Fachbereich Mathematik
Technische Universität Darmstadt

Gemischt-ganzzahlige Optimierung am Beispiel von Losgrößenproblemen

Bachelorarbeit

Philipp Walter

August 2010

Betreuer/Gutachter: Dr. habil. Marco Lübbecke
Zweitgutachter: Prof. Dr. Stefan Ulbrich

Vorwort

Vorab möchte ich Herrn Dr. habil. Lübbecke zur tatkräftigen Unterstützung bei der Auswahl der Thematik und der guten Betreuung danken.
Diese Ausarbeitung richtet sich an Leserinnen und Leser, die über ein solides Grundwissen in linearer Optimierung verfügen, also mit den Methoden zur Lösung linearer Probleme mit reellen Variablen vertraut sind, und an neuen Methoden, insbesondere zur Lösung (gemischt-)ganzzahliger Probleme interessiert sind. Diese Methoden werden in dieser Arbeit anhand der Losgrößenproblematik vorgeführt.
Da der Großteil der Forschung zu diesem Thema in englischen Texten und Büchern resultiert, werden in geeigneten Umfang auch die englischen Begrifflichkeiten aufgeführt um diese Arbeit zu anderen Quellen kompatibel zu machen. Auch wird in speziellen Fällen, in denen eine Übersetzung nicht angebracht erscheint, mit dem englischen Begriff gearbeitet.

Erklärung zur Bachelorarbeit

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 6.August 2010

(Philipp Walter)

Abbildungsverzeichnis

Notation

Daten

NI - Anzahl der Produkte

NK - Anzahl der Maschinen

NT - Anzahl der Perioden

d_t^i - Nachfrage nach Produkt i in Periode t

p_t^{ik} - Produktionsrate von Produkt i auf Maschine k in Periode t

C_t^{ik} - Maximale Produktion von Produkt i auf Maschine k in Periode t

L_t^k - Gesamtkapazität von Maschine k

a_t^{ik} - benötigte Kapazität von Produkt i auf Maschine k in Periode t

b_t^{ik} - benötigte Fixkapazität falls Maschine k für Produkt i in Periode t eingerichtet ist

Variablen

s_t^i - Bestand von Produkt i in Periode t

r_t^i - Auftragsbestand (*backlog*) von Produkt i in Periode t

x_t^{ik} - Produktion von Produkt i auf Maschine k in Periode t

y_t^{ik} - Set-up Variable von Produkt i auf Maschine k in Periode t

z_t^{ik} - Start-up Variable von Produkt i auf Maschine k in Periode t

w_t^{ik} - Switch-off Variable von Produkt i auf Maschine k in Periode t

Inhaltsverzeichnis

1 Einführung

Im Vergleich zur linearen Optimierung mit reellen Variablen, deren Zusammenhänge und Lösungsansätze weitestgehend befriedigend erforscht sind, bietet das Feld der (gemischt-) ganzzahligen Optimierung viele neue interessante Ansätze, deren Erforschung in den letzten Jahren viel Aufmerksamkeit gewidmet wurde. Wie bei der linearen Optimierung mit reellen Variablen haben die betrachteten Probleme auch hier lineare Zielfunktionen, jedoch ergibt sich durch die Ganzzahligkeitsbedingungen einiger oder aller Variablen ein völlig andersartiges Problemfeld. Der Fall der gemischt-ganzzahligen Probleme, im Folgenden MIPs (*mixed integer problems*) genannt, bei dem ein Teil der Variablen diskrete und ein anderer Teil reelle Werte hat, ist in der Praxis besonders relevant, da in der Realität in vielen Fällen nur ganzzahlige Mengen produziert, befördert oder gebaut werden können. Beispiele hierfür sind Netzwerkprobleme mit fixen Kosten (*Fixed-Charge Network Problems*), Standortplanungsprobleme (*Facility Location Problems*) und Losgrößenprobleme (*Lot-sizing Problems*). Auch der Spezialfall der rein-ganzzahligen Probleme kann mit Methoden der gemischt-ganzzahligen Programmierung angegangen werden.
Wir werden in diesem Kapitel nun zunächst den Begriff des MIPs einführen und daran ansetzend den Branch & Bound Algorithmus zur Lösung desselben erläutern. Danach vermitteln wir die Grundidee des Hizufügens gültiger Ungleichungen, die in dem darauffolgend vorgestellten Branch & Cut Algorithmus einfließt. Zuletzt veranschaulichen wir das Beschriebene an einem Beispiel.

1.1 MIPs

Definition 1.1 (MIP)**.**
Ein gemischt-ganzzahliges Problem (mixed integer problem) ist ein Problem der Art

$$\min\{cx + fy : (x, y) \in X\}$$

bezüglich der Zielfunktion $cx + fy$ über der zulässigen Menge

$$X = \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+ : Ax + By \geq b\}$$

Mit Zielfunktionsvektoren $c \in \mathbb{R}^n$ und $f \in \mathbb{R}^p$ sowie Nebenbedingungsmatrizen $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times p}$ und dem Vektor $b \in \mathbb{R}^m$ der rechten Seite. Den Optimalwert des MIPs nennen wir $Z(X)$. Ist $X = \emptyset$ so setzen wir $Z(X) := \infty$ (vgl. [20]).

Haben wir den Spezialfall $p = 0$ so liegt das bereits bekannte (reelle) lineare Problem vor. Demgegenüber steht der Fall $n =$ woraus sich ein rein-ganzzahliges Problem ergibt.

Im Folgenden werden wir in besonderem Maße die lineare Relaxation eines Problems benötigen, welche aus dem ursprünglichen MIP entsteht indem man den Zulässigkeitsbereich ins Reelle relaxiert.

Definition 1.2 (LR)**.** *Unter der linearen Relaxation LR eines MIPs, das wie in Definition 1.1 definiert ist, verstehen wir das lineare Programm*

$$\min\{cx + fy : (x, y) \in P_X\}$$

$$mit \quad P_X = \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ : Ax + By \geq b\}.$$

mit Optimalwert $Z(P_X)$. Dabei bezeichnen wir P_X als eine Formulierung für X.

Vorteil der linearen Relaxation ist, dass wir sie „einfach“ lösen können, z.B. mit dem uns aus der linearen Optimierung bekannten Simplex-Verfahren. Diesen Vorteil wollen wir ausnutzen.
Ein häufiger Ansatz bei Problemen mit ganzzahligen Daten ist das „Divide-and-conquer“-Prinzip. Dieses findet explizit Anwendung im Branch & Bound Algorithmus, der im Folgenden beschrieben wird. Leser, die diesen Algorithmus bereits kennen, können folgendes Kapitel überspringen.

1.2 Branch & Bound

Der nun beschriebene Branch & Bound Algorithmus ist der Basisalgorithmus zum Lösen (gemischt-)ganzzahliger Probleme. Dabei wird nach dem „Divide-and-conquer“-Prinzip das Problem immer weiter aufgespaltet und somit in kleinere Probleme zerlegt. In jedem Schritt wird dann der Wert der linearen Relaxation berechnet und damit untere und obere Schranken für den Optimalitätswert des MIP errechnet. Diese Idee wurde erstmals 1960 von A.H. Land und A.G. Doig in [10] formuliert.

Unser MIP habe den Optimalwert $Z(X)$, wie oben definiert. Betrachte nun mit $Z(P_X)$ den Optimalwert bezüglich der linearen Relaxation LR des MIPs mit

$$P_X = \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ : Ax + By \geq b\}.$$

Bemerkung 1.3.
Zwischen den Optimalwerten kann man die folgende Beziehung feststellen

$$Z(P_X) \leq Z(X) \leq \overline{Z},$$

wobei $\overline{Z}$ der Zielfunktionswert einer jeden gefundenen zulässigen Lösung ist. Damit ist $Z(P_X)$ also eine untere Schranke für den Optimalwert unseres MIPs.

Mithilfe des „Divide-and-conquer“-Prinzips gehen wir nun folgendermaßen vor, siehe [20]:

1. Unsere erste untere Schranke erhalten wir durch den Optimalwert der LR $Z(P_X)$. Dieser Wert ist aufgrund der reellen Variablen von LR leicht berechenbar, z.B. mit dem Simplex-Algorithmus. Sei (x^*, y^*) diese Optimallösung von LR.

2. a) Falls $y^* \in \mathbb{Z}^p$ dann ist die gefundene Lösung von LR offensichtlich auch zulässig für unser MIP, da $(x^*, y^*) \in X$ und ihr Optimalwert stellt eine obere Schranke dar. Da obere und untere Schranke für $Z(X)$ nun gleich sind haben wir eine Optimallösung gefunden

   b) Andernfalls gilt $y^* \notin \mathbb{Z}^p$ und die Lösung (x^*, y^*) ist nicht zulässig für MIP. Wir wollen nun Nebenbedingungen hinzufügen sodass wir diese nicht-ganzzahlige Lösung eliminieren. Wähle dazu $y_j^* \notin \mathbb{Z}$ mit $j \in 1, ..., p$. Wir bemerken, dass für jede zulässige Lösung von MIP nun gelten muss

   $$y_j \leq \lfloor y_j^* \rfloor \quad \text{oder} \quad y_j \geq \lceil y_j^* \rceil$$

   Branching: Zum Eliminieren der Lösung gehen wir jetzt wie folgt vor: Wir ersetzen die Formulierung P_X durch die Vereinigung der Formulierungen P_X^0 und P_X^1 mit

   $$P_X^0 = P_X \cap \{(x, y) \in \mathbb{R}_+^n \times \mathbb{R}_+^p : y_j \leq \lfloor y_j^* \rfloor\} \quad \text{und}$$

   $$P_X^1 = P_X \cap \{(x, y) \in \mathbb{R}_+^n \times \mathbb{R}_+^p : y_j \geq \lceil y_j^* \rceil\}.$$

   Diesen Schritt nennt man Branchen, zu deutsch „Verzweigen“ und y_j Branching-Variable.

3. Nun suchen wir nach einer Optimallösung in der Liste $L = \{P_X^0, P_X^1\}$. Diese Dekomposition können wir nun iterativ immer so weiter ausführen.

   Hauptiteration: Gegeben ist eine Liste von Formulierungen L und der Wert $\overline{Z}$ der besten ganzzahligen Lösung, die bisher gefunden wurde. Haben wir noch keine zulässige Lösung für unser MIP gefunden so setzen wir $\overline{Z} = +\infty$

   Auswahl und Lösung: Wir suchen uns nun aus L eine Formulierung V aus und berechnen den Optimalwert $Z(V)$ des LPs. Dieser stellt eine untere Schranke für den besten zulässigen Wert des MIPs bezüglich der Formulierung V dar.

   Pruning: zu deutsch „Abschneiden“, dabei können für den Optimalwert $Z(V)$ mehrere Fälle auftreten:

   a) Falls $Z(V) \geq \overline{Z}$ dann kann der beste Wert eine Lösung in V nicht besser sein als $\overline{Z}$ da $Z(V)$ eine untere Schranke für die beste Lösung des MIPs bzgl. V darstellt. Wir müssen V also nicht weiter betrachten und streichen es aus L. Dies nennt man *Pruning by bound*.

   b) Als Spezialfall des Ersten haben wir falls $V = \emptyset \Rightarrow Z(V) = +\infty$ per Definition und entfernen V wie oben aus L. Dies nennt man *Pruning by infeasibility*

   c) Falls $Z(V) < \overline{Z}$ und $y^V \in \mathbb{Z}^p$ haben wir die beste ganzzahlige Lösung in V gefunden. Wir müssen also V nicht weiter aufteilen. Wir setzen $\overline{Z} = Z(V)$ als obere Schranke und entfernen V aus L. Dies nennt man *Pruning by integrality*.

   d) Falls $Z(V) < \overline{Z}$ und $y^V \notin \mathbb{Z}^p$ haben wir keine zulässige Lösung für unser MIP bzgl. V gefunden und müssen weiter aufzweigen. Dies nennt man *Branching*(siehe 2b).

4. **Terminierung:** Der Algorithmus endet wenn die Liste L leer ist. Dies ist garantiert nach endlich vielen Schritten der Fall falls die Variablen y beschränkt sind. Leider kann die Länge der Liste L exponentiell viele Formulierungen bezüglich der Zahl der Variablen p haben. Aufgrund dieses Aspekts ist auch die Laufzeit des Branch & Bound Algorithmus exponentiell. Insbesondere bei der Anwendung im industriellen Bereich ist die Problemgröße oft derart, dass Branch & Bound nicht in akzeptabler Zeit die genaue Optimallösung bestimmen kann. Oftmals werden im Algorithmus deshalb zusätzlich Zeitschranken eingebaut, damit der Algorithmus spätestens nach einer gewissen Zeit endet. Alternativ kann man auch eine Genauigkeitsschranke angeben, damit der Algorithmus bei einer gewissen Güte der Lösung abbricht. Diese Güte geben wir mithilfe der Ganzzahligkeitslücke(*duality gap*) an.

Definition 1.4.
Explizit ist die Ganzzahligkeitslücke gegeben durch die relative Abweichung zur besten bekannten zulässigen Lösung

$$duality\ gap = \frac{Best\ UB - Best\ LB}{Best\ UB} \times 100[\%]$$

Dabei ist Best LB die kleinste untere Schranke der ausstehenden Probleme in der Liste L, also $\min_{V \in L} Z(V)$*, und Best UB der Wert der besten bisher gefundenen zulässigen Lösung.*

Durch verschiedene Knotenauswahl- bzw. Branchingregeln, wie z.B. in [1] aufgeführt, ergeben sich dabei verschiedene Varianten des Branch & Bound Algorithmus', auf die wir nicht näher eingehen wollen.

1.3 Grundidee - Gültige Ungleichungen

Beim oben beschriebenen Algorithmus haben wir nun oftmals das Problem dass die untere Schranke zu Beginn sehr schlecht ist und wir daher enorme Rechenzeit benötigen um ein akzeptables Ergebnis zu bekommen. Die Gründe dafür liegen oft in der mangelhaften Modellierung des gegebenen Problems. So kann die zulässige Menge auf die unterschiedlichste Art und Weise beschrieben sein, wie folgendes Beispiel illustriert.

Beispiel 1.5.
Hier wird die zulässige Menge sowohl durch X als auch durch Y dargestellt mit

$$\begin{aligned} X = \{(y_1, y_2) \in \mathbb{Z}^2 : -y_1 - 0.3y_2 &\geq 4.3, \\ 0.85y_1 - y_2 &\geq 2, \\ 2y_1 + y_2 &\geq 2.6, \\ y_1 + 2.4y_2 &\geq 3.8, \\ 4.5y_1 - y_2 &\geq 14.4\} \end{aligned}$$

$$\begin{aligned} Y = \{(y_1, y_2) \in \mathbb{Z}^2 : y_2 - y_1 &\geq -1.2, \\ y_1 + 1.1y_2 &\geq 2.7, \\ -1.7y_1 + y_2 &\geq -4.25, \\ -y_1 - 1.25y_2 &\geq -7\} \end{aligned}$$

Allerdings sind P_X und P_Y offensichtlich nicht gleich, wie in Abbildung 1 (vgl. [20]) zu erkennen.

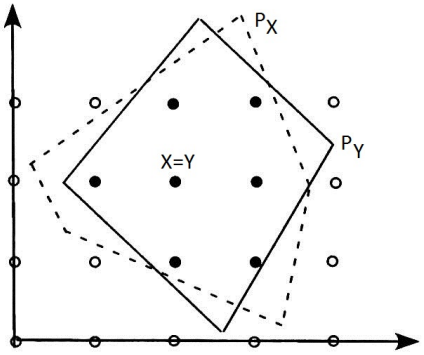

Abbildung 1: 2 Formulierungen für die selbe zulässige ganzzahlige Menge

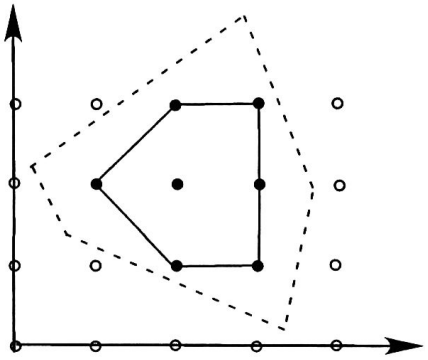

Abbildung 2: Die konvexe Hülle von X

Es ist also notwendig im Folgenden genauer auf den Raum der zulässigen Lösungen X einzugehen, welcher durch ein Polyeder beschrieben ist, um ein Verständnis einer guten Modellierung zu bekommen. Definieren wir uns dazu die konvexe Hülle von X wie sie in Abbildung 2 beispielhaft dargestellt ist.

Definition 1.6 (konvexe Hülle)**.**
Die konvexe Hülle einer Menge X, oder kurz $conv(X)$, ist die Menge aller Punkte der Form

$$x = \sum_{i=1}^{T} \lambda_i x^i, \quad \sum_{i=1}^{T} \lambda_i = 1, \quad \lambda_i \geq 0 \text{ für } i = 1, ..., T$$

wobei $\{x^1, ..., x^T\}$ jede endliche Menge von Punkten aus X ist. Ein Punkt der wie x dargestellt werden kann bezeichnet man als Konvexkombination der Punkte $x^1, ..., x^T$

Das Hauptziel beim Ansatz der gültigen Ungleichungen ist nun das Verstärken der gegebenen Formulierung in dem Sinne dass wir das Polyeder der zulässigen ganzzahligen Lösungen besser, also näher an $conv(X)$, modellieren wollen. Dazu bemerken wir

Bemerkung 1.7.
Die konvexe Hülle $conv(X)$ ist die engstmögliche und somit beste gültige Formulierung für die zulässige Menge X.

Beweis. vgl. [14] □

Somit liegt der Gedanke nah einfach eine lineare Formulierung für $conv(X)$ zu finden, da dann die Optimallösung der linearen Relaxation der des MIPs entspricht, da sie bekanntermaßen am Rand des Polyeders liegt und wir so immer eine ganzzahlige Lösung erhalten siehe [15]. Diese können wir dann direkt mit dem Simplex Algorithmus ausrechnen. Das Problem dabei ist allerdings, dass das Finden dieser linearen Formulierung von $conv(X)$ im Komplexitätssinne mindestens genauso „hart“ ist wie das Lösen der ursprünglichen Formulierung (vgl. [20]). Außerdem besteht diese Formulierung, haben wir sie denn gefunden, in aller Regel aus einer exponentiellen Anzahl von Ungleichungen bezüglich der Zahl der Variablen.
Allerdings können wir gültige Ungleichungen, wie unten definiert, finden und diese zur gegebenen Formulierung hinzufügen. Dabei gibt es mehrere Vorgehen die im Folgenden erläutert werden.

Definition 1.8 (gültige Ungleichung)**.**
Eine gültige Ungleichung für die zulässige Menge X eines MIPs

$$X = \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+ : Ax + By \geq b\}$$

wird definiert durch eine Ungleichung $\alpha x + \beta y \geq \gamma$ ($mit\ \alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^p\ und\ \gamma \in \mathbb{R}$) die von allen Punkten in X erfüllt wird.

Mit dem Hinzufügen solcher gültigen Ungleichungen geht dann, vorrausgesetzt wir schneiden die aktuelle fraktionale Lösung ab, eine Verbesserung der unteren Schranke einher, die wir als Optimalwert der linearen Relaxation LR des gegebenen Problems erhalten, und die maßgeblich die Laufzeit von einem Algorithmus wie Branch & Bound beeinflusst.
Ein Weg des Hinzufügens dieser Ungleichungen ist die Neuformulierung der Nebenbedingungen A Priori, also vor Start unseres Algorithmus'.

Definition 1.9 (Neuformulierung)**.**
Eine Neuformulierung von X durch eine Familie gültiger Ungleichungen $\mathcal{C}$ ist die Formulierung die durch die ursprünglichen Ungleichungen und die neuen Ungleichungen entsteht

$$\begin{aligned} X =& \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+ : \ Ax + By \geq b\} \\ =& \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+ : \ Ax + By \geq b \\ & \alpha^j x + \beta^j y \geq \gamma^j \quad \textit{für alle } j = 1, ..., |\mathcal{C}|\}, \end{aligned}$$

Wir erhalten damit eine verbesserte Relaxation mit zulässiger Menge $\tilde{P_X}$ mit

$$\tilde{P_X} = P_X \cap C \subseteq P_X$$

und

$$C = \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^p : \alpha^j x + \beta^j y \geq \gamma^j \quad \textit{für alle } j = 1, ..., |\mathcal{C}|\}$$

die Menge der Punkte die allen gültigen Ungleichungen der Familie $\mathcal{C}$ genügen.

Mit dem Hinzufügen von Familien gültiger Ungleichungen geht allerdings das Problem einher, dass wir häufig exponentiell viele gültige Ungleichungen bezüglich der Anzahl der Variablen erhalten. Wenn wir diese nun A Priori in Form einer Neuformulierung einbringen so verschlechtert sich die Laufzeit unseres Algorithmus deutlich, da wir in jedem Schritt viel mehr Nebenbedingungen zu beachten haben.
Als Alternative könnte man nun eine erweiterte Neuformulierung verwenden. Hierbei erweitert man den Raum der Variablen und formuliert die Menge X durch neue Variablen, in aller Regel wesentlich mehr als zuvor. Damit erreichen wir eine polynomielle, oftmals sogar lineare, Anzahl an Gleichungen in der Anzahl der Variablen. Wir werden darauf nicht weiter eingehen, der interessierte Leser erhält dazu weitere Informationen in [20].

Unser Augenmerk wird sich im Folgenden auf eine andere Alternative richten, respektive den Schnittebenenverfahren. Dahinter steckt die Idee die gültigen Ungleichungen erst im Algorithmus selbst in Form von Schnittebenen (*cutting planes*) einfließen lassen. Wir entwickeln unseren Branch & Bound Algorithmus also zu einem Branch & Cut Algorithmus weiter. Dieser ist der aktuell meistbenutzte Algorithmus zur Lösung ganzzahliger Probleme (vgl. [20]). Er ist in neuerer Software wie z.B. XPRESS oder CPLEX implementiert (vgl. [7] bzw. [8]). Insbesondere erkennen diese Programme gewisse Nebenbedingungen des Eingabeproblems und erzeugen diese Schnittebenen so im Ablauf des Algorithmus' selbst aus.

1.4 Branch & Cut

Bevor wir uns im Weiteren mit der Herleitung der gültigen Ungleichungen beschäftigen, werden wir zunächst, ohne diese genau zu kennen, einen Blick auf die explizite Verwendung dieser Ungleichungen im Branch & Bound Algorithmus richten. Haben wir eine oder mehrere Familien gültiger Ungleichungen gefunden, so haben wir wie schon beschrieben oftmals das Problem der exponentiellen Größe dieser Familie. Durch studieren der Lösungsstruktur beobachten wir

Beobachtung 1.10.
Für ein gegebenes MIP mit gegebener Formulierung P_X sei $\tilde{P_X}$ die Neuformulierung die durch das Hinzufügen exponentiell vieler Ungleichungen entstanden ist. Für eine gegebene Zielfunktion $cx + fy$ ist nur eine begrenzte Anzahl (maximal $n + p$) dieser Ungleichungen nötig um den Optimalwert der linearen Relaxation über der Menge $\tilde{P_X}$ zu bestimmen.

Beweis.
Lösen wir die lineare Relaxation über $\tilde{P_X}$, so kann man die gültigen Ungleichungen, die in der Optimallösung inaktiv sind, vernachlässigen und die Optimallösung als Schnitt von maximal $n + p$ Ungleichungen eindeutig bestimmen. (vgl. lineare Optimierung)

□

Wir sehen also dass es reichen würde bei jeder Berechnung einer linearen Relaxation nur linear viele Ungleichungen (in der Anzahl der Variablen) hinzuzufügen. Wir wollen als nächstes sehen wie wir diese Ungleichungen identifizieren können.
Sei dabei C, wie bereits definiert, die Menge aller Punkte, die allen Ungleichungen der Menge $\mathcal{C}$ genügen.

Definition 1.11 (SEP)**.**
Für gegebenes MIP mit zulässiger Menge X stellt das Separationsproblem $SEP((x^*, y^*)|\mathcal{C})$ *für einen gegebenen Vektor* $(x^*, y^*) \in P_X$ *fest ob*

- $(x^*, y^*) \in C$*, also ob* (x^*, y^*) *alle gültigen Ungleichungen in* $\mathcal{C}$ *erfüllt*
- *oder eine Ungleichung* $(\alpha^j x + \beta^j y \geq \gamma^j) \in \mathcal{C}$ *existiert, die verletzt wurde*

Einen Algorithmus der dieses Separationsproblem löst nennt man Separationsalgorithmus. Die Idee ist im Weiteren, für jede Lösung einer linearen Relaxation im Branching-Baum die nichtzulässige Optimallösung von der Menge X zu trennen. Für eine gegebene Formulierung erledigt das ein sogenanntes Schnittebenenverfahren (vgl. [20]):

Definition 1.12. *Ein Schnittebenenverfahren löst für eine gegebene Formulierung V das relaxierte Problem, löst danach das Separationsproblem bzgl. der gefundenen (nichtganzzahligen) Lösung und fügt die gefundenen Ungleichungen zur Formulierung hinzu. Danach wird iterativ so fortgefahren bis das Separationsproblem keinen verletzten Ungleichungen mehr identifiziert. Formeller:*

1. *Initialisiere* $W := V$*. W wird am Ende des Schnittebenenverfahrens die entgültige Formulierung enthalten.*
   Berechne die Optimallösung für das relaxierte Problem $Z(W)$*. Sei* (x^*, y^*) *diese optimale Lösung.*
2. *Löse nun das Separationsproblem* $SEP((x^*, y^*)|\mathcal{C})$
   a) *falls alle Ungleichungen der Familie* $\mathcal{C}$ *von* (x^*, y^*) *erfüllt sind ist* $(x^*, y^*) \in V \cap C$ *Optimallösung mit Wert* $Z(V \cap C)$ *STOP*
   b) *falls ansonsten SEP eine verletzte Ungleichung* $\alpha^j x + \beta^j y \geq \gamma^j$ *identifiziert so fügen wir* $VI_j = \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^p : \alpha^j x + \beta^j y \geq \gamma^j\}$ *zur Formulierung hinzu, also*

   $$W := W \cap VI_j$$

   Berechne wieder den Optimalwert $Z(W)$*, sei* (x^*, y^*) *eine Optimallösung die diesen Wert annimmt.*
   GEHE ZU 2)

Die Nebenbedingungen die im Laufe des Schnittebenenverfahrens zurückgegeben werden nennt man Schnitte oder Schnittebenen. Ebenen weil sie Hyperebenen beschreiben die Halbräume aufspannen und Schnitte weil sie die aktuelle Lösung (x^*, y^*) von unserer gemischt-ganzzahligen Lösungsmenge abschneiden.

Zusammengefasst besteht der Branch & Cut Algorithmus also aus folgenden Komponenten (vgl. [20]):

1. **Initialisierung:** Setze $L = \{P_X\}, \quad \tilde{Z} := +\infty$. Nehme an dass das LR beschränkt ist.
2. **Terminierung:** Falls $L = \emptyset$
   - falls $\tilde{Z} = +\infty$ dann ist $X = 0$, das Problem also unzulässig
   - falls $\tilde{Z} < +\infty$ dann ist die Lösung $(x, y) \in X$ mit $\tilde{Z} = cx + fy$ optimal
3. **Knotenauswahl und Schnittebenenverfahren:** Wähle $V \in L$ aus und setze $L = L \setminus \{V\}$
   a) Berechne $Z(V)$ und die Optimallösung (x^V, y^V) über V
   b) Löse $SEP((x^V, y^V)|\mathcal{C})$
      i. falls alle Ungleichungen der Familie $\mathcal{C}$ von (x^V, y^V) erfüllt sind ist $(x^V, y^V) \in V \cap C$ Optimallösung mit Wert $Z(V \cap C)$ GEHE ZU 4
      ii. falls ansonsten SEP eine verletzte Ungleichung $\alpha^j x + \beta^j y \geq \gamma^j$ identifiziert so fügen wir $VI_j = \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^p : \alpha^j x + \beta^j y \geq \gamma^j\}$ zur Formulierung hinzu, also

      $$W := W \cap VI_j$$

      Berechne wieder den Optimalwert $Z(W)$, sei (x^*, y^*) eine Optimallösung die diesen Wert annimmt.
      GEHE ZU 3*b*)
4. **Pruning:** siehe Branch & Bound
5. **Branching:** siehe Branch & Bound

Abschließend bemerken wir, dass für einen exakten Separationsalgorithmus und die gleichen Knotenauswahl- bzw. Branchingregeln der Branch & Bound Algorithmus für die A Priori Neuformulierung und den Schnittebenenansatz die Knoten des Branchingbaums in der gleichen Reihenfolge durchläuft und somit zum gleichen Ergebnis führt. Allerdings werden gerade in der Praxis oft Heuristiken angewendet um SEP zu lösen, da bei großen Datenmengen die Rechenzeit sonst nicht akzeptabel ist. Von daher kann es durchaus Unterschiede machen, wann die Ungleichungen hinzugefügt werden.

1.5 Ein einfaches Beispiel

Wir betrachten nun folgendes Beispiel eines MIPs, in dem alle Variablen ganzzahlig gefordert sind. Diesen speziellen Fall nennt man auch rein-ganzzahliges Problem:

$$Z(X) = \min_y \{-y_1 - 2y_2 : y = (y_1, y_2) \in X\}$$

wobei die zulässige Menge X definiert ist durch

$$\begin{aligned} X = \{y = (y_1, y_2) \in \mathbb{Z}_+^2 : y_1 &\geq 1 \\ -y_1 &\geq -5 \\ -y_1 - 0.8y_2 &\geq -5.8 \\ y_1 - 0.8y_2 &\geq 0.2 \\ -y_1 - 8y_2 &\geq -26\}. \end{aligned}$$

Mit den Variablen von Definition 1.1 ist dann $n = A = c = 0$ und

$$m = 2, p = 2, f = (-1, -2), B = \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ -1 & -0.8 \\ 1 & -0.8 \\ -1 & -8 \end{pmatrix} \text{ und } b = \begin{pmatrix} 1 \\ -5 \\ -5.8 \\ 0.2 \\ -26 \end{pmatrix}$$

Graphisch ist die zulässige Lösungsmenge X wie in Abbildung 3 (vgl. [20]) zu erkennen gerade die Menge aller ganzzahligen Punkte in P_X. Wie aus der linearen Optimierung bekannt erhält man nun eine Optimallösung der linearen Relaxation mit zulässiger Menge P_X durch Verschieben der Zielfunktion in Richtung der Minimierung (vgl. [15]).

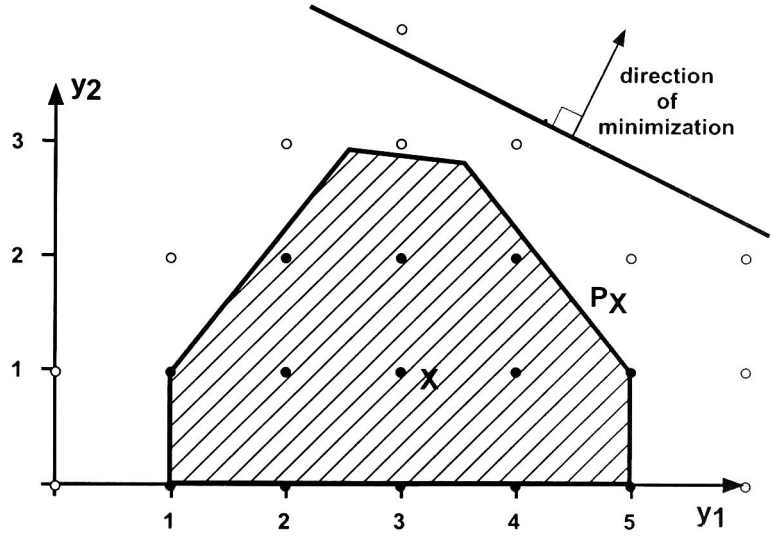

Abbildung 3: Unser MIP mit zulässigem Bereich X und Formulierung P_X

1. Lösen wir dieses MIP nun mit Branch & Bound so erhalten wir im ersten Schritt als untere Schranke den Wert der Optimallösung der linearen Relaxation. Diese liegt im Punkt $a = (\frac{32}{9}, \frac{101}{36})$ mit Optimalwert $-\frac{32}{9} - 2\frac{101}{36} = -\frac{165}{18}$
   Danach kommt es zum Branching bezüglich der Variable y_1 mit

$$P_X^0 = P_X \cap \{(x,y) \in \mathbb{R}_+^n \times \mathbb{R}_+^p : y_1 \leq \lfloor a_1 \rfloor = 3\} \quad \text{und}$$

$$P_X^1 = P_X \cap \{(x,y) \in \mathbb{R}_+^n \times \mathbb{R}_+^p : y_1 \geq \lceil a_1 \rceil = 4\}.$$

   Abbildung 4 illustriert nun das Aufzweigen der Menge P_X in die 2 Mengen P_X^0 und P_X^1. Man erkennt, dass die vorherige Optimallösung der LR nun in keiner Menge mehr enthalten ist und somit verworfen wird (vgl. [20]).

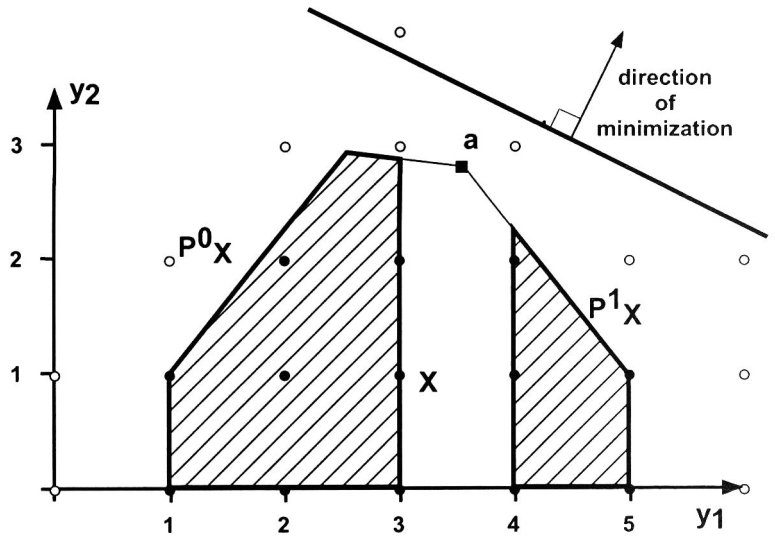

Abbildung 4: Branching unseres MIPs

Folgen wir nun unserem Branch & Bound Algorithmus so erhalten wir folgenden Branching-Baum in Abbildung 5 (vgl. [20]).

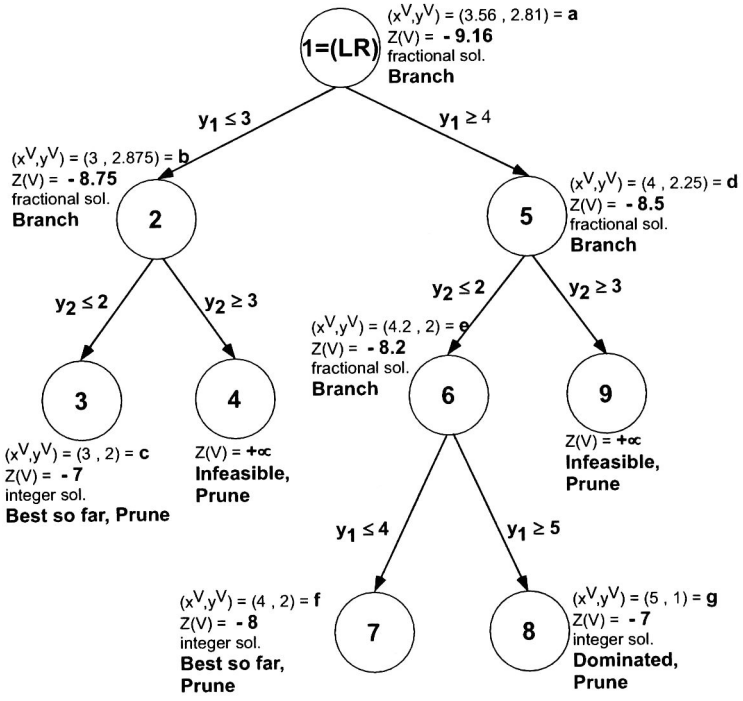

Abbildung 5: Branch & Bound Baum unseres MIPs

Die Knoten werden dabei in der Reihenfolge der Auswahl in Schritt 3 nummeriert. Der erste Knoten stellt die lineare Relaxation unseres Problems dar. An jedem weiteren Knoten wird entweder Pruning oder Branching angewendet. Haben wir ein Branching so ist auf der Kante die Branching Variable und deren Wert eingetragen.

2. Als Nächstes wollen wir den Ansatz der gültigen Ungleichungen in Form von Neuformulierungen A Priori veranschaulichen. Dazu betrachten wir in unserem Beispiel die Familie

$$\mathcal{C} = \{(y_1, y_2) \in \mathbb{R}^2_+ : \begin{matrix} -y_1 - y_2 \geq -6 \\ y_1 - y_2 \geq 0 \end{matrix}\}$$

Diese hat wie in Abbildung 6 zu sehen keinen Einfluss auf die zulässige Menge X, allerdings erhalten wir für unser relaxiertes Problem den zulässigen Bereich $\tilde{P_X} = P_X \cap C \subseteq P_X$ (vgl. [20]).

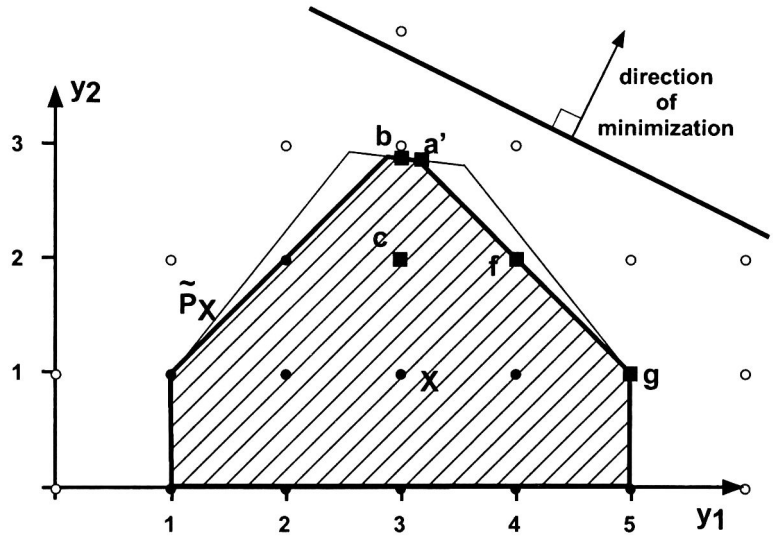

Abbildung 6: Engere Formulierung $\tilde{P_X}$ unseres MIPs

Durch das Hinzufügen dieser gültigen Ungleichungen haben wir also eine A Priori Neuformulierung vorgenommen. Wenden wir nun Branch & Bound auf unser neuformuliertes System an so erhalten wir den Branching-Baum, der in Abbildung 7 dargestellt ist. Dieser hat nun deutlich weniger Knoten, führt also somit schneller zum gewünschten Optimalwert. Dies ist zu erklären durch die verbesserte untere Schranke, die wir im 1. Knoten erhalten.

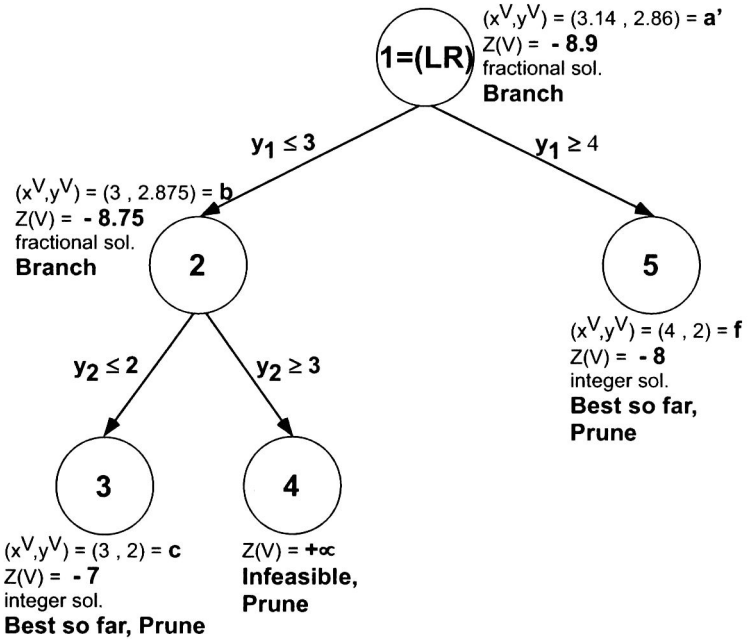

Abbildung 7: Branch & Bound Baum nach Neuformulierung unseres MIPs

3. Fügen wir nun die gleiche Familie gültiger Ungleichungen $\mathcal{C}$ nicht A Priori hinzu, sondern wie in Kapitel 1.4 beschrieben im Laufe des Branch & Cut Algorithmus so erhalten wir nach dem ersten Aufruf des Schnittebenenverfahrens den in Abbildung 8 eingezeichneten relaxierte Optimalwert a'. Dies geschieht wie folgt:
Zuerst lösen wir die lineare Relaxation mit zulässiger Menge P_X und erhalten den Punkt a. Nun lösen wir das Separationsproblem $SEP(a|\mathcal{C})$ und erhalten so die verletzte Ungleichung $y_1+y_2 \leq 6$. Diesen identifizierten Schnitt fügen wir nun zu unserer Formulierung hinzu und lösen wieder die lineare Relaxation um Punkt a' zu erhalten. In diesem Prozess haben wir also das Dreieck aga' von der zulässigen Menge abgeschnitten. Allerdings bleibt dadurch die Menge X unverändert. Nach Lösen des Separationsproblems $SEP(a'|\mathcal{C})$ stellen wir fest dass keine weiteren Ungleichungen der Familie $\mathcal{C}$ verletzt werden, also haben wir für unsere aktuelle Formulierung das Optimum gefunden. Da die Lösung aber nicht ganzzahlig ist branchen wir nun mit unserer beschnittenen Formulierung weiter. Es ist leicht einzusehen dass die andere Ungleichung $y_1 - y_2 \geq 0$ aufgrund der Richtung der Optimierung nie verletzt wird. Somit werden wir im weiteren Verlauf des Branch & Cut Algorithmus keine Schnitte mehr hinzufügen können und wir erhalten ebenfalls den Branching Baum 7, wie er beim Ansatz der A priori Neuformulierung entstanden ist (vgl. [20]).

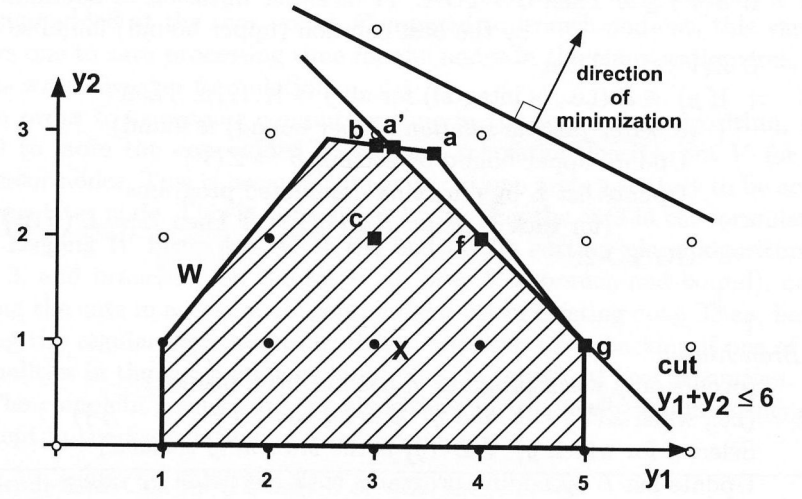

Abbildung 8: Hinzugefügter Schnitt $y_1 + y_2 \leq 6$ im ersten Schritt des Schnittebenenverfahrens

2 Losgrößenprobleme

Im Folgenden klären wir zunächst was wir unter einem Losgrößenproblem verstehen und werden dann im Anschluss eine mathematische Modellierung vornehmen.

2.1 Problemstellung

Immer wenn im Zeitverlauf Produkte zu verschiedenen Zeitpunkten nachgefragt werden stellt sich das Problem der Festlegung von Losgrößen der Produkte. Genau dies Bildung von Losen wollen wir in einem Losgrößenproblem beschreiben. Interessant sind Losgrößenprobleme vor allem wegen ihrer besonderen praktischen Relevanz da in jedem Unternehmen im Rahmen der operativen Produktionsplanung als zentrales Entscheidungsproblem erörtert werden muss welche Produkte wann in welcher Menge hergestellt werden gegeben eine prognostizierte Nachfrage. Dies ist durchaus ein schwieriges Problem, jedoch lassen sich durch Optimierung dieses Erörterungsprozesses Kosten beachtlicher Höhe sparen.
Bei der Modellierung eines Losgrößenproblems müssen wir verschiedenste Gegebenheiten beachten. Zunächst einmal wollen wir die Gesamtheit der Kosten, die mit der Produktion verbunden sind, minimieren. Diese bestehen in der Regel aus Produktionskosten, Lagerhaltungskosten und Set-Up-Kosten. Es geht also im weitesten Sinne um die Frage ob es sinnvoller ist früh zu produzieren und die Produkte dann zu lagern oder erst zu produzieren wenn die Produkte auch gebraucht werden und sie dann direkt abzusetzen. Ersteres hat den Vorteil dass die Maschine weniger oft gestartet werden muss und damit die Set-Up-Kosten klein sind, bei der zweiten Variante haben wir hingegen kleine Lagerhaltungskosten. Optimieren wir das Losgrößenproblem, so wollen wir also einen optimalen Trade-Off zwischen beidem finden.
Als Nebenbedingungen haben wir Anforderungen an die Produktion, die Nachfrage oder den Lagerbestand, wie z.B. Sicherheitsbestände oder Kapazitätsbeschränkungen an die Maschinen. Außerdem kann es noch gewisse Restriktionen bezüglich der Set-Ups der Maschinen geben. Dazu hat man in der Praxis noch sogenannte Auftragsbestände (*backlogs*), d.h. Aufträge die noch aus vorherigen Perioden stammen und die meist gegen eine gewisse Strafgebühr in folgende Perioden „mitgenommen“ werden dürfen.

2.2 Mathematische Modellierung

Die Modellierung eines Losgrößenproblems hat, wie schon im vorherigen Kapitel erwähnt, entscheidenden Einfluss auf die Laufzeit des Branch & Cut Algorithmus und die Güte der erhaltenen Lösung. Daher ist es unerlässlich die gegebenen Bedingungen von Anfang an vernünftig zu formulieren.

Wir stellen nun im Folgenden unser mathematisches Modell zu einem Losgrößenproblem Schritt für Schritt auf. Unser Losgrößenproblem habe dazu NI Produkte, die auf NK Maschinen produziert werden können und einen Planungshorizont von NT Perioden.

Die Basisdaten sind mit der Notation von [4] gegeben durch:

d_t^i ist die Nachfrage nach Produkt i in Periode t.

ρ_t^{ik} gibt die Produktionsrate von Produkt i auf Maschine k in Periode t an.

C_t^{ik} ist die maximale Produktion von Produkt i auf Maschine k in Periode t.

L_t^k steht für die Gesamtkapazität von Maschine k.

a_t^{ik} ist die benötigte Kapazität von Produkt i auf Maschine k in Periode t.

b_t^{ik} ist die benötigte Fixkapazität falls Maschine k für Produkt i in Periode t eingerichtet ist.

p_t^{ik} sind die Produktionskosten von Produkt i in Periode t auf Maschine k

f_t^{ik} sind die Set-Up-Kosten von Produkt i in Periode t auf Maschine k

h_t^i sind die Lagerhaltungskosten von Produkt i in Periode t

Unsere Basisvariablen definieren wir wie folgt:

s_t^i ist Lagerbestand von Produkt i in Periode t.

r_t^i ist Auftragsbestand/-überhang (*backlog*) von Produkt i in Periode t.

x_t^{ik} ist Produktion von Produkt i auf Maschine k in Periode t.

y_t^{ik} ist Set-up Variable von Produkt i auf Maschine k in Periode t.

Wir betrachten ein einstufiges Mehrprodukt, Mehrmaschinen Problem (*multi-item, multi-machine single level problem*), bei dem ein oder mehrere Produkte auf jeder Maschine während einer Periode hergestellt werden. Einstufig bedeutet dabei, dass wir keine Zwischenprodukte, sondern nur Endprodukte in unserem Modell haben. Mit der Überlegung von oben ergibt sich folgende Zielfunktion:

$$\min \sum_{t=1}^{NT} \sum_{i=1}^{NI} (h_t^i s_t^i + \sum_{k=1}^{NK} p_t^{ik} x_t^{ik} + f_t^{ik} y_t^{ik})$$

Dazu können folgende aufgestellt werden:

$$s_{t-1}^i - r_{t-1}^i + \sum_k \rho_t^{ik} x_t^{ik} = d_t^i + s_t^i - r_t^i \quad \text{für alle } i,t, \tag{1}$$

$$x_t^{ik} \leq C_t^{ik} y_t^{ik} \quad \text{für alle } i,k,t, \tag{2}$$

$$\sum_i a_t^{ik} x_t^{ik} + \sum_i b_t^{ik} y_t^{ik} \leq L_t^k \quad \text{für alle } k,t, \tag{3}$$

$$x, s, r \geq 0, \quad y \in \{0,1\}. \tag{4}$$

Gleichung 1 ist die Lagerhaltungsgleichung (*flow balance constraint*), die gewährleistet dass die Summe der Nachfrage und der Differenz (Bestand - Auftragsbestand) der aktuellen Periode mit der Summe aus Produktion der aktuellen Periode und Differenz (Bestand

- Auftragsbestand) der letzten Periode übereinstimmt. Diese lässt sich wie in Abbildung 9 als Fluss darstellen (mit $NK = 1$, $\rho_t^i = 1$ und ohne Backlogs) (vgl. [20]). Dabei muss in jeden Knoten genauso viel hineinfließen wie wieder herausfließt.

Gleichung 2 ist die sogenannte VUB Ungleichung (*variable upper bound constraint*), die sicherstellt dass nur produziert wird falls die Maschine auch zur Produktion bereitgestellt wurde.

Gleichung 3 ist die Maschinenkapazitätsungleichung (*machine capacity constraint*).

Gleichung 4 ist die Nichtnegativitätsbedingung

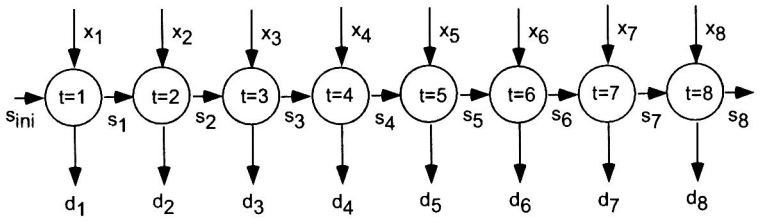

Abbildung 9: Die Flusserhaltungsgleichung

Da in der Praxis zur Bereitstellung einer Maschine häufig Kosten verursacht werden bzw. Zeit verbraucht wird, führen wir folgende zusätzliche Variablen ein:

z_t^{ik} ist die Start-up Variable, es gilt $z_t^{ik} = 1$ genau dann wenn Maschine k in Periode t zur Produktion von Produkt i läuft und in Periode i-1 nicht lief (also $y_t^{ik} = 1$ und $y_{t-1}^{ik} = 0$)

w_t^{ik} ist die Switch-off Variable, es gilt $w_t^{ik} = 1$ genau dann wenn Maschine k in Periode t zur Produktion von Produkt i läuft und in Periode i+1 nicht läuft (also $y_t^{ik} = 1$ und $y_{t+1}^{ik} = 0$)

Damit erhalten wir folgende zusätzliche Nebenbedingungen:

$$z_{t+1}^{ik} - w_t^{ik} = y_{t+1}^{ik} - y_t^{ik} \quad \text{für alle } i, k, t, \tag{5}$$

$$z_t^{ik} \leq y_t^{ik} \quad \text{für alle } i, k, t, \tag{6}$$

$$z, y \in \{0, 1\}. \tag{7}$$

Beweis.
1. Fall: $y_{t+1}^{ik} = 1 = y_t^{ik} \Longrightarrow z_{t+1}^{ik} = 0 = w_t^{ik}$ und $0 - 0 = 1 - 1$
2. Fall: $y_{t+1}^{ik} = 0 = y_t^{ik} \Longrightarrow z_{t+1}^{ik} = 0 = w_t^{ik}$ und $0 - 0 = 0 - 0$
3. Fall: $y_{t+1}^{ik} = 1$ and $y_t^{ik} = 0 \Longrightarrow z_{t+1}^{ik} = 1, \quad w_t^{ik} = 0$ und $1 - 0 = 1 - 0$
4. Fall: $y_{t+1}^{ik} = 0$ and $y_t^{ik} = 1 \Longrightarrow z_{t+1}^{ik} = 0, \quad w_t^{ik} = 1$ und $0 - 1 = 0 - 1$

□

Oftmals findet man die Nebenbedingungen 5 in der Literatur auch in der alternativen Form

$$z_{t+1}^{ik} \geq y_{t+1}^{ik} - y_t^{ik} \quad \text{für alle } i, k, t, \tag{8}$$

bzw

$$w_t^{ik} \geq y_t^{ik} - y_{t+1}^{ik} \quad \text{für alle } i, k, t, \tag{9}$$

Darüberhinaus nimmt Nebenbedingung 3 bei Anwesenheit von Start-Up's oft die einfache Form

$$\sum_i y_t^{ik} \leq 1 \quad \text{für alle } k, t, \tag{10}$$

an mit $L_t^k = 1 \quad$ für alle k, t und $a_t^{ik} = 0, b_t^{ik} = 1 \quad$ für alle i, k, t.

3 Gültige Ungleichungen

Im Folgenden werden wir uns nun schrittweise gültige Ungleichungen bzw. Neuformulierungen für gegebene Nebenbedingungen herleiten. Dabei gehen wir zunächst nochmals auf den Unterschied zwischen dem Einfügen von gültigen Ungleichungen a Priori und dem Einfügen während des Branch & Cut Algorithmus' ein. Danach betrachten wir Neuformulierungen für Sicherheitsbestände und Probleme mit Zwischenprodukten. Gefolgt werden diese Abschnitte von einer etwas anderen Sichtweise, die Variablen für Start-Ups und Changeovers der Maschinen beinhaltet. Dabei werden wir zwischen einem *„Small Bucket Model“* und einem *„Big Bucket Model“* unterscheiden. Abschließend befassen wir uns mit Neuformulierungen für minimale Produktionsläufe und Produktion unter Vollauslastung. Zusätzlich dazu existieren allgemein gültige Ungleichungen wie z.B. die Gomory, Flow Cover oder Lift-and-Project Ungleichungen, deren Herleitung der interessierte Leser in [11], [21] und [12] nachlesen kann.

3.1 Neuformulierung A Priori

3.1.1 Unkapazitiertes Modell

Betrachte zunächst nur ein Produkt auf einer Maschine ohne Backlogging. Wir erhalten

$$s_{t-1} + x_t = d_t + s_t \quad \text{für alle } t, \tag{11}$$

$$x_t \leq C_t y_t \quad \text{für alle } t, \tag{12}$$

$$x, s \geq 0, \quad y \in \{0, 1\}.$$

Ignoriert man nun die Werte von C_t und betrachtet das Problem somit als unbeschränkt, so können wir die Nachfrage im Intervall $[t, ..., l]$ wie in Abbildung 10 betrachten (vgl. [3]).

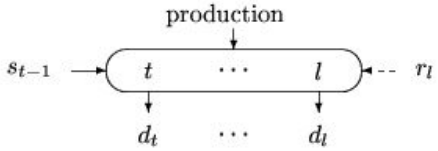

Abbildung 10: Nachfrage im Intervall $[t, ..., l]$

Um die aggregierte Nachfrage $d_{tl} = \sum_{\tau=t}^{l} d_\tau$ zu befriedigen gibt es nur die Möglichkeit in diesem Intervall zu produzieren oder aus dem Eingangsbestand s_{t-1} zu zehren. Somit ergeben sich die gültigen Ungleichungen

$$s_{t-1} \geq d_{tl}(1 - y_t - y_{t+1} - ... - y_l) \tag{13}$$

Von diesen Ungleichungen gibt es für jede Wahl von t genau t Ungleichungen. Damit erhalten wir

$$\sum_{t=1}^{NT} t = \frac{NT(NT+1)}{2} \in O(NT^2)$$

Ungleichungen welche A Priori zum Modell hinzugefügt werden können.
In der Praxis bewirkt allerdings schon das Hinzufügen der NT gültigen Ungleichungen

$$s_{t-1} \geq d_t(1 - y_t) \quad \text{für alle } t = 1, ..., NT \tag{14}$$

entstanden durch simples Setzen von $l = t$ in obiger Gleichung (13), enorme Verbesserungen in der Qualität der unteren Schranke des Werts der LP Relaxation (vgl. [4]).

Betrachten wir noch den Fall der Anwesenheit von Start-Up's und Backlogging-Variablen.

Haben wir Start-Up's gegeben so beobachten wir dass $y_t + y_{t+1} + ... + y_l = 0$ genau dann wenn $y_t + z_{t+1} + ... + z_l = 0$. So lässt sich Gleichung (13) auch schreiben als

$$s_{t-1} \geq d_{tl}(1 - y_t - z_{t+1} - ... - z_l) \tag{15}$$

Zusätzlich sehen wir aber dass der Startbestand der betrachteten Periode $[t, t+1, ..., l]$ die Nachfrage d_τ enthalten muss falls $y_t + y_{t+1} + ... + y_\tau = y_t + z_{t+1} + ... + z_\tau = 0$. So erhalten wie die stärkere gültige Ungleichung

$$s_{t-1} \geq \sum_{\tau=t}^{l} d_\tau(1 - y_t - z_{t+1} - ... - z_\tau). \tag{16}$$

Haben wir zusätzlich Auftragsbestände (*backlogging*) gegeben so lassen sich (15) und (16) unter Benutzung der Backlogging-Variablen r_τ verallgemeinern zu

$$s_{t-1} + r_l \geq d_{tl}(1 - y_t - z_{t+1} - ... - z_l) \tag{17}$$

und

$$s_{t-1} + \sum_{\tau=t}^{l} r_\tau \geq \sum_{\tau=t}^{l} d_\tau(1 - y_t - z_{t+1} - ... - z_\tau) \tag{18}$$

Zuletzt erlauben wir nun mehrere Maschinen und erhalten als direkte Verallgemeinerung von (18)

$$s_{t-1}^i + \sum_{\tau=t}^{l} r_\tau^i \geq \sum_{\tau=t}^{l} d_\tau^i [1 - \sum_{k=1}^{NK} (y_t^{i,k} - z_{t+1}^{i,k} - ... - z_\tau^{i,k})] \tag{19}$$

(vgl. [3]).

3.1.2 Kapazitiertes Modell

Für den Fall dass mit $C = \max_t C_t$ eine Beschränkung vorliegt erhält man noch schärfere Ungleichungen.
Aggregiert man z.B. die Flusserhaltungsnebenbedingungen über dem oben betrachteten Interval $[t, l]$ und schätzt mit den VUB Ungleichungen ab

$$\sum_{j=k}^{t} s_{j-1} + x_j = \sum_{j=k}^{t} d_j + s_j$$

$$\leq C \sum_{j=k}^{t} y_j + \sum_{j=k}^{t} s_{j-1}$$

so erhalten wir folgende Ungleichung

$$s_{t-1} + C \sum_{\tau=t}^{l} y_\tau \geq d_{tl} \tag{20}$$

$$s_{k-1} \geq 0 \quad \text{und} \quad \sum_{j=k}^{t} y_j \in \mathbb{Z}$$

Halten wir nun t fix und variieren $l \geq t$ so erhalten wir für diese Menge von Ungleichungen die gemischt-ganzzahlige Rundungs-Ungleichung (*MIR = mixed-integer rounding inequality*) (vgl. [16])

$$s_{t-1} \geq \rho_{tl}(\eta_{tl} - \sum_{\tau=t}^{l} y_\tau) \tag{21}$$

$$\eta_{tl} = \lceil d_{tl}/C \rceil \quad \text{und} \quad \rho_{tl} = d_{tl} - C(\eta_{tl} - 1)$$

Da für jedes t genau t solcher Ungleichungen existieren erhalten wir nach der Rechnung von oben wieder $O(NT^2)$ Ungleichungen die A Priori hinzugefügt werden können.
Darüberhinaus zeigt [13] wie weitere MIR-Ungleichungen aus dem 0-1 Knapsack Problem hergeleitet werden können.
Wählen wir nun eine Teilmenge $Q \subseteq \{t, t+1, ..., NT\}$ der Größe p und sortieren die Elemente von Q nach den nichtfallenden Werten von ρ_q, z.B. sei $Q = \{q_1, ..., q_p\}$ mit $0 < \rho_{q_1} \leq ... \leq \rho_{q_p} \leq C$. Dann lässt sich Gleichung 21 verallgemeinern zu

$$s_{t-1} \geq \sum_{u=1}^{p} (\rho_{q_u} - \rho_{q_{u-1}})(\eta_{q_u} - \sum_{\tau=t}^{q_u} y_\tau) \tag{22}$$

Für den Fall eines Mehr-Maschinen Problems mit Backlogging lässt sich diese Ungleichung auch verallgemeinern, siehe dazu [3].

3.2 Schnittebenen

Wie schon in Kapitel 1.3 beschrieben lassen sich in Branch & Cut basierten Systemen wie z.B. CPLEX, XPRESS, MINTO oder BC-OPT die hergeleiteten zulässigen Ungleichungen A Priori hinzufügen. Eine Alternative dazu ist jedoch die automatische Generierung von Schnittebenen, die diese Systeme beherrschen. Dabei erkennen sie gewisse Nebenbedingungen und kombinieren die Ungleichungen wie im vorherigen Abschnitt vorgeführt.
Als Beispiel für eine solche Schnittebene könnte das Kombinieren von (11) mit $t = k$ mit der VUB-Ungleichung (12) sein. Dabei wird auch die Nichtnegativität der s_k benutzt. Aus der erhaltenen Ungleichung

$$s_{k-1} + C_k y_k \geq d_k \tag{23}$$

$$s_{k-1} \geq 0, \quad y_k \in \{0, 1\}, \tag{24}$$

generieren die Systeme automatisch Ungleichung (14) und durch weiteres Kombinieren auch die Ungleichungen (13) und (21).

Sowohl Ungleichung (13) als auch (21) sind Teil einer exponentiellen Familie von Ungleichungen. So ist (13) ein Teil der folgenden exponentiellen Familie

$$s_{k-1} + \sum_{j \in \{k,...,t\} n T} x_j \geq \sum_{j=k}^{t} d_j (1 - \sum_{i \leq j, i \in T} y_i) \tag{25}$$

$$\text{für alle } k, t, T \quad \text{mit} \quad k \leq t \quad \text{und} \quad T \subseteq \{k, ..., t\}.$$

Für den kapazitierten Fall kann man diese Ungleichungen auch verallgemeinern siehe [18] oder [2].
Damit solche Systeme möglichst viele solcher Schnittebenen generieren ist es wichtig, dass eine den Systemen angepasste Formulierung zu Grunde liegt. Wenn z.B. die Flusserhaltungsgleichung 11 in der Form

$$\sum_{j=1}^{t} x_j \geq d_{1t} \quad \text{für alle } t \tag{26}$$

vorliegt, so wird keines der genannten Systeme die Ungleichung (14) generieren (vgl. [4]).

3.3 Sicherheitsbestände - Nettonachfragen

Als Nächstes beziehen wir nun Sicherheitsbestände in unsere Modellierung mit ein. Dazu definieren wir uns unseren Startbestand mit $\underline{S}_0$ und positive Sicherheitsbestände zum Ende von Periode t mit $\underline{S}_t$.

$$\underline{S}_0 + x_1 = d_1 + s_1 \tag{27}$$

$$s_{t-1} + x_t = d_t + s_t \quad \text{für alle } t = 2, ..., NT \tag{28}$$

$$s_t \geq \underline{S}_t \quad \text{für alle } t = 1, ..., NT \tag{29}$$

Nun ist Ungleichung (14) immer noch gültig, aber es ist unwahrscheinlich dass sie verletzt wird da $s_{t-1} \geq \underline{S}_{t-1}$ (vgl. [3]).
Eine Neuformulierung können wir nun durchführen indem wir uns sogenannte Nettonachfragen $\tilde{d}_t$ definieren. Dazu setzen wir zunächst
$\underline{S}_t := \max\{\underline{S}_t, \underline{S}_{t-1} - d_t\}$ für alle $t = 1, ...NT$.
Nun definieren wir uns $\tilde{s}_t = s_t - \underline{S}_t \geq 0$ und die Nettonachfragen $\tilde{d}_t = d_t + \underline{S}_t - \underline{S}_{t-1}$. Damit erhalten wir

$$\tilde{s}_{t-1} + x_t = \tilde{d}_t + \tilde{s}_t, \quad \tilde{s}_t \geq 0.$$

Ungleichung (14) nimmt nun folgende Form an

$$\tilde{s}_{t-1} \geq \tilde{d}_t(1 - y_t)$$

Diese könnte durchaus verletzt sein, auch wenn die ursprüngliche Ungleichung es nicht war (vgl. [4]).

Darüberhinaus könnte man noch eine weitere Änderung vornehmen falls für manche t $\tilde{d}_t > C_t$ gilt. Indem man sich von $t = NT, ..., 2$ zurückarbeitet sehen wir dass $\tilde{s}_{t-1} \geq \hat{s}_{t-1} = (\tilde{d}_t + \hat{s}_t - C_t)^+$mit $\hat{s}_{NT} = 0$ gilt. Danach arbeiten wir wie oben um die unteren Schranken $\hat{s}_t$ auf den neuen Bestandsvariablen $\tilde{s}_t$ zu eliminieren.

3.4 Zwischenprodukte - Staffelbestände

Da in der Praxis im Herstellungsprozess oft Zwischenprodukte auftreten, wollen wir im Folgenden auch den Bestand dieser in unsere Überlegungen miteinbeziehen. Diese zusätzliche Anforderung macht das Modell sehr komplex. In der Praxis wendet man dann meist eine Dekomposition des Problems an. Ähnlich gehen wir auch vor.
Zur Vereinfachung betrachten wir ein Produktionssystem in dem jedes Produkt genau eine Einheit seines Vorgängerprodukts benötigt. Genauso gut kann man die folgenden Transformationen aber auch auf beliebige Produktionsstrukturen anwenden.
Wir definieren uns zunächst den Vorgänger von (Zwischen-)Produkt i als $\sigma(i)$ wobei unser

Endprodukt 1 sei. Die Flusserhaltungsgleichung ändert sich dann zu

$$s_{t-1}^i + x_t^i = x_t^{\sigma(i)} + s_t^i \quad \text{für alle } i = 2, ..., NI \; t = 1, ..., NT \tag{30}$$

$$s_{t-1}^1 + x_t^1 = d_t^1 + s_t^1 \quad \text{für alle } t = 1, ..., NT \tag{31}$$

Nun könnte man durch Kombinieren dieser Gleichungen mit der VUB-Ungleichung $x_i \leq C_t^i y_t^i$ für $i = 1$ Ungleichung (13) herleiten. Allerdings ist es für alle anderen Zwischenprodukte nicht möglich diese gültigen Ungleichungen zu generieren. Aufgrund dieser Tatsache führen wir sogenannte Staffelbestände(*echelon stocks*) ein um dieses Problem zu umgehen (vgl. [4]).

Definition 3.1.
Der Staffelbestand eines Produkts i zur Zeit t ist dabei genau der gesamte Bestand im System, also der eigene Bestand plus alle Bestände von Nachfolgeprodukten die Produkt i beinhalten (vgl. [5]).

$$e_t^1 = s_t^1 \quad \textit{und} \quad e_t^i = s_t^i + e_t^{\sigma(i)} \quad \textit{für alle } i \geq 2, \quad t = 1, ..., NT$$

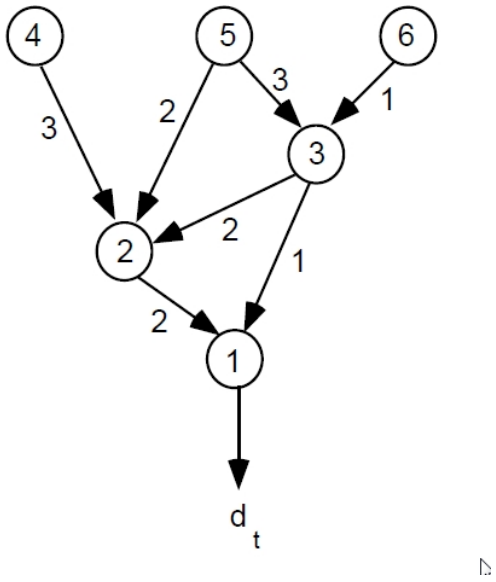

Abbildung 11: Ein System mit Zwischenprodukten

In Abbildung 11 ist beispielhaft ein solches System mit Zwischenprodukten dargestellt (vgl. [17]). Die Staffelbestände in diesem Beispiel sind gegeben durch

$$
\begin{aligned}
e_t^1 &= s_t^1 \\
e_t^2 &= s_t^2 + 2e_t^1 = s_t^2 + 2s_t^1 \\
e_t^3 &= s_t^3 + 2e_t^2 = s_t^3 + 2s_t^2 + 5s_t^1 \\
e_t^4 &= s_t^4 + 3e_t^2 = s_t^4 + 3s_t^2 + 6s_t^1 \\
e_t^5 &= s_t^5 + 2e_t^2 + 3e_t^3 = s_t^5 + 3s_t^3 + 8s_t^2 + 19s_t^1 \\
e_t^6 &= s_t^6 + e_t^3 = s_t^6 + s_t^3 + 2s_t^2 + 5s_t^1
\end{aligned}
$$

Gleichung (30) und (31) können nun umgeschreiben werden zu

$$e_{t-1}^i + x_t^i = d_t^i + e_t^i \quad \text{für alle } i,t \tag{32}$$

$$e_t^i \geq e_t^{\sigma(i)} \quad \text{für alle } i,t \tag{33}$$

wobei sich Gleichung (33) aus der Nichtnegativität der s_t^i ergibt.
Nun ist es möglich die gültige Ungleichung (14)

$$e_{t-1}^i \geq d_t^1(1 - y_t^i)$$

für alle Produkte herzuleiten. Bezüglich der alten Variablen lässt sich die Ungleichung folgendermaßen schreiben

$$\sum_{j \in P(i)} s_{t-1}^j \geq d_t^1(1 - y_t^i)$$

wobei $P(i)$ die Menge aller Produkte ist die Produkt i enthalten.

3.5 Startups und Changeovers

Im Weiteren gehen wir nun auf Nebenbedingungen für Start-Ups und Changeovers ein. Dabei findet auf einer Maschine ein Changeover von i nach j genau dann statt wenn einem Startup von Produkt j ein unmittelbarer switch-off von Produkt i voraus geht. Wenn wir nun im Folgenden unser Modell beschreiben so ist eine entscheidende Größe die Wahl der betrachtete Zeitintervalle. Betrachten wir größere Zeitintervalle so sprechen wir von einem Big-Bucket-Modell, in dem viele Ereignisse pro Periode eintreten können. Der gegengesetzte Fall ist das Small-Bucket-Modell. Hier sind die Zeitintervalle kurz und es kann vergleichsweise wenig pro Intervall geschehen (vgl. [4]).

3.5.1 Small-Bucket Modelle

Zunächst betrachten wir ein Modell in dem nur ein oder zwei Produkte pro Periode hergestellt werden können. Unser Ziel ist es dabei enge Formulierungen für Startups und Changeovers zu finden. Können wir nur ein Produkt pro Periode produzieren so lässt sich die Produktionssequenz einfach schreiben. Für $NT = 6$ gibt die Sequenz {1}{2}{4}{3}{3}{6} an dass die Maschine in Periode 1 für Produkt 1 eingerichtet wurde, in Periode 2 für Produkt 2, usw. Diese Situation lässt sich einfach modellieren durch die Nebenbedingungen (5), (6) und (10), eingeschränkt auf eine Maschine und mit der zusätzlichen Ungleichung

$$z_t^i \leq 1 - y_{t-1}^i \quad \text{für alle } i, t$$

Erlauben wir nun die Produktion von 2 Produkten so lassen wir dabei nur einen Produktwechsel pro Periode zu. Eine zulässige Sequenz {12}{23}{3}{34}{42}{2} für $NT = 6$ gibt nun an dass in Periode 1 die Produkte 1 und 2 produziert werden, in Periode 2 die Produkte 2 und 3 usw. Schreiben wir das System nun um zu {12}{23}{33}{34}{42}{2} so erkennen wir dass Produkt 2 in Periode t-1 gleich Produkt 1 in Periode t sein muss. Folgende Ungleichungen beschreiben nun zusammen mit (5) diese Sequenzen:

$$\sum_i y_t^i \leq 1 + \sum_i z_t^i \quad \text{für alle } t \tag{34}$$

$$\sum_i z_t^i \leq 1 \quad \text{für alle } t \tag{35}$$

$$z_t^i + z_{t-1}^i \leq y_t^i \quad \text{für alle } i, t \tag{36}$$

Um nun zu verstehen ob diese Formulierung verbessert werden kann betrachten Karmarkar und Schrage ein Modell, das sequenzabhängige Changeovers beinhaltet (vgl. [9]).
Dabei wird angenommen dass ein Dummy-Produkt existiert, welches für den Stillstand der Maschine steht. Betrachte nun das Polyeder Q der Flüsse von Produkt i zu Produkt j bezüglich der Zeit t

$$\sum_i \chi_t^{ij} = q_t^j \quad \text{für } j = 1, ..., NI \quad t = 1, ..., NT \tag{37}$$

$$\sum_j \chi_t^{ij} = q_{t-1}^i \quad \text{für } i = 1, ..., NI \quad t = 1, ..., NT \tag{38}$$

$$\sum_i q_0^i = 1, \tag{39}$$

$$\chi_t^{ij} \geq 0 \quad \text{für } i, j = 1, ..., NI \quad t = 1, ..., NT \tag{40}$$

Dabei ist q_t^i der Fluss durch Knoten (i, t) und χ_t^{ij} der Fluss von Knoten $(i, t-1)$ zu Knoten (j, t). Dies die engstmögliche Formulierung in dem Sinne, dass falls keine anderen Nebenbedingungen vorhanden sind, wir eine ganzzahlige Lösung bekommen (vgl. [4]). Ohne die Variablen $\chi_t^{i,j}$ mit $i \neq j$ ergibt sich folgendes Resultat

Proposition 3.2 (Constantino 1996)**.** *:*
Für das Polyeder Q ist $proj_{q,\chi^{ii}}(Q)$ *gegeben durch*

$$\sum_t q_t^i = 1 \qquad \text{für } t = 0, ..., NT \tag{41}$$

$$q_t^i \geq \chi_t^{ii} \qquad \text{für } i = 1, ...NI \quad t = 1, ..., NT \tag{42}$$

$$q_{t-1}^i \geq \chi_t^{ii} \qquad \text{für } i = 1, ...NI \quad t = 1, ..., NT \tag{43}$$

$$q_t^i + q_{t-1}^i + \sum_{i:j \neq i} \chi_t^{jj} \leq 1 + \chi_t^{ii} \qquad \text{für } i = 1, ...NI \quad t = 1, ..., NT \tag{44}$$

$$\chi_t^{ii} \geq 0 \qquad \text{für } i = 1, ...NI \quad t = 1, ..., NT \tag{45}$$

eine verstärkende Neuformulierung (vgl. [6]).

Beweis. siehe [6] □

Im Folgenden interpretieren wir diese Proposition für die Fälle einer bzw zweier Setups pro Periode (vgl. [3]).

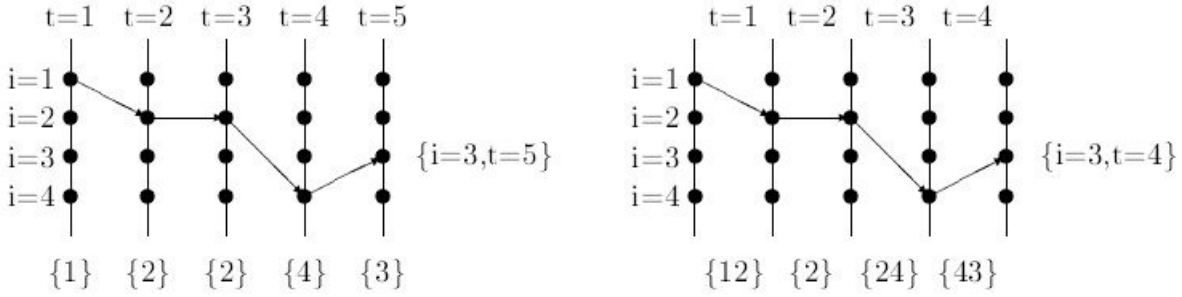

Abbildung 12: Small-Bucket Modell: 1 und 2 Set-Ups

3.5.1.1 1 Set-Up pro Periode Um nun Proposition 3.2 nutzen zu können müssen wir unsere Set-up und Start-up Variablen adäquat definieren um das Polyeder Q darstellen zu können.
Den Fluss durch Knoten (j,t) können wir nun so interpretieren dass ein Set-up von Produkt j in t vorliegt. Also setzen wir $y_t^j = q_t^j$. Als Nächstes betrachten wir den Fluss von Knoten $(i,t-1)$ nach (j,t). Nun gilt $\chi_t^{ij} = 1$ genau dann wenn $y_{t-1} = y_t^j$. Also sind unsere Start-up und Switch-Off Variable definiert durch

$$z_t^i = \sum_{j:j\neq i} \chi_t^{ji} = y_t^i - \chi_t^{ii}$$

$$w_t^i = \sum_{j:j\neq i} \chi_{t+1}^{ij} = y_t^i - \chi_{t+1}^{ii}$$

Damit haben wir unser Polyeder Q definiert und können Proposition 3.2 anwenden um eine stärkere Formulierung für Set-Ups, Start-Ups und Switch-Offs zu erhalten:

$$\sum_i y_t^i = 1 \qquad \text{für alle } t = 0,...,NT \tag{46}$$

$$z_t^i \leq y_t^i \qquad \text{für alle } i = 1,...,NI,\ t = 1,...,NT \tag{47}$$

$$y_t^i - z_t^i = y_{t-1}^i - w_{t-1}^i \qquad \text{für alle } i = 1,...,NI,\ t = 1,...,NT \tag{48}$$

$$y_{t-1}^i + z_t^i + \sum_{j:j\neq i} (y_t^j - z_t^j) \leq 1 \qquad \text{für alle } i = 1,...,NI,\ t = 1,...,NT \tag{49}$$

$$z_t^i, w_t^i \geq 0 \qquad \text{für alle } i = 1,...,NI,\ t = 1,...,NT \tag{50}$$

Dabei folgt (46) aus (41), (50) mit Schlupfvariablen z_t^i und w_{t-1}^i aus (42) und (43). Gleichung (48) folgt durch Eliminierung der χ_t^{ij} in der Definition von z_t^i und w_t^i, Ungleichung (47) aus (45 und der definition der z_t^i und zu guter Letzt (49)) aus (44) nach Substitution. Nun erhalten wir mit Ungleichung (49) einen Schnitt der die Formulierung am Anfang des Kapitels verstärkt (vgl. [6]). Interpretativ sagt er dass genau eine der 3 Möglichkeiten *„es besteht ein setup für Produkt i in Periode t-1“*, *„es besteht ein Startup für Produkt i in Periode t“* und *„ein anderes Produkt als i wird sowhl in $t-1$ als auch t produziert“* eintreten kann (vgl. [3]).

3.5.1.2 2 Set-Ups pro Periode Wieder wollen wir nun Proposition 3.2 anwenden, dieses Mal allerdings für den Fall zweier Setups pro Periode. Für diesen Fall müssen wir die Variablen jedoch anders interpretieren. Dazu betrachten wir nochmals Abbildung 12.
Es existiert genau dann ein Setup für Produkte i und j wenn $q^i_{t-1} = g^j_t = \chi^{ij}_t = q^i_t - \chi^{ii}_t$. Also bedeutet $q^j_t = 1$ dass die Maschine am Ende von Periode t für Produkt j aufgestellt ist. Dazu definieren wir unsere Start-Up Variable durch

$$z^i_t = \sum_{j:j\neq i} \chi^{ji}_t = q^i_t - \chi^{ii}_t$$

und unsere Switch-Off Variable durch

$$w^i_t = \sum_{j:j\neq i} \chi^{ij}_t = q^i_{t-1} - \chi^{ii}_t.$$

Als Set-Up Variable für Produkt i und Periode t leiten wir

$$y^i_t = q^i_{t-1} + q^i_t - \chi^{ii}_t = q_{t-1} + z^i_t = w^i_t + z^i_t + \chi^{ii}_t$$

her. Mit diesen Gleichheiten und $\text{proj}_{q,\chi^{ii}}(Q)$ erhalten wir folgende Formulierung:

$$\sum_i (y^i_t - z^i_t) = 1 \qquad \text{für alle } t = 1, ..., NT, \tag{51}$$

$$z^i_t + z^i_{t-1} \leq y^i_t \qquad \text{für alle } i = 1, ..., NI, \quad t = 1, ..., NT, \tag{52}$$

$$y^i_t - z^i_t = y^i_{t-1} - w^i_{t-1} \qquad \text{für alle } i = 1, ..., NI, \quad t = 1, ..., NT, \tag{53}$$

$$y^i_t + \sum_{j:j\neq i} (y^j_{t+1} - z^j_{t+1} - z^j_t) \leq 1 \qquad \text{für alle } i = 1, ..., NI, \quad t = 1, ..., NT, \tag{54}$$

$$z^i_t, w^i_t \geq 0 \qquad \text{für alle } i = 1, ..., NI, \quad t = 1, ..., NT. \tag{55}$$

Dabei folgt (51) aus (41) und den Definitionen von y^i_t und z^i_t, (52) aus (45) und den Definitionen von z^i_t und y^i_t, (55) mit Schlupfvariablen z^i_t und w^i_t aus (42) und (43). Gleichung (53) rührt wieder von den Definitionen von z^i_t und w^i_{t+1}, die genutzt wurden um q^i_t zu eliminieren und danach die Definition von y^i_t um χ^{ii}_t zu eliminieren. Außerdem Ungleichung (54) von (44) mit geeigneten Substitutionen.
Hier ist nun (54) ein Schnitt der unsere ursprüngliche Formulierung verstärkt. Die Ungleichung sagt aus dass genau eins der 2 Ereignisse *„es exisitiert ein Setup für Produkt i in Periode t“* und *„es existiert ein Setup für irgendein anderes Produkt als i“* eintritt (vgl. [4]).

3.5.2 Big Bucket Modell

Im Folgenden betrachten wir eine andere Situation, in der Setups für mehrere Produkte pro Periode erlaubt sind. Dabei wollen wir die Changeover Kosten modellieren.
Wir gehen zunächst von einer Periode aus. Sei das erste produzierte Produkt $i = 0$ und das letzte $i = n + 1$. Dazu sei y^i die Anzahl der Setups von Produkt i in der Periode, χ^{ij} die Anzahl der Wechsel von Produkt i auf Produkt j und q eine obere Schranke für die y_i. Haben wir als Beispiel die Setup Sequenz $1, 2, 2, 3, 4, 2, 3$ so nehmen unsere Variablen folgende Werte an

$$y^1 = y^4 = 1, y^2 = 3, y^3 = 2 \quad \text{und}$$
$$\chi^{01} = \chi^{12} = \chi^{22} = \chi^{34} = \chi^{42} = \chi^{35} = 1, \chi^{23} = 2$$

Das Big-Bucket Modell kann durch folgende Formulierung angegeben werden

$$\sum_{j \neq 0} \chi^{0j} = 1 \tag{56}$$

$$\sum_{j} \chi^{ij} = y^i \tag{57}$$

$$\sum_{i} \chi^{ij} = y^j \tag{58}$$

$$\sum_{i \neq n+1} \chi^{i,n+1} = 1 \tag{59}$$

$$y_i, \chi^{ij} \in \{0, 1, ..., q\} \quad \text{für alle } (i, j), \tag{60}$$

Die Modellierung erinnert nun stark an das Travelling Salesman Problem(TSP), siehe [16]. Der Unterschied dazu ist nur, dass wir hier mit ganzzahligen Variablen umgehen müssen wohingegen wir beim TSP nur Binärvariablen haben.

Proposition 3.3.

1. *Ein Vektor (y, χ) der obigen Anforderungen (56)-(60) genügt steht für eine zulässige Produktionssequenz genau dann wenn im induzierten Multigraph kein unzusammenhängender direkter Eulerscher Teilgraph existiert.*
2. *Die zulässige Ungleichung*

$$\sum_{(i,j) \in E(S)} \chi^{ij} \leq \sum_{i \in S} y^i - \frac{1}{q} y^k \quad \textit{für } k \in S \tag{61}$$

eliminiert solch einen Multigraph auf der Knotenmenge $S \subset \{1, ..., n\}$, wobei E(S) für die Menge aller Paare (i, j) steht mit $i, j \in S$ (vgl. [3]).

3.6 Minimale Produktionsläufe - Produktion unter Vollauslastung

Wir zeigen im Folgenden wie man Neuformulierungen für typische Nebenbedingungen von Small-Bucket Modellen herleiten kann, wie sie in der Praxis auftreten. Dabei betrachten wir wieder ein Ein-Produkt-System (vgl. [19]).

3.6.1 Set-Up Zeiten

Falls auf einer Maschine für eine bestimmte Anzahl von Perioden ein Set-Up für ein Produkt bestehen soll, so erhalten wir für diese Anzahl α folgende Bedingung

$$z_\gamma \leq y_t \quad \text{für } \gamma = t - \alpha + 1, ..., t \tag{62}$$

Als engere Formulierung lassen sich

$$z_{t-\alpha+1} + ... + z_t \leq y_t, \tag{63}$$

bzw.

$$w_t + ... + w_{t+\alpha-1} \leq y_t \tag{64}$$

Haben wir nun den entgegengesetzen Fall, dass nach der Produktion eines Produkts auf der Maschine für die nächsten β Perioden kein Set-Up bestehen soll, erhalten wir

$$z_\gamma \leq 1 - y_t \quad \text{für } \gamma = t + 1, ..., t + \beta \tag{65}$$

und als engere Formulierung analog

$$z_{t+1} + ... + z_{t+\beta} \leq 1 - y_t \tag{66}$$

bzw. (67)

$$w_{t-1} + ... + w_{t-\beta} \leq 1 - y_t \tag{68}$$

3.6.2 Produktion unter Vollauslastung

Eine andere Bedingung kann nun sein, dass ein Produkt in allen Perioden, außer der ersten und letzten, bei Vollauslastung produziert werden muss. Falls also in Periode $t-1$ und t ein Set-Up auf der Maschine existiert und in keiner Periode ein Switch-Off stattfindet so muss die Produktion unter Vollauslastung laufen. Dies ergibt die Nebenbedingung

$$x_t \geq C(y_{t-1} + y_t - w_{t-1} - w_t - 1). \tag{69}$$

Diese kann wieder umformuliert werden zur engeren Formulierung

$$x_t \geq C(y_{t-1} - w_{t-1} - w_t) \tag{70}$$

Dabei geht obige Beobachtung ein.

3.6.3 Minimale Produktionsläufe und Vollauslastung

Nehmen wir nun an, dass es für ein Produkt eine minimale Produktionsmenge P gibt, die produziert werden muss. Die volle Kapazität sei C und die maximale Kapazität in einer Changeover-Periode sei $\tilde{C}$ mit $\tilde{C} < C < P$. Setzen wir

$$a = \lceil (P - \tilde{C})/C \rceil$$
$$b = \lceil P/C \rceil$$

Dann ist die kleinste Anzahl der Perioden, die benötigt wird um P zu produzieren genau $\max\{a+1, b\}$ und die höchste Anzahl ist $b+1$.
Betrachten wir nun die Ungleichung

$$\sum_{\tau=t-b}^{t} x_\tau \geq P w_t$$

so können wir dieses verbessern. Falls $a = b$ so erhalten wir für alle $t > a$ die gültige Ungleichung

$$x_{t-a} + x_t \geq [P - (a-1)C] w_t \tag{71}$$

und falls $a = b - 1$ für alle $t > b$

$$x_{t-b} + x_{t-a} + x_t \geq [P - (a-1)C] w_t. \tag{72}$$

Des weiteren betrachte den Fall, dass die minimale Produktionsmenge in einem Intervall $[t, t+1, ..., l]$ schon produziert wurde und den Gesamtbedarf dieses Intervalls, also $d_{tl} = \sum_{\tau=t}^{l} d_\tau$ übersteigt. Dieser Überschuss muss dann Bedarfe von Perioden außerhalb des Intervalls decken. Dies spiegelt sich in folgenden Ungleichungen wieder

$$s_l + r_{t-1} \geq (P - d_{tl})^+ \sum_{\tau=t+b}^{l} w_\tau \tag{73}$$

$$s_l + r_{t-1} \geq (P - d_{tl})^+ \sum_{\tau=t}^{l-b} z_\tau \tag{74}$$

(vgl. [3]).

4 Anwendung

Nachdem wir nun in Kapitel 3 für verschiedenste Gegebenheiten gültige Ungleichungen hergeleitet haben wollen wir diese nun exemplarisch an einem Beispiel aus der Praxis anwenden.

4.1 Probleminstanz b4

Wir betrachten nun ein Problem aus der Realität, nämlich die Probleminstanz b4 des Unternehmens BASF (vgl. [4]).
Dieses Problem ist ein Mehrprodukt, Mehrmaschinen Problem mit Backlogging, Säuberungszeiten CLT^i und oberen sowie unteren Schranken für die Bestände. Es ist ein Small-Bucket Modell mit maximal 2 Set-Ups pro Periode. Die spezielle Anforderung bei diesem Problem ist die untere Schranke MB^i auf die Produktionsläufe in Tagen von Endprodukten EP. Dazu soll in allen Perioden außer der Ersten und Letzten die Produktion unter voller Kapazität laufen. Dazu gibt es ein paar Zwischenprodukte IP, denen gewisse Maschinen zugewiesen sind. Die Koeffizienten RML^{ij} stehen für die Anzahl des Zwischenprodukts $i \in IP$ die zur Produktion eines Endprodukts $j \in EP$ benötigt werden.
Mit diesem Wissen ist das Modell gegeben durch:

$$\min \sum_{i,k,t} (p^i \rho^{i,k} x_t^{i,k}) + \sum_{i,t} (h^i s_t^i + e^i r_t^i)$$

unter den Nebenbedingungen

$$s_{t-1}^i + \sum_k \rho^{i,k} x_t^{i,k} = \sum_{j,k:j\neq i} \mathrm{RML}^{i,j} \rho^{j,k} x_t^{j,k} + s_t^i \tag{75}$$

$$s_{t-1}^i - r_{t-1}^i + \sum_k \rho^{i,k} x_t^{i,k} = d_t^i + s_t^i - r_t^i \quad \text{für alle } t \text{ und } i \in EP, \tag{76}$$

$$x_t^{i,k} + \mathrm{CLT}^i w_t^{i,k} \leq C^k y_t^{i,k} \quad \text{für alle } i,k,t \tag{77}$$

$$\sum_i x_t^{i,k} + \sum_i \mathrm{CLT}^i w_t^{i,k} \leq C^k \quad \text{für alle } k,t \tag{78}$$

$$w_{t-1}^{i,k} \geq y_{t-1}^{i,k} - y_t^{i,k} \quad \text{für alle } i,k,t \tag{79}$$

$$w_t^{i,k} + w_{t+1}^{i,k} \leq y_t^{i,k} \quad \text{für alle } i,k,t \tag{80}$$

$$\sum_i y_t^{i,k} - \sum_i w_t^{i,k} \leq 1 \quad \text{für alle } k,t \tag{81}$$

$$\sum_i w_t^{i,k} \leq 1 \quad \text{für alle } k,t \tag{82}$$

$$w_l^{i,k} \leq y_t^{i,k} \quad \text{für alle } k,t,t \leq l \leq t + \alpha^{i,k}, i \in EP \tag{83}$$

$$\sum_{l=t-\beta^{i,k}} \rho^{i,k} x_l^{i,k} \geq MB^i w_t^{i,k} \quad \text{für alle } k,t,i \in EP \tag{84}$$

$$x_t^{i,k} \geq C^k (y_{t-1}^{i,k} + y_t^{i,k} - w_t^{i,k} - w_{t-1}^{i,k} - 1) \quad \text{für alle } i,k,t \tag{85}$$

$$s,r,t \geq 0, y,w \in \{0,1\}$$

$$s_0^i = S_0^i,\ r_0^i = 0,\ \underline{s} \leq s \leq \overline{s},$$

wobei $\beta^{i,k} = \lceil MB^i / C^k \rceil$ und $\alpha^{i,k} = \lceil (MB^i + CLT^i - C^k)/C^k \rceil$.

Dabei sind die Gleichungen (75) und (76) die Flusserhaltungsgleichungen für die Zwischen- und Endprodukte. Ungleichungen (77) und (78) sind die VUB und die aggregierten VUB Ungleichungen und (79) die Start-Up Gleichungen. (80) bis (82) stellen die 2 Set-Ups pro Periode sicher, (83) die minimalen Produktionsläufe, (84) die minimale Batchanzahl und (85) die Produktion mit voller Kapazität.

Nun können wir eine Neuformulierung mit den hergeleiteten Ungleichungen aus Kapitel 3 durchführen:

$$\min \sum_{i,k,t} (p^i \rho^{i,k} x_t^{i,k}) + \sum_{i,t} (h^i s_t^i + e^i r_t^i)$$

unter den Nebenbedingungen

$$s_{t-1}^i + \sum_k \rho^{i,k} x_t^{i,k} = \sum_{j,k:j\neq i} \mathrm{RML}^{i,j} \rho^{j,k} x_t^{j,k} + s_t^i \tag{86}$$

$$s_{t-1}^i - r_{t-1}^i + \sum_k \rho^{i,k} x_t^{i,k} = d_t^i + s_t^i - r_t^i \quad \text{für alle } t \text{ und } i \in EP, \tag{87}$$

$$x_t^{i,k} + \mathrm{CLT}^i w_t^{i,k} \leq C^k y_t^{i,k} \quad \text{für alle } i,k,t \tag{88}$$

$$\sum_i x_t^{i,k} + \sum_i \mathrm{CLT}^i w_t^{i,k} \leq C^k \quad \text{für alle } k,t \tag{89}$$

$$w_{t-1}^{i,k} \geq y_{t-1}^{i,k} - y_t^{i,k} \quad \text{für alle } i,k,t \tag{90}$$

$$\sum_i y_t^{i,k} - \sum_i w_t^{i,k} \leq 1 \quad \text{für alle } k,t \tag{91}$$

$$\sum_{l=t}^{t+\alpha^{i,k}} w_l^{i,k} \leq y_t^{i,k} \quad \text{für alle } k,t,i \in EP \tag{92}$$

$$\sum_{l=t-\beta^{i,k}}^{t-\alpha^{i,k}} x_l^{i,k} + x_t^{i,k} \geq (MB^i/\rho^{i,k} - (\alpha^{i,k}-1)C^k) w_t^{i,k} \quad \text{für alle } k,t,i \in EP \tag{93}$$

$$x_t^{i,k} \geq C^k (y_{t-1}^{i,k} - w_t^{i,k} - w_{t-1}^{i,k}) \quad \text{für alle } i,k,t \tag{94}$$

$$r_{t-1}^i + s_l^i \geq (MB^i - d_{tl}^i)^+ \sum_k \sum_{\tau=t+b} w_\tau^{i,k} \quad \text{für alle } i,k,l,t \text{ mit } t+b<l, \tag{95}$$

$$s_{t-1}^i + \sum_{\tau=t}^{l} r_\tau^i \geq \sum_{\tau=t}^{i} d_\tau^i (1 - \sum_k \sum_{u=t}^{\tau-1} w_u^{i,k} - \sum_k y_\tau^{i,k}) \quad \text{für alle } i,k,l,t \tag{96}$$

$$s,r,t \geq 0, y,w \in \{0,1\}$$

$$s_0^i = S_0^i,\ r_0^i = 0,\ \underline{s} \leq s \leq \overline{s},$$

Dabei rührt Ungleichung (92) aus (63), (93) aus (71) und (72), (94) aus (70), (95) aus (73) und (96) aus Kombination von (15) und (17)

Um nun zu sehen wie die Neuformulierung A Priori die Lösung des Problems durch verschiedene Algorithmen beeinflusst betrachten wir die Ergebnisse einiger Instanzen von b4 wie sie in [3] aufgeführt sind

Instance	Type	NI	NK	NT	r	c	int
b4-10	SB2-ST	22	7	10	1576	1233	699
b4-10a	SB2-ST	22	7	10	1687	1233	699
b4-12	SB2-ST	22	7	12	1904	1485	849
b4-12a	SB2-ST	22	7	12	2236	1485	849

Abbildung 13: b4 Instanzen

In der Tabelle sind zu jeder Instanz die Anzahl der Produkte NI, die Anzahl der Maschinen NK sowie die Anzahl der Prioden NT angegeben. Darüberhinaus steht r für die Anzahl der Reihen, c für die Anzahl der Spalten und *int* für die Anzahl der ganzzahligen Variablen in der MIP Formulierung.
Instanz $b4-10a$ ist dabei die Neuformulierung zur Instanz $b4-10$ und $b4-12a$ analog die Neuformulierung zu $b4-12$. Betrachten wir nun das Ergebnis verschiedener Algorithmen mit den gegebenen Formulierungen.

instance	code	LP	XLP	IP	Secs	#cuts add	#cuts del	Gap
b4-10	bc-prod	12895.3	13509.4	14039	174	617	492	0
	bc-opt	12931.1	13198.7	14039	190	262	187	0
	mp-opt	12931.1		14039	108			0
b4-10a	bc-prod	13693.7	13852.6	14050.8	28	111	78	0
	bc-opt	13715.3	13846.5	14050.8	27	143	122	0
	mp-opt	13715.6		14050.8	30			0
b4-12	bc-prod	14177.4	14994.3	16093.4*	7200*	767	583	4.02
	bc-opt	14212.5	14493	16091.6*	7200*	345	249	5.05
	mp-opt	14212.5		16091.6*	7200*			5.61
b4-12a	bc-prod	15564.1	15721.5	16103.9	567	69	35	0
	bc-opt	15585.9	15660.9	16103.9	533	40	22	0
	mp-opt	15585.9		16103.9	844			0

Abbildung 14: b4 Ergebnisse

Die Spalte *code* notiert dabei das benutzte System, siehe [3], die Spalte LP enthält den Wert der ursprünglichen LP Formulierung ohne Schnitte, in der Spalte XLP stehen der Wert des LPs im Wurzelknoten nach Einfügen der Schnittebenen und die Spalte IP enthält die beste Lösung die nach 2h Rechenzeit gefunden wurde. Die Spalte *# cuts add* zeigt die Anzahl der hinzugefügten Schnitte an wohingegen *# cuts del* die Anzahl der gelöschten Schnitte im Wurzelknoten wiederspiegelt. Zuletzt notiert *gap* die Ganzzahligkeitslücke (*duality gap*) der besten gefundenen Lösung.

Man erkennt hierbei gut, dass die Instanzen, die A Priori neuformuliert wurden, deutlich schneller zur Optimallösung führen als die normalen Instanzen. So terminiert Instanz

$b4-10$ zwar noch innerhalb der 2h, braucht jedoch ca. 7 mal so lange wie die neuformulierte Instanz $b4-10a$. Allerdings sehen wir, dass schon bei Erhöhung der Periodenanzahl NT von 10 auf 12, wie es in Instanz $b4-12$ der Fall ist, keines der Systeme mit der Standartformulierung terminiert. Dies hängt vor allem mit der verbesserten Eingangschranke der Instanz $b4-12a$, also dem LP-Wert des Wurzelknoten notiert in Spalte XLP, zusammen.

5 Fazit

Losgrößenprobleme können, wie gezeigt, die verschiedensten Strukturen annehmen. In Kapitel 2 haben wir dazu eine grundlegende Modellierung eines solchen Problems vorgenommen. Für das Teilproblem eines Systems mit einem Produkt und einer Maschine haben wir damit gültige Ungleichungen hergeleitet. Darauf aufbauend sind wir in Kapitel 3 auf die verschiedensten Erweiterungen eingegangen. Unter anderem haben wir zusätzliche Anforderungen an den Bestand, die Produktion, die Nachfrage oder die Set-Ups betrachtet und uns für jeden Fall verschärfende Formulierungen abgeleitet. Allerdings steht außer Frage dass wir damit nur einen Teil der möglichen Strukturen, die in der Praxis auftreten können, abgedeckt haben. Wenngleich obere bzw. untere Schranken für Bestand und Produktion oder das Erlauben von backlogging häufig auftretende Bedingungen sind so bieten die Losgrößenprobleme doch ein weitaus breiteres Feld an Forschungsmöglichkeiten. Nur ein Beispiel könnte da die Losgrößenplanung auf parallel geschalteten Maschinen sein. Des Weiteren bietet aber auch die Integration der Losgrößenproblematik mit den uns bekannten Bedingungen in globalere Modelle ein breites Feld für neue Forschungsarbeit. So macht es vom betriebswirtschaftlichen Standpunkt auch durchaus Sinn Produktion und Distribution von Produkten als einen Prozess in der operativen Planung anzusehen und diesen simultan zu optimieren. Zudem könnte die Koordination der Losgrößenplanung mit Entscheidungen aus funktionalen Bereichen wie der Nachfrageplanung oder der Preissetzung ein Ansatzpunkt zukünftiger Forschung sein.

Auch für diese zukünftigen Forschungsbereiche bieten sich die Methoden der gemischtganzzahligen Programmierung an, da sie sich als Lösungsansatz für bisherig betrachtete Losgrößenprobleme als sehr nützlich erwiesen. So können auch große Probleme in akzeptabler Zeit gelöst werden. Dabei spielen Branch & Cut und Weiterentwicklungen die tragende Rolle. Moderne Systeme verwenden dabei Familien von Ungleichungen wie wir sie hergeleitet haben. Sie erkennen dabei bestimmte Nebenbedingungen der Problemstellung, wählen Schnitte aus und fügen sie zur Formulierung hinzu. Ihnen liegt dabei oft eine Klassifizierung von Losgrößenproblemen zugrunde, die gegebene Probleme z.B. anhand der 3 Felder PROB-CAB-VAR analysiert, siehe dazu [22]. Dank solcher Systeme können Blackbox Ansätze, basierend auf einer Sammlung von Neuformulierungen wie z.B. LS-LIB, verfolgt werden wie [20] detailiert beschreibt. Damit wird auch Nichtmathematikern ein Zugang zur Optimierung von gemischt-ganzzahligen Programmen verschafft, wie er noch vor 10 Jahren nicht möglich war.

Literaturverzeichnis

[1] Achterberg, Tobias ; Koch, Thorsten ; Martin, Alexander ; Revisited, Branching R. ; Achterbergt, Tobias ; Alex, Thorsten K. ; Martin, Er: Branching rules revisited. In: *Operations Research Letters* 33 (2004), S. 42–54

[2] Barany, Imre ; Roy, Tony J. V. ; Wolsey, Laurence A.: Strong Formulations for Multi-Item Capacitated Lot Sizing. In: *Management Science* 30 (1984), S. 1255–1261

[3] Belvaux, Gaetan ; Wolsey, Laurence A.: Lot-Sizing Problems: Modelling Issues and a Specialized Branch-and-Cut System BC-PROD. In: *CORE DP 9849, Louvain-la-Neuve*, 1998, S. 46–724

[4] Belvaux, Gaetan ; Wolsey, Laurence A.: Modelling Practical Lot-Sizing Problems as Mixed-Integer Programs. In: *MANAGEMENT SCIENCE* 47(7) (2001), S. 993–1007

[5] Clark, Andrew J. ; Scarf, Herbert: Optimal Policies for Multi-Echelon Inventory Problems. In: *Management Science* 6 (1960), S. 475–490

[6] Constantino, Miguel: A cutting plane approach to capacitated lot-sizing with start-up costs. In: *Mathematical Programming* 75 (1996), S. 353–376

[7] FICO: *Xpress Optimization suite*. `http://optimization.fico.com/student-version-of-fico-xpress.html`

[8] IBM: *ILOG CPLEX Optimizer*. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`

[9] Karmarkar, Uday S. ; Schrage, Linus: The Deterministic Dynamic Product Cycling Problem. In: *Operations Research* 33 (1985), S. 326–345

[10] Land, A. H. ; Doig, A. G.: An Automatic Method of Solving Discrete Programming Problems. In: *Econometrica* 28 (1960), Nr. 3, 497–520. `http://jmvidal.cse.sc.edu/library/land60a.pdf`

[11] Linderoth, J.T. ; Ralphs, T.K.: Noncommercial Software for Mixed-Integer Linear Programming. In: *Integer Programming: Theory and Practice*. CRC Press, 2005

[12] Marchand, Hugues ; Martin, Alexander ; Weismantel, Robert ; Wolsey, Laurence: *Cutting Planes in Integer Programming And Mixed Integer Programming*. 1999

[13] Marchand, Hugues ; Wolsey, Laurence A.: *Aggregation and Mixed Integer Rounding to Solve MIPs*. 1998

[14] Martin, Alexander: *Diskrete Optimierung.* `http://www.mathematik.tu-darmstadt.de/lehrmaterial/SS2006/OptimierungII/download/do.pdf`. Version: 2006. – Skript zur Vorlesung SS2006

[15] Martin, Alexander ; Dür, Mirjam ; Ulbrich, Stefan: *Optimierung I - Einführung in die Optimierung.* `https://www3.mathematik.tu-darmstadt.de/index.php?id=84&evsid=32&evsver=182&evsdir=656&evsfile=skript_Opt1.pdf`. Version: 2009. – Skript zur Vorlesung WS2008/2009

[16] Nemheuser, George L. ; Wolsey, Laurence A.: *Integer and Combinatorial Optimization.* Wiley-Interscience, 1999

[17] Pochet, Yves: Mathematical Programming Models and Formulations for Deterministic Production Planning Problems. In: *Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions [based on a Spring School].* London, UK : Springer-Verlag, 2001. – ISBN 3–540–42877–1, S. 57–111

[18] Pochet, Yves ; Wolsey, Laurence A.: Lot-Sizing with Constant Batches. Formulation and Valid Inequalities. In: *Mathematics of Operations Research* 18 (1993), S. 767–785

[19] Pochet, Yves ; Wolsey, Laurence A.: Adding Flexibility in Lot-Sizing Models Workshop on Production Planning and Control. (1996)

[20] Pochet, Yves ; Wolsey, Laurence A.: *Production Planning by Mixed Integer Programming.* Springer, 2006

[21] Schrijver, Alexander: *Theory of Linear and Integer Programming.* Wiley Interscience, 1999

[22] Wolsey, Laurence A.: Solving Multi-Item Lot-Sizing Problems with an MIP Solver Using Classification and Reformulation. In: *Management Science* 48 (2002), S. 1587–1602