \times \mathbf{R}^d$ mit $\|\mathbf{x}\|_1 = \nu$. Falls $(\mathbf{x},\mathbf{y})$ nicht wesentlich reduzibel ist, so ist nichts zu zeigen. Seien also $(\mathbf{x}^{\mathrm{I}},\mathbf{y}^{\mathrm{I}})$ und $(\mathbf{x}^{\mathrm{II}},\mathbf{y}^{\mathrm{II}})$ zwei Tupel wie in Definition 3.2 b. Angenommen, o. B. d. A. ist $\mathbf{x}^{\mathrm{I}} = \mathbf{0}$; dann ist $\mathbf{y}^{\mathrm{I}} \notin \ker \mathbf{C}$. Es gilt aber $\mathbf{C}\mathbf{y}^{\mathrm{I}} = \mathbf{A}\mathbf{x}^{\mathrm{I}} + \mathbf{C}\mathbf{y}^{\mathrm{I}} \preceq \mathbf{A}\mathbf{x}+\mathbf{C}\mathbf{y} = \mathbf{0}$. – Somit sind $\mathbf{x}^{\mathrm{I}}$, $\mathbf{x}^{\mathrm{II}} \neq \mathbf{0}$, und damit gilt $\|\mathbf{x}^i\|_1 < \|\mathbf{x}\|_1 = \nu$ für $i \in \{\mathrm{I},\mathrm{II}\}$. Nach Induktionsvoraussetzung gibt es dann Zahlen k_{I} und k_{II} sowie Tupel $(\mathbf{x}^1,\mathbf{y}^1),\dots,(\mathbf{x}^{k_{\mathrm{I}}},\mathbf{y}^{k_{\mathrm{I}}}) \in \mathbf{Z}^n \times \mathbf{R}^d$ und $(\mathbf{x}^{k_{\mathrm{I}}+1},\mathbf{y}^{k_{\mathrm{I}}+1}),\dots,(\mathbf{x}^{k_{\mathrm{I}}+k_{\mathrm{II}}},\mathbf{y}^{k_{\mathrm{I}}+k_{\mathrm{II}}}) \in \mathbf{Z}^n \times \mathbf{R}^d$ mit

$$\begin{aligned}(\mathbf{x}^i,\mathbf{y}^i) &\preceq_{\mathbf{A}|\mathbf{C}} \begin{cases}(\mathbf{x}^{\mathrm{I}},\mathbf{y}^{\mathrm{I}}) & \text{für } i \in \{1,\dots,k_{\mathrm{I}}\},\\ (\mathbf{x}^{\mathrm{II}},\mathbf{y}^{\mathrm{II}}) & \text{für } i \in \{k_{\mathrm{I}}+1,\dots,k_{\mathrm{I}}+k_{\mathrm{II}}\}\end{cases}\\ &\preceq_{\mathbf{A}|\mathbf{C}} (\mathbf{x},\mathbf{y}),\end{aligned}$$

welche die Bedingungen

$$\begin{aligned}(\mathbf{x}^{\mathrm{I}},\mathbf{y}^{\mathrm{I}}) &= (\mathbf{x}^1,\mathbf{y}^1)+\dots+(\mathbf{x}^{k_{\mathrm{I}}},\mathbf{y}^{k_{\mathrm{I}}}),\\ (\mathbf{x}^{\mathrm{II}},\mathbf{y}^{\mathrm{II}}) &= (\mathbf{x}^{k_{\mathrm{I}}+1},\mathbf{y}^{k_{\mathrm{I}}+1})+\dots+(\mathbf{x}^{k_{\mathrm{I}}+k_{\mathrm{II}}},\mathbf{y}^{k_{\mathrm{I}}+k_{\mathrm{II}}})\end{aligned}$$

erfüllen. Mit $k := k_{\mathrm{I}} + k_{\mathrm{II}}$ folgt die Behauptung. □

Die nicht wesentlich reduziblen Tupel werden der Gegenstand der Betrachtungen in den nächsten Abschnitten sein.

3.2 Die Polyeder des kontinuierlichen Anteils

Hat die Matrix $\mathbf{C}$ einen nichttrivialen Kern, so können wir nicht erwarten, eine *endliche* Menge nicht wesentlich reduzibler Tupel zu erhalten. Um wenigstens eine *endliche Beschreibung* zu erhalten, betrachten wir zu jedem diskreten Tupel $\mathbf{x} \in \mathbf{Z}^n$ das Polyeder der kontinuierlichen Tupel $\mathbf{y} \in \mathbf{R}^d$ in einem festen Orthanten, welche $\mathbf{x}$ zu einem Element des Kerns von $(\mathbf{A}|\mathbf{C})$ ergänzen.

Zunächst geben wir einige wohlbekannte Strukturresultate für Polyeder wieder.

Satz 3.5 (Minkowski–Steinitz–Weyl–Motzkin). *Eine Menge $P \subseteq \mathbf{R}^d$ ist genau dann ein konvexes Polyeder, wenn $P = P' + \mathscr{C}$ mit einem Polytop P' und einem konvexen polyedrischen Kegel $\mathscr{C}$ gilt.*

Siehe z. B. [Sch86, Korollar 7.1 b]. □

Satz 3.6. *Sei $\varnothing \neq P \subset \mathbf{R}^d$ ein* spitzes *konvexes Polyeder, d. h., P enthalte keine Gerade. Dann hat P eine eindeutige minimale Darstellung der Gestalt*

$$P = \operatorname{conv}\left\{\mathbf{v}^1, \ldots, \mathbf{v}^r\right\} + \operatorname{cone}\left\{\mathbf{r}^1, \ldots, \mathbf{r}^s\right\}. \tag{3.3}$$

Insbesondere gibt es genau ein inklusionsminimales Polytop P' und genau einen konvexen polyedrischen Kegel $\mathscr{C}$ mit $P = P' + \mathscr{C}$. Dabei ist $\mathscr{C}$ der charakteristische Kegel (Rezessionskegel) *von P,*

$$\mathscr{C} = \left\{\mathbf{t} \in \mathbf{R}^d : \mathbf{y} + \mathbf{t} \in P \text{ für alle } \mathbf{y} \in P\right\}. \tag{3.4}$$

Die Punkte $\mathbf{v}^1, \ldots, \mathbf{v}^r$ sind die Ecken von P (und von P'), und die $\mathbf{r}^1, \ldots, \mathbf{r}^s$ sind Repräsentanten der Extremalstrahlen von $\mathscr{C}$.

Ist $P = \{\mathbf{y} : \mathbf{C}\mathbf{y} \leq \mathbf{f}\}$, so ist $\mathscr{C} = \{\mathbf{y} : \mathbf{C}\mathbf{y} \leq \mathbf{0}\}$.

Siehe z. B. [Sch86, Abschnitte 8.2 und 8.9]. □

Mit diesen Resultaten sind wir in der Lage, das Polyeder des kontinuierlichen Anteils und seine Zerlegung in ein Polytop und einen Kegel einzuführen:

Definition 3.7. Zu $\mathbf{x} \in \mathbf{Z}^n$ und $\varepsilon \in \{\pm 1\}^d$ sei

$$P_\varepsilon(\mathbf{x}) := \{\mathbf{y} \in \mathscr{O}_\varepsilon : \mathbf{A}\mathbf{x} + \mathbf{C}\mathbf{y} = \mathbf{0}\} \subseteq \mathbf{R}^d, \tag{3.5}$$

ein spitzes konvexes Polyeder. Weiterhin seien $P'_\varepsilon(\mathbf{x}) := P'$ das Polytop und $\mathscr{C}_\varepsilon(\mathbf{x}) := \mathscr{C}$ der konvexe polyedrische Kegel mit $P_\varepsilon(\mathbf{x}) = P' + \mathscr{C}$ gemäß Satz 3.6.

Lemma 3.8. *Es gilt $\mathscr{C}_\varepsilon(\mathbf{x}) = \ker \mathbf{C} \cap \mathscr{O}_\varepsilon$. Insbesondere ist $\mathscr{C}_\varepsilon(\mathbf{x})$ unabhängig von $\mathbf{x}$ und wird daher als $\mathscr{C}_\varepsilon$ notiert.*

Beweis. Es ist

$$P_\varepsilon(\mathbf{x}) = \left\{\mathbf{y} \in \mathbf{R}^d : \mathbf{C}\mathbf{y} \leq -\mathbf{A}\mathbf{x},\ -\mathbf{C}\mathbf{y} \leq \mathbf{A}\mathbf{x},\ -\varepsilon_i y_i \leq 0 \text{ für } 1 \leq i \leq d\right\};$$

nach Satz 3.6 gilt dann

$$\begin{aligned}\mathscr{C}_\varepsilon(\mathbf{x}) &= \left\{\mathbf{y} \in \mathbf{R}^d : \mathbf{C}\mathbf{y} \leq \mathbf{0},\ -\mathbf{C}\mathbf{y} \leq \mathbf{0},\ -\varepsilon_i y_i \leq 0 \text{ für } 1 \leq i \leq d\right\} \\ &= \ker \mathbf{C} \cap \mathscr{O}_\varepsilon.\end{aligned}$$

□

Es zeigt sich nun, daß wir uns auf die Betrachtung der Punkte im Polytop $P'_\varepsilon(\mathbf{x})$ zurückziehen können:

Lemma 3.9. *Seien $\mathbf{x} \in \mathbf{Z}^n$ und $\varepsilon \in \{\pm 1\}^d$. Ist $\mathbf{y} \in P_\varepsilon(\mathbf{x})$, so gibt es ein $\mathbf{y}' \in P'_\varepsilon(\mathbf{x})$ mit $(\mathbf{x}, \mathbf{y}') \preceq_{\mathbf{A}|\mathbf{C}} (\mathbf{x}, \mathbf{y})$. Ist zudem $(\mathbf{x}, \mathbf{y})$ nicht wesentlich reduzibel bezüglich $\mathbf{A}$ und $\mathbf{C}$, so auch $(\mathbf{x}, \mathbf{y}')$.*

Beweis. Da $P_\varepsilon(\mathbf{x}) = P'_\varepsilon(\mathbf{x}) + (\ker \mathbf{C} \cap \mathscr{O}_\varepsilon)$ gilt, gibt es ein $\mathbf{y}' \in P'_\varepsilon(\mathbf{x})$ und ein $\mathbf{t} \in \ker \mathbf{C} \cap \mathscr{O}_\varepsilon$ mit $\mathbf{y} = \mathbf{y}' + \mathbf{t}$. Wegen $\mathbf{y}$, $\mathbf{y}'$, $\mathbf{t} \in \mathscr{O}_\varepsilon$ gilt $\mathbf{y}' \preceq \mathbf{y}$. Weiterhin ist $\mathbf{C}\mathbf{y}' = \mathbf{C}(\mathbf{y} - \mathbf{t}) = \mathbf{C}\mathbf{y}$; damit werden die übrigen Reduktionsaussagen erfüllt.

Sei nun angenommen, daß $(\mathbf{x}, \mathbf{y}')$ wesentlich reduzibel bezüglich $\mathbf{A}$ und $\mathbf{C}$ ist; seien $(\mathbf{x}^1, \mathbf{y}^1)$ und $(\mathbf{x}^2, \mathbf{y}^2)$ zwei Tupel wie in Definition 3.2 b. Wegen $\mathbf{y}^1 \preceq \mathbf{y}'$ und $\mathbf{t}$, $\mathbf{y}^1 \in \mathscr{O}_\varepsilon$ gilt $\mathbf{y}^1 + \mathbf{t} \preceq \mathbf{y}' + \mathbf{t} = \mathbf{y}$; außerdem ist $\mathbf{y}^2 \preceq \mathbf{y}$. Somit gelten

$$(\mathbf{x}^1, \mathbf{y}^1 + \mathbf{t}),\ (\mathbf{x}^2, \mathbf{y}^2) \preceq_{\mathbf{A}|\mathbf{C}} (\mathbf{x}, \mathbf{y}),$$

da die übrigen Reduktionsaussagen trivialerweise erfüllt sind. Da $\mathbf{t} \in \ker \mathbf{C}$ und $(\mathbf{x}^1, \mathbf{y}^1) \notin \{\mathbf{0}\} \times \ker \mathbf{C}$ sind, ist $(\mathbf{x}^1, \mathbf{y}^1 + \mathbf{t}) \notin \{\mathbf{0}\} \times \ker \mathbf{C}$. Somit ist auch $(\mathbf{x}, \mathbf{y})$ wesentlich reduzibel bezüglich $\mathbf{A}$ und $\mathbf{C}$. □

Die Umkehrung der zweiten Aussage in Lemma 3.9 gilt jedoch i. a. nicht. – Lemma 3.9 erlaubt es uns, eine Variante der Zerlegungseigenschaft aus Lemma 3.4 zu formulieren:

Folgerung 3.10. *Sei $(\mathbf{x}, \mathbf{y}) \in \mathbf{Z}^n \times \mathbf{R}^d$ mit $\mathbf{A}\mathbf{x} + \mathbf{C}\mathbf{y} = \mathbf{0}$. Sei $\varepsilon \in \{\pm 1\}^d$ mit $\mathbf{y} \in \mathscr{O}_\varepsilon$. Dann gibt es eine Zahl $k \in \mathbf{N}_0$ und Tupel $(\mathbf{x}^1, \mathbf{y}^1), \dots, (\mathbf{x}^k, \mathbf{y}^k) \in \mathbf{Z}^n \times \mathbf{R}^d$ mit $\mathbf{y}^i \in P'_\varepsilon(\mathbf{x}^i)$ und $(\mathbf{x}^i, \mathbf{y}^i) \preceq_{\mathbf{A}|\mathbf{C}} (\mathbf{x}, \mathbf{y})$ für $i \in \{1, \dots, k\}$, welche nicht wesentlich reduzibel bezüglich $\mathbf{A}$ und $\mathbf{C}$ sind, sowie ein Tupel $(\mathbf{x}^0, \mathbf{y}^0) \in \{\mathbf{0}\} \times (\ker \mathbf{C} \cap \mathscr{O}_\varepsilon)$, so daß gilt:*

$$(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^0, \mathbf{y}^0) + (\mathbf{x}^1, \mathbf{y}^1) + \dots + (\mathbf{x}^k, \mathbf{y}^k).$$

Beweis. Dies folgt aus Lemma 3.4 und Lemma 3.9. □

Wir werden im folgenden Beziehungen zwischen den Polyedern $P_\varepsilon(\mathbf{x})$ für verschiedene Parameter $\mathbf{x}$ benötigen. Eine einfache Inklusionsbeziehung können wir bereits jetzt formulieren:

Lemma 3.11. *Für $\mathbf{x}^1$, $\mathbf{x}^2 \in \mathbf{Z}^n$ und $\varepsilon \in \{\pm 1\}^d$ gilt*

$$P_\varepsilon(\mathbf{x}^1) + P_\varepsilon(\mathbf{x}^2) \subseteq P_\varepsilon(\mathbf{x}^1 + \mathbf{x}^2).$$

Beweis. Sei $\mathbf{y} \in P_\varepsilon(\mathbf{x}^1) + P_\varepsilon(\mathbf{x}^2)$. Dann gibt es $\mathbf{y}^i \in P_\varepsilon(\mathbf{x}^i)$ für $i \in \{1, 2\}$ mit $\mathbf{y} = \mathbf{y}^1 + \mathbf{y}^2$. Es gilt $\mathbf{A}\mathbf{x}^i + \mathbf{C}\mathbf{y}^i = \mathbf{0}$ für $i \in \{1, 2\}$ und daher auch $\mathbf{A}\mathbf{x} + \mathbf{C}\mathbf{y} = \mathbf{0}$. Ferner ist $\mathbf{y} \in \mathscr{O}_\varepsilon$, und somit ist $\mathbf{y} \in P_\varepsilon(\mathbf{x}^1 + \mathbf{x}^2)$. □

3.3 Die Menge der nicht wesentlich reduziblen Tupel

Eine Klasse von nicht wesentlich reduziblen Tupeln $(\mathbf{x},\mathbf{y})$ erhält man durch das „Erweitern“ irreduzibler $\mathbf{x}$:

Lemma 3.12. *Sei $\mathbf{x} \in \mathbf{Z}^n$ irreduzibel bezüglich der Matrix $\mathbf{A}$. Dann ist das Tupel $(\mathbf{x},\mathbf{y})$ für jedes $\mathbf{y} \in \mathbf{R}^d$ mit $\mathbf{A}\mathbf{x}+\mathbf{C}\mathbf{y}=\mathbf{0}$ nicht wesentlich reduzibel bezüglich $\mathbf{A}$ und $\mathbf{C}$.*

Beweis. Seien $(\mathbf{x}^1,\mathbf{y}^1)$ und $(\mathbf{x}^2,\mathbf{y}^2)$ zwei Tupel wie in Definition 3.2 b. Dann gilt insbesondere $\mathbf{x}=\mathbf{x}^1+\mathbf{x}^2$ mit $\mathbf{x}^1$, $\mathbf{x}^2 \preceq_{\mathbf{A}} \mathbf{x}$. Angenommen, es ist $\mathbf{x}^i=\mathbf{0}$ für ein $i \in \{1,2\}$. Dann gilt $\mathbf{A}\mathbf{x}^i=\mathbf{0}$, also $\mathbf{C}\mathbf{y}^i=\mathbf{A}\mathbf{x}^i+\mathbf{C}\mathbf{y}^i \preceq \mathbf{A}\mathbf{x}+\mathbf{C}\mathbf{y}=\mathbf{0}$, und damit ist bereits $(\mathbf{x}^i,\mathbf{y}^i) \in \{\mathbf{0}\} \times \ker\mathbf{C}$; das ist ein Widerspruch zur Wahl der Tupel $(\mathbf{x}^i,\mathbf{y}^i)$. – Somit ist $\mathbf{x}$ reduzibel bezüglich $\mathbf{A}$. □

Diese Aussage ist offensichtlich nur für diejenigen Vektoren $\mathbf{x} \in \mathbf{Z}^n$ interessant, für die $\mathbf{A}\mathbf{x} \in \operatorname{im}\mathbf{C}$ gilt. Es ist aber i. a. nicht gesichert, daß es überhaupt einen bezüglich $\mathbf{A}$ irreduziblen Vektor mit dieser Eigenschaft gibt. Deshalb wollen wir mit einer anderen Menge von Vektoren arbeiten. Wir benutzen dazu den folgenden Satz aus der linearen Algebra:

Satz 3.13 (Fredholm). *Für $\mathbf{C} \in \mathbf{R}^{m\times d}$ gilt $\ker\mathbf{C}^\top = (\operatorname{im}\mathbf{C})^\perp$.* □

Sei nun $\{\mathbf{p}^1,\dots,\mathbf{p}^r\} \subset \mathbf{Q}^m$ eine Basis von $\ker\mathbf{C}^\top$, und sei

$$\mathbf{A}' := \begin{pmatrix} (\mathbf{p}^1)^\top \\ \vdots \\ (\mathbf{p}^r)^\top \end{pmatrix} \mathbf{A} \in \mathbf{Q}^{r\times n};$$

dann gilt $\mathbf{A}\mathbf{x} \in \operatorname{im}\mathbf{C}$ genau dann, wenn $\mathbf{x} \in \ker\mathbf{A}'$ ist. – Es zeigt sich, daß auch das Erweitern von bezüglich $\mathbf{A}'$ irreduziblen Vektoren $\mathbf{x}$ zu nicht wesentlich reduziblen Tupeln $(\mathbf{x},\mathbf{y})$ führt:

Lemma 3.14. *Sei $\mathbf{x} \in \mathbf{Z}^n \cap \ker\mathbf{A}'$ irreduzibel bezüglich $\mathbf{A}'$. Dann ist das Tupel $(\mathbf{x},\mathbf{y})$ für jedes $\mathbf{y} \in \mathbf{R}^d$ mit $\mathbf{A}\mathbf{x}+\mathbf{C}\mathbf{y}=\mathbf{0}$ nicht wesentlich reduzibel bezüglich $\mathbf{A}$ und $\mathbf{C}$.*

Beweis. Seien $(\mathbf{x}^1,\mathbf{y}^1)$ und $(\mathbf{x}^2,\mathbf{y}^2)$ zwei Tupel wie in Definition 3.2 b. Wie im Beweis von Lemma 3.12 gilt dann $\mathbf{0} \neq \mathbf{x}^1, \mathbf{x}^2 \preceq \mathbf{x}$. Wegen $\mathbf{A}\mathbf{x}^i+\mathbf{C}\mathbf{y}^i \preceq \mathbf{A}\mathbf{x}+\mathbf{C}\mathbf{y}=\mathbf{0}$ gilt zudem $\mathbf{A}\mathbf{x}^i \in \operatorname{im}\mathbf{C}$, also $\mathbf{A}'\mathbf{x}^i=\mathbf{0} \preceq \mathbf{A}'\mathbf{x}$, für $i \in \{1,2\}$. Somit ist $\mathbf{x}$ reduzibel bezüglich $\mathbf{A}'$. □

Die bezüglich $\mathbf{A}'$ irreduziblen Vektoren $\mathbf{x} \in \mathbf{Z}^n \cap \ker\mathbf{A}'$ können mit dem Hilbertbasis-Algorithmus 1.8 berechnet werden.

Lemma 3.15. *Seien $\mathbf{x}^1$, $\mathbf{x}^2 \in \mathbf{Z}^n \setminus \{\mathbf{0}\}$ zwei Vektoren mit $(\mathbf{x}^1,\mathbf{A}\mathbf{x}^1)$, $(\mathbf{x}^2,\mathbf{A}\mathbf{x}^2) \in \mathcal{O}_\chi$ für ein $\chi \in \{\pm1\}^{n+m}$. Dann sind alle Tupel $(\mathbf{x},\mathbf{y})$ mit $\mathbf{x}=\mathbf{x}^1+\mathbf{x}^2$ und $\mathbf{y} \in P_\varepsilon(\mathbf{x}^1)+P_\varepsilon(\mathbf{x}^2)$ für beliebige $\varepsilon \in \{\pm1\}^d$ wesentlich reduzibel bezüglich $\mathbf{A}$ und $\mathbf{C}$.*

Beweis. Ist $\mathbf{y} \in P_\varepsilon(\mathbf{x}^1) + P_\varepsilon(\mathbf{x}^2)$, so gibt es $\mathbf{y}^i \in P_\varepsilon(\mathbf{x}^i)$ für $i \in \{1,2\}$ mit $\mathbf{y} = \mathbf{y}^1 + \mathbf{y}^2$. Dann gelten bereits $(\mathbf{x}^i, \mathbf{y}^i) \preceq_{\mathbf{A}|\mathbf{C}} (\mathbf{x}, \mathbf{y})$ und $(\mathbf{x}^i, \mathbf{y}^i) \notin \{\mathbf{0}\} \times \ker \mathbf{C}$ für $i \in \{1,2\}$. □

Definition 3.16. **a.** $\mathscr{I}_\mathbf{C} \subseteq 2^{\{1,\dots,m+d\}}$ sei das Matroid der Indexmengen linear unabhängiger Zeilen von $\bar{\mathbf{C}} := \binom{\mathbf{id}}{\mathbf{C}} \in \mathbf{R}^{(m+d)\times d}$. Weiter sei $\mathscr{B}_\mathbf{C}$ das Basissystem von $\mathscr{I}_\mathbf{C}$.

b. Für $I \in \mathscr{B}_\mathbf{C}$ seien $\mathbf{C}_I \in \mathbf{R}^{d\times d}$ die von den Zeilen mit Indizes in I gebildete Teilmatrix von $\bar{\mathbf{C}}$ und $\mathbf{b}_I \in \mathbf{R}^d$ das von den Einträgen von $\bar{\mathbf{b}} := \binom{\mathbf{0}}{\mathbf{b}} \in \mathbf{R}^{m+d}$ mit Indizes in I gebildete Tupel.

Satz 3.17. *Für $I \in \mathscr{B}_\mathbf{C}$ und $\mathbf{x} \in \mathbf{Z}^n$ mit $\mathbf{b} := -\mathbf{A}\mathbf{x} \in \operatorname{im}\mathbf{C}$ sei $\mathbf{y}_I(\mathbf{x}) := \mathbf{y} := \mathbf{C}_I^{-1}\mathbf{b}_I$ der eindeutig bestimmte Vektor mit $\mathbf{C}\mathbf{y} = \mathbf{b}$ und $y_i = 0$ für $i \in I \cap \{1,\dots,d\}$. Dann ist die Menge der Ecken von $P'_\varepsilon(\mathbf{x})$ gegeben durch*

$$V_\varepsilon(\mathbf{x}) = \{\, \mathbf{y}_I(\mathbf{x}) : I \in \mathscr{B}_\varepsilon(\mathbf{x}) \,\} \quad \textit{mit} \quad \mathscr{B}_\varepsilon(\mathbf{x}) := \{\, I \in \mathscr{B} : \mathbf{y}_I(\mathbf{x}) \in \mathscr{O}_\varepsilon \,\}.$$

Siehe [Sch86, Abschnitt 8.5]. □

Im folgenden seien $\mathscr{B}_\mathbf{C} = \{I_1, \dots, I_M\}$,

$$\tilde{\mathbf{M}} := \begin{pmatrix} \mathbf{C}_{I_1}^{-1}\mathbf{A} \\ \vdots \\ \mathbf{C}_{I_M}^{-1}\mathbf{A} \end{pmatrix} \in \mathbf{Q}^{mM\times n}$$

und $z \in \mathbf{N}$ derart gewählt, daß $\mathbf{M} := z\tilde{\mathbf{M}} \in \mathbf{Z}^{mM\times n}$ gilt.

Satz 3.18. *Seien $\varepsilon \in \{\pm 1\}^d$ und $\mathbf{x}^1$, $\mathbf{x}^2 \in \mathbf{Z}^n$.*

a. *Ist $\mathscr{B}_\varepsilon(\mathbf{x}^1) \cap \mathscr{B}_\varepsilon(\mathbf{x}^2) \supseteq \mathscr{B}_\varepsilon(\mathbf{x}^1 + \mathbf{x}^2)$, so gelten*

$$P'_\varepsilon(\mathbf{x}^1) + P'_\varepsilon(\mathbf{x}^2) = P'_\varepsilon(\mathbf{x}^1 + \mathbf{x}^2), \tag{3.6a}$$

$$P_\varepsilon(\mathbf{x}^1) + P_\varepsilon(\mathbf{x}^2) = P_\varepsilon(\mathbf{x}^1 + \mathbf{x}^2). \tag{3.6b}$$

b. *Insbesondere gilt* (3.6)*, wenn $\mathbf{M}\mathbf{x}^1$ und $\mathbf{M}\mathbf{x}^2$ im gleichen Orthanten liegen.*

Beweis. Wegen Lemma 3.11 ist nur die Inklusion „$\supseteq$“ zu zeigen. Sei $\mathbf{x} := \mathbf{x}^1 + \mathbf{x}^2$, und sei $\mathbf{y} \in V_\varepsilon(\mathbf{x})$ eine Ecke von $P'_\varepsilon(\mathbf{x})$. Nach Satz 3.17 ist $\mathbf{y} = \mathbf{y}_I(\mathbf{x})$ für ein $I \in \mathscr{B}_\varepsilon(\mathbf{x}^1)$. Ist die Voraussetzung in Punkt a erfüllt, so ist $\mathbf{y}^i := \mathbf{y}_I(\mathbf{x}^i) \in V_\varepsilon(\mathbf{x}^i)$ für $i \in \{1,2\}$. – Somit gilt $V_\varepsilon(\mathbf{x}) \subseteq P'_\varepsilon(\mathbf{x}^1) + P'_\varepsilon(\mathbf{x}^2)$; daraus folgt $P'_\varepsilon(\mathbf{x}) = \operatorname{conv} V_\varepsilon(\mathbf{x}) \subseteq P'_\varepsilon(\mathbf{x}^1) + P'_\varepsilon(\mathbf{x}^2)$.

Liegen $\mathbf{M}\mathbf{x}^1$ und $\mathbf{M}\mathbf{x}^2$ im gleichen Orthanten $\mathscr{O}_\eta$, so ist auch $\mathbf{M}\mathbf{x} \in \mathscr{O}_\eta$, und die Indexmengen $\mathscr{B}_\varepsilon(\mathbf{x}^1)$, $\mathscr{B}_\varepsilon(\mathbf{x}^2)$ und $\mathscr{B}_\varepsilon(\mathbf{x}^1 + \mathbf{x}^2)$ stimmen alle überein. Damit ist die Voraussetzung in Punkt a erfüllt. □

Wir benutzen nun eine als „Gordan-Lemma“ bekannte geometrische Version des Gordan–Dickson-Lemmas:

Satz 3.19 (Gordan–Dickson). *Ist $\varnothing \neq V \subseteq \mathbf{N}_0^n$, so gibt es eine eindeutig bestimmte endliche minimale Menge $\{\mathbf{v}^1,\ldots,\mathbf{v}^l\} \subseteq V$ derart, daß es für jeden Punkt $\mathbf{v} \in V$ einen Index $i \in \{1,\ldots,l\}$ mit $\mathbf{v}^i \leq \mathbf{v}$ gibt.*

Siehe z. B. [Urb98, HMW]. □

Angewendet auf die Menge

$$S := \{ (\mathbf{x}^+, \mathbf{x}^-, (\mathbf{M}\mathbf{x})^+, (\mathbf{M}\mathbf{x})^-) : \mathbf{0} \neq \mathbf{x} \in \mathbf{Z}^n \cap \ker \mathbf{A}' \}$$

liefert dieser Satz unter Beachtung von Bemerkung 1.3 b:

Folgerung 3.20. *Es gibt eine endliche Menge $S_0 \subseteq \mathbf{Z}^n \cap \ker \mathbf{A}' \setminus \{\mathbf{0}\}$, die für jeden Vektor $\mathbf{x} \in \mathbf{Z}^n \cap \ker \mathbf{A}' \setminus \{\mathbf{0}\}$ einen Reduzierer $\tilde{\mathbf{x}} \preceq_{\mathbf{M}} \mathbf{x}$ mit $\tilde{\mathbf{x}} \neq \mathbf{0}$ enthält.* □

Satz 3.21. *Ist $\mathbf{0} \neq \mathbf{x} \in \mathbf{Z}^n \cap \ker \mathbf{A}'$ mit $\mathbf{x} \notin S_0$, so sind alle Tupel $(\mathbf{x},\mathbf{y})$ mit $\mathbf{y} \in P_\varepsilon(\mathbf{x})$ für beliebige $\varepsilon \in \{\pm 1\}^d$ wesentlich reduzibel bezüglich $\mathbf{A}$ und $\mathbf{C}$.*

Beweis. Sei $\mathbf{x}^1 \in S_0$ mit $\mathbf{x}^1 \preceq_{\mathbf{M}} \mathbf{x}$. Mit $\mathbf{x}^2 := \mathbf{x} - \mathbf{x}^1$ gilt auch $\mathbf{x}^2 \preceq_{\mathbf{M}} \mathbf{x}$. Insbesondere liegen $\mathbf{M}\mathbf{x}^1$, $\mathbf{M}\mathbf{x}^2$ und $\mathbf{M}\mathbf{x}$ im gleichen Orthanten; nach Satz 3.18 b gilt daher

$$P_\varepsilon(\mathbf{x}^1) + P_\varepsilon(\mathbf{x}^2) = P_\varepsilon(\mathbf{x}) \tag{3.7}$$

für alle $\varepsilon \in \{\pm 1\}^d$. – Für $I := \{1,\ldots,d\} \in \mathscr{B}_{\mathbf{C}}$ gilt $\mathbf{C}_I = \mathbf{id}$, also $\mathbf{C}_I^{-1}\mathbf{A} = \mathbf{A}$; daher folgt aus $\mathbf{x}^1$, $\mathbf{x}^2 \preceq_{\mathbf{M}} \mathbf{x}$, daß $(\mathbf{x}^1, \mathbf{A}\mathbf{x}^1)$ und $(\mathbf{x}^2, \mathbf{A}\mathbf{x}^2)$ im gleichen Orthanten liegen. Lemma 3.15 liefert dann die Aussage des Satzes. □

3.4 Augmentieren mit Testmengen vom Graver-Typ

Definition 3.22. $\mathrm{MIP}_{\mathbf{A}|\mathbf{C}}$ sei die Familie aller gemischt-ganzzahligen Programme der Gestalt (3.1) mit fest gewählten Matrizen $\mathbf{A}$ und $\mathbf{C}$.

Definition 3.23. Für $\mathbf{x} \in \mathbf{Z}^n$ und $\varepsilon \in \{\pm 1\}^d$ sei $Q_\varepsilon(\mathbf{x})$ die Menge der $\mathbf{y} \in P'_\varepsilon(\mathbf{x})$, so daß $(\mathbf{x},\mathbf{y})$ nicht wesentlich reduzibel bezüglich $\mathbf{A}$ und $\mathbf{C}$ ist, und $\bar{Q}_\varepsilon(\mathbf{x})$ ein Polytop mit

$$Q_\varepsilon(\mathbf{x}) \subseteq \bar{Q}_\varepsilon(\mathbf{x}) \subseteq P'_\varepsilon(\mathbf{x}).$$

Die *Gravermenge H* für die Familie $\mathrm{MIP}_{\mathbf{A}|\mathbf{C}}$ ist die Menge

$$H := \{ (\mathbf{x}, \bar{Q}_\varepsilon(\mathbf{x})) : \mathbf{x} \in \mathbf{Z}^n,\ \varepsilon \in \{\pm 1\}^d,\ Q_\varepsilon(\mathbf{x}) \neq \varnothing \}.$$

Wegen Folgerung 3.20 und Satz 3.21 ist die Gravermenge endlich. – Ist die Gravermenge für eine Familie von gemischt-ganzzahligen Programmen bekannt, so können wir das *Augmentierungsproblem* lösen:

Algorithmus 3.24 (Augmentieren mit Gravermenge). Sei ein gemischt-ganzzahliges Programm durch $\mathbf{A} \in \mathbf{Q}^{m\times n}$, $\mathbf{C} \in \mathbf{Q}^{m\times d}$, $\mathbf{b} \in \mathbf{Q}^{m}$, $\mathbf{c}^\top \in \mathbf{Q}^{(n)}$, $\mathbf{d}^\top \in \mathbf{Q}^{(d)}$ und $\mathbf{l}, \mathbf{u} \in \mathbf{Q}^{n+d}$ gegeben, welches eine optimale Lösung besitzt.

1: **input** Gravermenge H, zulässiger Punkt $(\mathbf{x},\mathbf{y}) \in \mathbf{Z}^n \times \mathbf{Q}^d$;
2: **output** Vektor $(\mathbf{s},\mathbf{t})$ mit $\mathbf{c}^\top\mathbf{s}+\mathbf{d}^\top\mathbf{t} > 0$ und $(\mathbf{x}+\mathbf{s},\mathbf{y}+\mathbf{t})$ zulässig, falls existent;
3: **for all** $(\mathbf{s},\bar{Q}) \in H$ **do**
4:   **if** $\mathbf{l}_X \leq \mathbf{x}+\mathbf{s} \leq \mathbf{u}_X$ **then**
5:     Löse das LP: $\max\left\{\mathbf{d}^\top\mathbf{t} : \mathbf{t} \in \bar{Q},\ \mathbf{l}_Y - \mathbf{y} \leq \mathbf{t} \leq \mathbf{u}_Y - \mathbf{y}\right\}$;
6:     **if** Programm lösbar **then**
7:       Sei $\mathbf{t} \in \mathbf{Q}^d$ eine Lösung;
8:       **if** $\mathbf{c}^\top\mathbf{s}+\mathbf{d}^\top\mathbf{t} > 0$ **then**
9:         **return** $(\mathbf{s},\mathbf{t})$
10:       **fi**
11:     **fi**
12:   **fi**
13: **od**;
14: Sei $\mathbf{t}$ eine Lösung des LP: $\max\left\{\mathbf{d}^\top\mathbf{t} : \mathbf{C}\mathbf{t} = \mathbf{0},\ \mathbf{l}_Y - \mathbf{y} \leq \mathbf{t} \leq \mathbf{u}_Y - \mathbf{y}\right\}$;
15: **return** $(\mathbf{0},\mathbf{t})$.

Satz 3.25. *Algorithmus 3.24 ist korrekt.*

Beweis. Sei $(\mathbf{x}^*,\mathbf{y}^*)$ eine optimale Lösung von MIP, und sei $(\mathbf{x},\mathbf{y})$ ein nicht-optimaler zulässiger Punkt. Sei $\varepsilon \in \{\pm 1\}^d$ mit $\mathbf{y}^* - \mathbf{y} \in \mathscr{O}_\varepsilon$. Nach Folgerung 3.10 gibt es eine Zahl $k \in \mathbf{N}_0$ und Tupel $(\mathbf{x}^1,\mathbf{y}^1),\dots,(\mathbf{x}^k,\mathbf{y}^k) \in \mathbf{Z}^n \times \mathbf{R}^d$ mit $\mathbf{y}^i \in Q_\varepsilon(\mathbf{x}^i) \subseteq \bar{Q}_\varepsilon(\mathbf{x}^i)$ sowie ein Tupel $(\mathbf{x}^0,\mathbf{y}^0) \in \{\mathbf{0}\} \times \mathscr{C}_\varepsilon$, so daß

$$(\mathbf{x}^*,\mathbf{y}^*) - (\mathbf{x},\mathbf{y}) = (\mathbf{x}^0,\mathbf{y}^0) + (\mathbf{x}^1,\mathbf{y}^1) + \cdots + (\mathbf{x}^k,\mathbf{y}^k) \tag{3.8}$$

und $(\mathbf{x}^i,\mathbf{y}^i) \preceq_{\mathbf{A}|\mathbf{C}} (\mathbf{x}^*,\mathbf{y}^*) - (\mathbf{x},\mathbf{y})$ für $i \in \{0,\dots,k\}$ gelten. Da $\mathbf{c}^\top\mathbf{x}^* + \mathbf{d}^\top\mathbf{y}^* > \mathbf{c}^\top\mathbf{x} + \mathbf{d}^\top\mathbf{y}$ ist, gibt es einen Index $i \in \{0,\dots,k\}$ mit

$$\mathbf{c}^\top\mathbf{x}^i + \mathbf{d}^\top\mathbf{y}^i > 0. \tag{3.9}$$

Wegen $\mathbf{x}^i \preceq \mathbf{x}^* - \mathbf{x}$ und $\mathbf{y}^i \preceq \mathbf{y}^* - \mathbf{y}$ folgt wie im Beweis von Lemma 1.16 die Zulässigkeit von $(\mathbf{x}+\mathbf{x}^i,\mathbf{y}+\mathbf{y}^i)$ für MIP.

Falls es einen Index $i \in \{1,\dots,k\}$ mit (3.9) gibt, so ist $(\mathbf{s},\bar{Q}) := (\mathbf{x}^i,\bar{Q}_\varepsilon(\mathbf{x}^i)) \in H$, und $\mathbf{y}^i$ ist eine zulässige Lösung für das lineare Programm aus Zeile 5 mit $\mathbf{c}^\top\mathbf{x}^i + \mathbf{d}^\top\mathbf{y}^i > 0$. Da das Programm einen beschränkten zulässigen Bereich besitzt, gibt es eine optimale Lösung $\mathbf{t}$, und auch diese erfüllt die Bedingung $\mathbf{c}^\top\mathbf{s}+\mathbf{d}^\top\mathbf{t} > 0$ aus Zeile 8. Somit wird die Zeile 9 im Verlauf des Verfahrens erreicht. Wegen der Wahl des zulässigen Bereichs des linearen Programms aus Zeile 5 erfüllt der konstruierte Punkt $(\hat{\mathbf{x}},\hat{\mathbf{y}}) := (\mathbf{x}+\mathbf{s},\mathbf{y}+\mathbf{t})$ die Schranken, und wegen $\mathbf{t} \in \bar{Q} \subseteq P'_\varepsilon(\mathbf{s})$ gilt auch $\mathbf{A}\hat{\mathbf{x}} + \mathbf{C}\hat{\mathbf{y}} = \mathbf{A}\mathbf{x} + \mathbf{C}\mathbf{y} = \mathbf{b}$. Somit ist die Bedingung aus Zeile 2 erfüllt.

Falls in der Schleife in Zeilen 3–13 keine verbessernde Richtung gefunden wird, so gibt es insbesondere keinen Index $i \in \{1,\dots,k\}$ mit (3.9). Daher wird (3.9) mit $i = 0$ erfüllt. Da $(\mathbf{x}, \mathbf{y} + \mathbf{y}^0)$ ein zulässiger Punkt für MIP ist, ist $\mathbf{y}^0$ eine zulässige Lösung für das lineare Programm aus Zeile 14. Da dieses Programm einen beschränkten zulässigen Bereich hat, gibt es eine optimale Lösung $\mathbf{t}$. Der Punkt $(\mathbf{x}, \mathbf{y} + \mathbf{t})$ erfüllt dann die Bedingung aus Zeile 2. $\square$

Bemerkung 3.26. In Verallgemeinerung von Definition 1.12 c können wir das *gerichtete Augmentierungsproblem* für gemischt-ganzzahlige Programme formulieren:

> Gegeben $\mathbf{c}^\top, \bar{\mathbf{c}}^\top \in \mathbf{Q}^{(n)}$ und $\mathbf{d}^\top, \bar{\mathbf{d}}^\top \in \mathbf{Q}^{(d)}$ und ein nicht-optimaler Punkt $(\mathbf{x}, \mathbf{y})$, finde einen Vektor $(\mathbf{s}, \mathbf{t})$, so daß
>
> $$\mathbf{c}^\top \mathbf{s}^+ + \mathbf{d}^\top \mathbf{t}^+ - \bar{\mathbf{c}}^\top \mathbf{s}^- - \bar{\mathbf{d}}^\top \mathbf{t}^- > 0$$
>
> gilt und $(\mathbf{x} + \mathbf{s}, \mathbf{y} + \mathbf{t})$ zulässig ist.

Mit einer Variante von Algorithmus 3.24 läßt sich auch dieses Problem lösen. Die entscheidende Beobachtung hierzu ist, daß in jedem Schritt der Schleife 3–13 mit dem Element $(\mathbf{s}, \bar{Q}_\varepsilon(\mathbf{s})) \in H$ der Gravermenge ein Orthant $\mathscr{O}_\varepsilon \subset \mathbf{R}^d$ fixiert ist. Auf einem Orthanten $\mathscr{O}_\varepsilon$ ist das Funktional $\mathbf{t} \mapsto \mathbf{d}^\top \mathbf{t}^+ - \bar{\mathbf{d}}^\top \mathbf{t}^-$ aber linear; es wird dort dargestellt durch $(\mathbf{d}^\varepsilon)^\top \in \mathbf{R}^{(d)}$ mit

$$d_i^\varepsilon := \begin{cases} d_i & \text{für } \varepsilon_i = 1, \\ \bar{d}_i & \text{für } \varepsilon_i = -1. \end{cases}$$

Somit muß wie im Fall des gewöhnlichen Augmentierungsproblems nur eine Folge von linearen Programmen gelöst werden.

Es ist nicht bekannt, ob sich das Optimierungsproblem für gemischt-ganzzahlige Programme polynomiell auf ein geeignetes Augmentierungsproblem reduzieren läßt, wie dies bei ganzzahligen Programmen der Fall ist. Untersuchungen dieser Art sind nicht Gegenstand der vorliegenden Arbeit gewesen.

Thesen

1. Mit einem kombinatorischen Verfahren, das auf R. Urbaniak zurückgeht, können die primitiven Partitionsidentitäten rechnerisch bestimmt werden. Eine genaue theoretische Untersuchung von notwendigen und hinreichenden Bedingungen für die Irreduzibilität der im Verfahren erzeugten Vektoren führt zu einer Variante des Verfahrens mit verbesserter Laufzeit.

2. Die Aufgabe, zu einem gegebenen Vektor einen Reduzierer zu finden, läßt sich mit einer Datenstruktur für orthogonale Bereichssuche in Punktmengen effizient lösen, sobald alle irreduziblen Vektoren berechnet und in der Datenstruktur gespeichert sind. Für diese Aufgabe geeignete Datenstrukturen werden vorgestellt.

3. Auf der Menge der irreduziblen Vektoren wird eine Halbordnung, die Vergröberungsrelation, eingeführt; unter einer Vergröberung bleibt die Irreduzibilität eines Tupels erhalten. Der Aufwand zur Speicherung der irreduziblen Vektoren kann auf Kosten einer erhöhten Anfragezeit verringert werden, indem nur die minimalen Elemente gespeichert werden.

4. Mit den genannten und weiteren Techniken wird das Verfahren von R. Urbaniak implementiert, so daß die primitiven Partitionsidentitäten mit größtem Teil höchstens gleich 27 berechnet werden können. Zuvor war es nur möglich, die primitiven Partitionsidentitäten mit größtem Teil höchstens gleich 22 zu berechnen, da die Laufzeit in Abhängigkeit von der Problemgröße viel stärker wuchs.

5. Die primitiven Partitionsidentitäten sind eng mit der minimalen erzeugenden Menge einer speziellen Familie von Rucksackproblemen verbunden. Eine Modifikation des Verfahrens von R. Urbaniak kann als Prozedur in einem rekursiven Berechnungsschema verwendet werden, welches die minimale erzeugende Menge von beliebigen Familien von Rucksackproblemen bestimmt. Dieser Algorithmus ist für Probleminstanzen mit genügend dichten Gewichtmengen schneller als ein *General-purpose*-Verfahren zur Berechnung der minimalen erzeugenden Menge.

6. Für Differenzvektoren zulässiger Punkte von gemischt-ganzzahligen Programmen wird ein Reduktionsbegriff eingeführt. Dieser erlaubt die Zerlegung eines Differenzvektors in endlich viele Summanden, die nicht weiter zerlegt werden können.

7. Es wird eine endliche Beschreibung einer Menge angegeben, die für jeden Differenzvektor einen Reduzierer enthält; dazu werden die Polyeder des kontinuierlichen Anteils untersucht. Offen bleibt die Frage nach einem kombinatorischen Verfahren zur Berechnung der Beschreibung dieser Menge.

8. Wenn diese Beschreibung berechnet ist, kann das gewöhnliche und das gerichtete Augmentierungsproblem für gemischt-ganzzahlige Programme gelöst werden. Dazu wird ein Algorithmus angegeben, in dessen Verlauf eine Folge von linearen Programmen gelöst wird. Es bleibt offen, ob sich das Optimierungsproblem für gemischt-ganzzahlige Programme polynomiell auf ein geeignetes Augmentierungsproblem reduzieren läßt.

Literaturverzeichnis

[Cha90] Chazelle, Bernard: *Lower bounds for orthogonal range searching: I. The reporting case*. J. ACM, 37(2):200–212, April 1990.

[dB$^+$97] de Berg, Mark et al.: *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin–Heidelberg–New York, 1997.

[DGS96] Diaconis, P., R. L. Graham und B. Sturmfels: *Primitive partition identities*. In: Miklós, D., V. T. Sós und D. Szőnyi (Herausgeber): *Combinatorics, Paul Erdős is Eighty, Volume 2*, Band 2 der Reihe *Bolyai Society Mathematical Studies*, Seiten 173–192, Budapest, 1996.

[GLS93] Grötschel, Martin, László Lovász und Alexander Schrijver: *Geometric Algorithms and Combinatorial Optimization*, Band 2 der Reihe *Algorithms and Combinatorics*. Springer-Verlag, Berlin–Heidelberg–New York, Zweite Auflage, 1993.

[HMW] Helmberg, Christoph, Alexander Martin und Robert Weismantel: *Lectures in Linear, Integer and Semidefinite Programming*. Manuskript.

[Knu98] Knuth, Donald E.: *Sorting and Searching*, Band 3 der Reihe *The Art of Computer Programming*. Addison–Wesley, Reading, Massachusetts, zweite Auflage, 1998.

[Meh84] Mehlhorn, Kurt: *Sorting and Searching*, Band 1 der Reihe *Data Structures and Algorithms*. Springer-Verlag, Berlin–Heidelberg–New York, 1984.

[MN90] Mehlhorn, Kurt und Stefan Näher: *Dynamic fractional cascading*. Algorithmica, 5:215–241, 1990.

[Ove88] Overmars, Mark H.: *Efficient data structures for range searching on a grid*. Journal of Algorithms, 9:254–275, 1988.

[Sch86] Schrijver, Alexander: *Theory of Linear and Integer Programming*. Wiley–Interscience series in discrete mathematics. John Wiley & Sons, Chichester–New York–Brisbane–Toronto–Singapore, 1986.

[Smi94] Smid, Michiel: *Selected topics in data structures*. Lecture notes MPI-I-94-155, Max-Planck-Institut für Informatik, Saarbrücken, Dezember 1994.

[Smi97] Smid, Michiel: *Implementing dictionaries using hashing*. Lecture notes, Otto-von-Guericke-Universität Magdeburg, 1997.

[SO89] Scholten, Hans W. und Mark H. Overmars: *General methods for adding range restrictions to decomposable searching problems*. J. Symbolic Computation, 7:1–10, 1989.

[SWZ95] Schulz, Andreas S., Robert Weismantel und Günter M. Ziegler: *0/1-integer programming: Optimization and augmentation are equivalent*. In: Spirakis, P. (Herausgeber): *Algorithms – ESA '95*, Band 979 der Reihe *Lecture Notes in Computer Science*, Seiten 473–483. Springer-Verlag, Berlin – Heidelberg – New York, 1995.

[Urb98] Urbaniak, Regina: *Decomposition of Generating Sets and of Integer Programs*. Doktorarbeit, Technische Universität Berlin, 1998.

[Wil83] Willard, Dan E.: *Log-logarithmic worst-case range queries are possible in space* $\Theta(N)$. Information Processing Letters, 17:81–84, 1983.

[Wil84] Willard, Dan E.: *New trie data structures which support very fast search operations*. Journal of Computer and System Sciences, 28:379–394, 1984.

Erklärung

Ich versichere, daß ich diese Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Magdeburg, den 30. August 1999

Matthias Köppe