Kapazitätsplanung bei Variantenfließfertigung unter Berücksichtigung der Flexibilität der Produktionslinien

An der Fakultät für Mathematik eingereichte

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Wirtschaftsmathematikerin
(Dipl.-Math. oec.)

Vorgelegt von Stefanie Weißbach,
geboren am 25. November 1984 in Karl-Marx-Stadt

Betreuer: Prof. Dr. Christoph Helmberg
Stefan Hornivius

Chemnitz, den 1. Juni 2008

Aufgabenstellung:

Ein Betrieb fertigt verschiedene Varianten eines Produktes in Fließbandfertigung. Für die Kapazitätsplanung soll auf Basis eines gegebenen Produktionsprogrammes die optimale Zuordnung der Produktvarianten zu den vorhandenen Maschinen ermittelt werden. Dabei sind Produktionsflüsse und verschiedene Einflussparameter zu berücksichtigen. Das Problem soll zunächst modelliert, das Modell mathematisch diskutiert und anschließend auf geeignete Weise gelöst werden. Gegebenenfalls sind für die praktische Umsetzbarkeit Heuristiken zu entwickeln und zu bewerten.

Vorwort

In der vorliegenden Arbeit wird ein Modell für die Kapazitäts- und Investitionsplanung in einem mehrstufigen Produktionsprozess mit mehreren parallel herzustellenden Produkttypen aufgestellt und gelöst. Besonderes Augenmerk wird dabei auf typ- und maschinenspezifische Parameter sowie die beschränkte Zuordnung von Produkten zu Maschinen gelegt. Zunächst werden in Kapitel 1 die spezielle Ausgangssituation, Restriktionen und Ziele verbal vorgestellt. Anschließend wird unter Berücksichtigung der relevanten Parameter und Zielstellungen die Kapazitätsplanung als lineares gemischt-ganzzahliges Problem modelliert. Kapitel 3 unternimmt eine kritische Betrachtung sowohl der Vorteile als auch Einschränkungen der gewählten Modellierung. Nach einem allgemeinen Überblick über exakte Lösungsverfahren erfolgt deren Anwendung auf das spezielle Problem in Kapitel 5. Es werden numerische Ergebnisse mit verschiedenen Instanzen angegeben und interpretiert. Als weiteren Ausblick behandelt der letzte Abschnitt die Anwendung auf mehrere Bedarfsperioden und Szenarien.
Es sei an dieser Stelle auf einige verwendete Schreibweisen verwiesen. Vektoren $x \in \mathbb{R}^n$ seien stets Spaltenvektoren. Zahlen wurden zur besseren Lesbarkeit mit Tausendertrennzeichen dargestellt, z. B. 10.000. Die Wahl der Wir-Form soll den Leser zum aktiven Nachvollziehen der Inhalte anregen und bezieht sich nicht auf mehrere Autoren.

Ich möchte mich bei Prof. Dr. Helmberg für die herzliche und kompetente Betreuung der Diplomarbeit sowie Herrn Stefan Hornivius und den Mitarbeitern von Continental für die konstruktive und engagierte Zusammenarbeit bedanken. Meinen Eltern gilt besonderer Dank für viele Jahre liebevoller und nachsichtiger Unterstützung.

Chemnitz, 1. Juni 2008 Stefanie Weißbach

Inhaltsverzeichnis

1 Problembeschreibung **1**
- 1.1 Das Unternehmen . . . 1
- 1.2 Produktion und Parameter . . . 1
- 1.3 Kapazitätsplanung . . . 4
  - 1.3.1 Probleme der bisherigen Planung . . . 5
  - 1.3.2 Ziele und Anforderungen . . . 5

2 Modellierung **7**
- 2.1 Mengen und Notation . . . 7
- 2.2 Parameter . . . 8
- 2.3 Variablen . . . 9
- 2.4 Nebenbedingungen . . . 10
- 2.5 Zielfunktion . . . 12
- 2.6 Zusammenfassung des Modells . . . 12
- 2.7 Größenordnung . . . 12
- 2.8 Komplexität . . . 13

3 Diskussion des Modells **16**
- 3.1 Kapazitäts- vs. Fertigungsplanung . . . 16
- 3.2 Vorteile gegenüber der bisherigen Planung . . . 17
- 3.3 Genauigkeit der Rüstzeiten . . . 17
- 3.4 Stillstandzeiten . . . 20
- 3.5 Unsicherheit von Daten . . . 21

4 Exakte Lösungsverfahren **24**
- 4.1 Das Prinzip exakter Lösungsverfahren . . . 24
- 4.2 Gute Formulierungen . . . 25
- 4.3 Branch-and-Bound . . . 29
- 4.4 Schnittebenenverfahren und Separierungsalgorithmen . . . 31
  - 4.4.1 Prinzip . . . 31
  - 4.4.2 Spezielle Schnittebenen . . . 31
  - 4.4.3 Branch-and-Cut . . . 33

5 Lösung des Modells und numerische Ergebnisse **34**
5.1 Der Solver lp_solve 34
5.2 Testinstanzen 34
5.3 Numerische Ergebnisse 35

6 Erweiterungsmöglichkeiten **39**
6.1 Semidefinite Relaxierung 39
6.2 Robuste und stochastische Optimierung 40

7 Fazit **43**

Abbildungsverzeichnis **44**

Tabellenverzeichnis **45**

Literaturverzeichnis **46**

Eidesstattliche Erklärung **48**

Kapitel 1

Problembeschreibung

1.1 Das Unternehmen

Continental fertigt am Standort Limbach-Oberfrohna Piezo-Common-Rail-Injektoren und weitere Komponenten für Dieseleinspritzsysteme für Common-Rail-, Amplified Common-Rail- und Pumpe-Düse-Einspritzsysteme. Beliefert werden i.d.R. direkt die Motorenwerke verschiedener Automobilhersteller.
Das Werk in Limbach-Oberfrohna geht auf das 1950 gegründete Bremsenwerk Limbach zurück, das zunächst Hydraulikkomponenten und -systeme fertigte. Der Standort wurde 1997 ein Unternehmen der Siemens AG und ging 2001 zur Siemens VDO Automotive AG über. Seit dem 5. Dezember 2007 ist diese unter dem neuen Namen VDO Automotive AG vollständig im Besitz der Continental AG. Continental rückte nach der Übernahme von Siemens VDO hinsichtlich Umsatzstärke unter die Top-Fünf der Automobilindustrie und erwartet für 2008 einen Umsatz von über 26 Mrd.Euro. Bis 2012 werden sich die Auftragszahlen für das Werk in Limbach-Oberfrohna verdoppeln. Diese Stückzahlentwicklung sowie die zunehmende Produktvielfalt in der Automobilindustrie erfordern eine exakte Planung der vorhandenen Ressourcen.

1.2 Produktion und Parameter

Die Common-Rail-Injektoren werden auf hochautomatisierten Linien in Serienfertigung produziert. Im Werk werden die Bauteile Düse, Drosselmodul, Aktor und Injektorkörper bearbeitet, anschließend erfolgen die Montage der Injektoren sowie mehrere Qualitätsprüfungen. Neben den kompletten Injektoren gibt es Düsen und Aktoren, die nicht verbaut, sondern direkt an andere injektormontierende Standorte ausgeliefert werden.

Der grobe Produktionsablauf ist in Abbildung 1.1 dargestellt. Am Beispiel Drosselmodul wird im Bild außerdem veranschaulicht, dass die Prozesse in aufeinanderfolgende Arbeitsgänge mit jeweils mehreren Maschinen unterteilt sind.

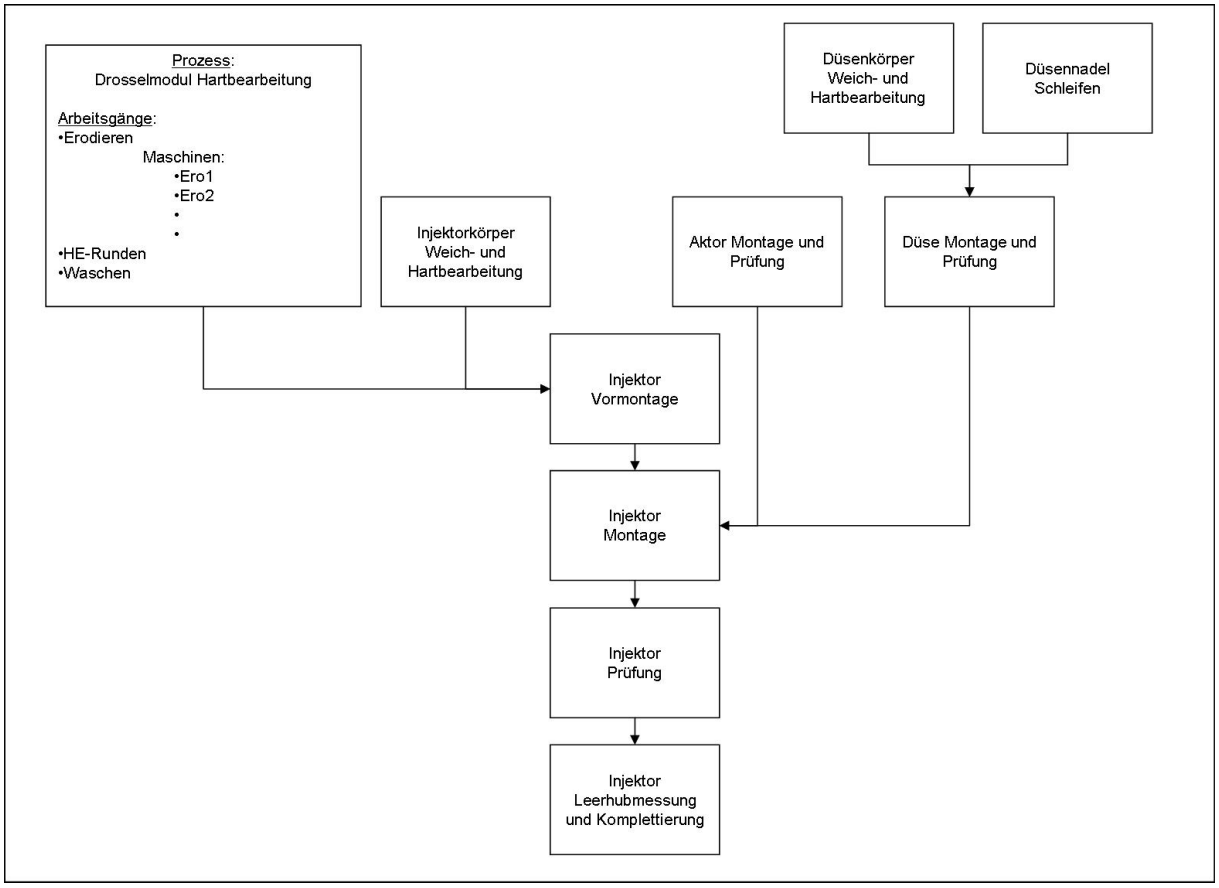

Abbildung 1.1: Prozessfluss Injektorfertigung

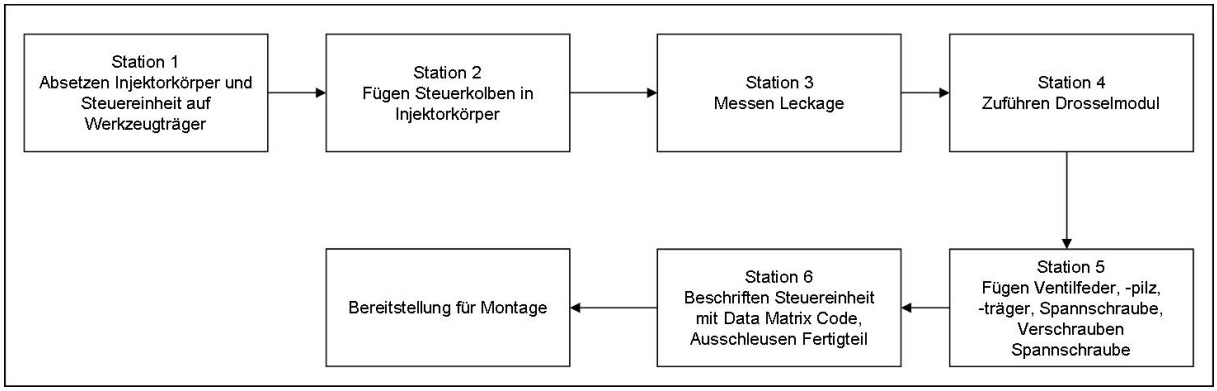

Abbildung 1.2: Arbeitsstationen einer Vormontagelinie

Eine Maschine eines Arbeitsganges umfasst elementare Produktionsschritte als Arbeitsstationen, welche bei Montagen durch Förderbänder und Handlingsysteme verbunden sind und in einem festen zeitlichen Rhythmus ausgeführt werden. Abbildung 1.2 zeigt beispielhaft den Aufbau einer Linie des Arbeitsganges Vormontage. Der detaillierte Aufbau der Fertigungslinien ist nur zum Verständnis der Fließfertigung aufgeführt und wird für das Problem keine direkte Rolle spielen. Wir betrachteten später nur die organisatorischen Einheiten Arbeitsgang und Maschine.

Für die Kapazitätsplanung werden einige grundlegende Objekte und Parameter benötigt:

Arbeitsplan
Im Wesentlichen durchlaufen alle Typen die gleichen Arbeitsgänge, z.T. sind aber auch typspezifische Sonderarbeitsgänge notwendig. Deshalb existiert für jedes Kundenprojekt ein Arbeitsplan, in dem die Produktionsstufen hinterlegt sind.

Für jeden Arbeitsgang stehen mehrere Maschinen zur Verfügung, die sich hinsichtlich ihrer Technologien unterscheiden können. Verschiedene Technologien resultieren z.B. aus dem Alter von Maschinen oder Kosteneinsparungen bei der Anschaffung. Maschinen werden durch folgende Parameter charakterisiert:

Kapazität
Als Kapazität benutzen wir die verfügbare Zeit in einem festen Zeitraum abzüglich einer maschinenspezifischen erwarteten Stillstandzeit. Bisher wird die Kapazität üblicherweise als Anzahl einwandfreier Teile, die in einem bestimmten Zeitraum produziert werden können, angegeben. Da für deren Berechnung typspezifische Bearbeitungszeiten gemittelt werden, ist dieser Wert für unsere Zwecke zu ungenau.

Typfreigabe
Aufgrund der verschiedenen Technologien können auf manchen Maschinen nicht alle Typen bearbeitet werden. Zusätzlich sind für die Produktion eines Types auf einer Maschine Kundenfreigaben notwendig, die vom Kunden anhand von Qualitätsmerkmalen erteilt werden. Außerdem kann eine Maschine für einen Kunden reserviert werden, so dass keine anderen Produkte darauf gefertigt werden dürfen, selbst wenn dies aus technologischer Sicht möglich wäre.

Taktzeit
Die Taktzeit ist der Zeitraum zwischen der Fertigstellung zweier Teile auf einer Maschine und wird bestimmt durch die Arbeitsstation mit der längsten Bearbeitungsdauer. Sie ist nicht zu verwechseln mit der Bearbeitungszeit, die ein Teil von der ersten bis zur letzten Arbeitsstation benötigt. Die Taktzeit kann sowohl maschinen- als auch typspezifisch sein.

Yield
Während der Bearbeitung werden Teile aufgrund von Qualitätsmängeln aussortiert und können nicht weiterverarbeitet werden. Der Yield gibt den prozentualen Anteil einwandfreier Teile nach der Bearbeitung auf einer Maschine an. Hierbei ist zu unterscheiden zwischen Yield und First Pass Yield (FPY). Der FPY berechnet sich aus Teilen, die erstmals über die Maschine laufen. Allerdings werden fehlerhafte Teile in manchen Arbeitsgängen nicht sofort verschrottet, sondern in einem Befundungsarbeitsgang untersucht und erneut in die Produktion eingeschleust. Man spricht hierbei von Zweitläufern. Der Gesamt-Yield einer Anlage bezeichnet also den Anteil von einwandfreien Teilen einer Inputmenge nach Befundung und Zweitlauf und ist demzufolge mindestens so hoch wie der FPY. Der Yield kann wie die Taktzeit maschinen- und typabhängig sein.

Zweitläuferquote
Diese gibt den prozentualen Anteil von Zweitläufern einer bestimmten Eingangsmenge an und unterscheidet sich ebenfalls je nach Typ und Maschine.

Rüstzeiten
Rüstzeit fällt an, wenn eine Maschine bei einem Typwechsel für die Abarbeitung vorbereitet werden muss. Im einfachsten Fall resultiert sie aus dem Leerfahren der Anlage, wenn alle Teile eines Types die Maschine verlassen haben müssen, bevor der nächste Typ eingeschleust werden kann. Aufwändige Rüstvorgänge treten bei Werkzeugwechsel, Neueinstellung von Messgeräten u.Ä. auf. Dabei können die Rüstzeiten stark vom Typwechsel abhängig sein. In unserem Beispiel sind die Rüstzeiten symmetrisch, der Wechsel von Typ A nach B kostet die gleiche Zeit wie der Wechsel von B nach A.

1.3 Kapazitätsplanung

Für jeden Injektortypen sind auf Basis von Kundenverträgen und online eingehenden Abrufen jährliche Bedarfe bekannt. Diese werden in Abhängigkeit von der Anzahl der Arbeitstage in Monatsscheiben unterteilt und bilden die Grundlage für die Investitionsplanung.
Neben der langfristigen Investitionsplanung ist auch eine mittelfristige Kapazitätsplanung durchzuführen. Grundlage hierfür ist ein Produktionsplan, in dem die wöchentlich zu produzierenden Stückzahlen festgelegt sind. Der Produktionsplan wird von der Abteilung Logistik unter Berücksichtigung von Lieferterminen, Bestandsentwicklungen, Umbaumaßnahmen, Bedarfsspitzen etc. erstellt.
Die Abteilung Technology and Manufacturing Planning (kurz: TE) hat zur Aufgabe, die Erfüllung der langfristigen Kundenbedarfe sowie die Umsetzung des Produktionsplanes zu gewährleisten und bei Bedarf rechtzeitig Investitionen in neue Maschinen oder Maschinenumbaumaßnahmen zu tätigen. Ziel der Planungen ist es, die vorhandenen Produktionskapazitäten unter Beachtung der Typabhängigkeiten optimal auszunutzen und Kapazitätsengpässe aufzuzeigen.

1.3.1 Probleme der bisherigen Planung

In der bisherigen Vorgehensweise können nur ungenaue Aussagen zu vorhandenen Produktionskapazitäten gemacht werden. Die Reaktion auf geänderte Kundenbedarfe ist häufig zu langsam oder fehlerhaft. Darüber hinaus erfordert die Planung einen großen Personaleinsatz, für ein Bedarfsszenario sind im Durchschnitt fünf Technologen für ein bis zwei Tage gebunden.

Dies liegt vor allem daran, dass kein standardisiertes und automatisiertes Verfahren für die Kapazitäts- und Investitionsplanung existiert. Die Arbeitsbereiche Düse, Aktor, Drosselmodul, Injektorkörper und Montage benutzen verschiedene MS Excel-Tabellen als Planungstools, wobei die Typen per Hand den Maschinen zugeordnet werden. Durch die Komplexität der Produktionsprozesse kann auf diesem Weg jedoch oft keine optimale Ausnutzung der Flexibilität der Linien und der vorhandenen Kapazität erreicht werden. Zusätzlich sind wichtige Eingangsgrößen wie Yield oder Verfügbarkeiten z.T. nicht eindeutig definiert, ungenau oder nicht aktuell.
Die Rüstzeiten sind bisher fest in die prozentuale Verfügbarkeit einer Maschine eingerechnet und somit unabhängig von der Maschinenbelegung.
Als Maschinenkapazität wird die Stückzahlkapazität benutzt, die wie bereits erwähnt aufgrund von typspezifischen Parametern unpräzise sein kann.

Diese fehlerhafte Kapazitätsbetrachtung und die fehlende Kommunikation zwischen Arbeitsbereichen führt zu falschen Investitionsplänen, sei es in Form von zu hohen Investausgaben oder fehlender Produktionskapazität mit der Folge, dass Kundenaufträge nicht erfüllt werden können.

1.3.2 Ziele und Anforderungen

Das Ziel dieser Diplomarbeit ist die Entwicklung eines zentralen Kapazitätsplanungstools, das den Aufwand für die Bearbeitung verschiedener Stückzahlszenarien reduziert und die Qualität der Investitionsplanung erhöht. Die einzelnen Technologen sollen in Zukunft primär für die Anlagenbeschaffung und Überwachung bzw. Aktualität der Parameter verantwortlich sein.
Aus den Ergebnissen soll erkennbar sein, zu welchem Zeitpunkt in welchen Arbeitsgängen die Kapazitäten angepasst werden müssen. Eine schnelle Bewertung verschiedener Szenarien soll das Vergleichen verschiedener Strategien wie den Neukauf oder den Umbau einer Maschine ermöglichen.
Die Flexibilität der Linien ist auszunutzen, um eine bestmögliche Maschinenbelegung zu finden. Dafür sollen genauere, eindeutig definierte Parameter, insbesondere typwechselabhängige Rüstzeiten, einbezogen werden.
Die Verfügbarkeit von Zulieferteilen und Arbeitskräften soll als gewährleistet vorausge-

setzt werden, da die Sicherstellung dieser Faktoren nicht in den Zuständigkeitsbereich der Abteilung TE fällt.
Ebensowenig ist eine Reihenfolgeplanung gewünscht, wofür es zwei wesentliche Gründe gibt:
Zum einen sind vor allem die weit in der Zukunft liegenden Kundenbedarfe mit Unsicherheiten behaftet und können sich noch mehrmals ändern. Eine aufwändige Feinplanung müsste häufig wieder verworfen werden. Desweiteren erfolgt die Fertigungsplanung in einer anderen Abteilung, der weitere Einflussgrößen wie Liefertermine bekannt sind. Es sind also im betrachteten Problem für die Arbeitsgänge keine Fertigungstermine, Übergangszeiten und andere Gesichtspunkte der Feinterminierung zu berücksichtigen.
Im Hinblick auf den Produktionsfluss muss beachtet werden, dass pro Arbeitsgang nur die Menge bearbeitet werden kann, die vom Vorgängerprozess abzüglich mangelhafter Teile bereitgestellt wird. Eine Überproduktion und der damit verbundene Aufbau von Vorräten ist unerwünscht.
Die technische Umsetzung eines Algorithmus zur automatisierten Planung soll möglichst mit bereits existierenden Dateien arbeiten. Ziel ist ein Tool, das nutzerfreundlich, robust gegenüber Anwenderfehlern und leicht zu aktualisieren ist.

Kapitel 2

Modellierung

Anhand der Problemstellung soll nun das mathematische Modell für die Kapazitätsplanung aufgestellt werden. Es betrachtet vorerst einen festen Bedarf für einen Zeitraum T. Eine einperiodige Betrachtung ist zulässig, da keine Lagerhaltung oder Vorproduktion einbezogen werden soll.
Zunächst definieren wir die benötigten Objekte, Daten und Variablen.

2.1 Mengen und Notation

P $:=\{1,\ldots,n_p\}$: Menge aller Produkte, $n_p \in \mathbb{N}$: Anzahl der Produkttypen
$\mathcal{A}$: Menge aller im Werk durchführbarer Arbeitsgänge
$\mathcal{A}_i$ $:=(A_1^{(i)},\ldots,A_{n_i}^{(i)})$: Arbeitsplan für Produkt $i \in P$, $A_k^{(i)} \in \mathcal{A}$, $k = 1,\ldots,n_i$, $n_i \in \mathbb{N}$

Da die Arbeitspläne typspezifisch sind, ist es bei $|\mathcal{A}_i| > 1$ wichtig zu wissen, in welcher Reihenfolge die Arbeitsgänge abgearbeitet werden müssen. Dabei ist die Baumstruktur des Produktionsprozesses zu beachten (vgl. Abb. 1.1). Für den letzten Bearbeitungsschritt definieren wir als Nachfolger einen Arbeitsgang „Versandlager“.

$N_k^{(i)}$: Nachfolgearbeitsgang von $A_k^{(i)}$, $N_k^{(i)} \in \mathcal{A} \cup \{\text{„Versandlager“}\}$, $i \in P$, $k = 1,\ldots,n_i$
$\mathcal{N}_i$ $:= (N_1^{(i)},\ldots,N_{n_i}^{(i)})$

$\mathcal{N}_i$ ist eindeutig, da jeder Arbeitsgang genau einen Nachfolger hat. Es können aber Arbeitsgänge mehrere Vorgänger haben. Desweiteren sei

M_A: Menge der Maschinen für Arbeitsgang $A \in \mathcal{A}$.

Da für jeden Arbeitsgang angegeben werden soll, welche Menge der geforderten Stückzahlen mit der vorhandenen Kapazität nicht erfüllt werden kann, enthalte M_A neben den bereits im Werk vorhandenen Maschinen noch je eine imaginäre Maschine z_A, die den Stückzahlüberschuss bearbeitet. Die Kapazität dieser Zusatzmaschinen sei unendlich, da nicht bekannt ist, wieviele Maschinen benötigt werden. Potentielle Neumaschinen werden

mit diesen Variablen nicht modelliert, so dass z_A eher als ein Auffanglager für nicht erfüllbaren Bedarf zu interpretieren ist, dessen Umfang minimiert werden soll. Aus dem Wert soll später abgeleitet werden, in welchem Umfang Kapazitätsanpassungsmaßnahmen wie Schichterhöhung oder Maschinenkauf ergriffen werden müssen.
Es seien weiterhin

$Z := \{z_A : A \in \mathcal{A}\}$: Menge der künstlichen Maschinen
$M := \bigcup_{A \in \mathcal{A}} M_A$: Menge aller realen und künstlichen Maschinen
$P_m \subset P$: Menge der freigegebenen Produkte auf Maschine m, $m \in M$.

Es gelte

$$P_m \equiv P, \quad \forall m \in Z$$

und

$$M_A \cap M_B = \varnothing \quad \forall A, B \in \mathcal{A}, A \neq B.$$

2.2 Parameter

Im Folgenden führen wir die relevanten Parameter für die Kapazitätsbetrachtung ein. Dabei werden wie bereits erwähnt keine Maschinenkenngrößen für die imaginären Maschinen definiert, da diese nur wie Stückzahllager behandelt werden. Es bezeichne

c^m: Kapazität der Maschine m in Zeiteinheiten für den Produktionszeitraum T, $m \in M \backslash Z$, $c^m \geq 0$,
d_i : Soll-Stückzahl von Produkt i, $i \in P$, $d_i \geq 0$.

Aus den bereits dargelegten Gründen wird die Kapazität für den betrachteten Zeitraum berechnet, indem von der verfügbaren Gesamtzeit (abhängig von Schichtsystem, Arbeitstagen, geplanten Wartungsschichten) ein Anteil für maschinenabhängige organisatorische und technische Verfügbarkeit abgezogen wird. Es ist zu beachten, dass wir für die Kapazitätsplanung innerhalb eines Arbeitsganges nur die verfügbare Zeitspanne angeben, nicht aber den Termin für den Beginn der Produktion. In der Fertigungsplanung und -terminierung werden später diese Zeitfenster so verschoben, dass der Bedarf zum Liefertermin erfüllt werden kann. Dieses Thema diskutieren wir in Kapitel 3.
Weiterhin benötigen wir

t_i^m : Taktzeit für die Bearbeitung von Produkt i auf Maschine m in Zeiteinheiten, $m \in M \backslash Z$, $i \in P_m$, $t_i^m \geq 0$,
a_i^m: relativer Anteil einwandfreier Produkte (Yield) vom Typ i an der Gesamtproduktionsmenge auf Maschine m, $m \in M \backslash Z$, $i \in P_m$, $0 \leq a_i^m \leq 1$,
z_i^m: Zweitläuferquote von Produkt i auf Maschine m, $m \in M \backslash Z$, $i \in P_m$, $0 \leq z_i^m \leq 1$,
r_{ij}^m: Rüstzeit bei der Produktion der Produkte i und j auf Maschine m für den Betrachtungszeitraum in Zeiteinheiten, $m \in M \backslash Z$, i,$j \in P_m$, $i < j$, $r_{ij}^m \geq 0$.

Wir werden später sehen, dass es sinnvoll ist, Yields auch für die imaginären Maschinen zu definieren.
Die Rüstzeitmatrizen $R_m := [r_{ij}^m]_{i,j \in P_m}$ seien nichtnegative obere Dreiecksmatrizen mit Nulldiagonale.
Wegen mangelnder Reihenfolgebetrachtung wird die Rüstzeit mithilfe von mittleren Häufigkeiten berechnet. Betrachtet man z.B. für die Investitionsplanung den Zeitraum von einem Monat, kann man aus Daten der Vergangenheit eine mittlere Häufigkeit des Umrüstens von Produkt i auf Produkt j ermitteln und so eine durchschnittliche monatliche Rüstzeit angeben. Die berechnete Rüstzeit ist also nur ein erwarteter Wert, der von der Realität abweichen kann. In Kapitel 3.3 geben wir eine Schranke für diese Abweichung an.

2.3 Variablen

Entscheidungsvariablen sind die auf jeder Maschine zu bearbeitenden Mengen eines jeden Injektortypes. Dabei erfolgt keine terminliche Festlegung der Produktion eines bestimmten Types auf einer Maschine:

x_i^m: auf Maschine m zu bearbeitende Menge von Produkt i, $i \in P_m$, $m \in M$, $x_i^m \geq 0$.

Es ist zu beachten, dass x_i^m die Eingangsmenge auf Maschine m bezeichnet, der Output der Maschine ist $a_i^m x_i^m$.

Um Rüstzeiten einzubeziehen, sobald zwei verschiedene Typen auf einer Linie gefertigt werden, benötigen wir 0-1-Variablen u_{ij}^m, $m \in M \backslash Z$, $i,j \in P_m$, $i < j$, mit der Eigenschaft:

$$u_{ij}^m := \begin{cases} 1 & \text{i und j werden auf m bearbeitet,} \\ 0 & \text{sonst.} \end{cases}$$

Außerdem sei für $m \in M \backslash Z$, $i \in P_m$,

$$u_i^m := \begin{cases} 1 & \text{i wird auf m bearbeitet,} \\ 0 & \text{sonst.} \end{cases}$$

Damit gilt $u_{ij}^m = u_i^m u_j^m$.
Im Folgenden sei angenommen, dass für die Menge M eine Ordnung definiert ist, beispielsweise mittels eindeutiger Identifikationsnummern oder alphabetisch geordneter Bezeichnungen.
Dann bezeichnen wir mit

$x := [x_i^m]_{m \in M, i \in P_m}$	den Vektor der stetigen Variablen und mit
$u := [u_i^m, u_{ij}^m]^T_{m \in M, i,j \in P_m, i<j}$	den Vektor aller Binärvariablen.

2.4 Nebenbedingungen

Aus der Problemformulierung ergeben sich folgende Nebenbedingungen:

Kapazitätsnebenbedingung
Für jede Maschine muss die verfügbare Kapazität eingehalten werden. Diese wird beansprucht durch die Bearbeitung aller Inputmengen zuzüglich der Zweitläufer sowie durch Rüstzeiten bei Wechseln zwischen verschiedenen Typen.

$$\sum_{i \in P_m} t_i^m (1 + z_i^m) x_i^m + \sum_{i \in P_m} \sum_{\substack{j \in P_m \\ j > i}} r_{ij}^m u_{ij}^m \leq c^m \qquad \forall m \in M \backslash Z \tag{K}$$

Hier wird für jeden möglichen Typwechsel eine Rüstzeit berechnet, da die Anzahl und Art der tatsächlich stattfindenden Wechsel reihenfolgeabhängig und somit nicht bekannt ist. Diese Problematik wird im nächsten Kapitel diskutiert.

Indikatorvariablen-Bedingung
Hierbei handelt es sich um den Zusammenhang $x_i^m > 0 \Rightarrow u_i^m = 1$:

$$x_i^m - u_i^m S_i^m \leq 0 \qquad \forall m \in M \backslash Z, i \in P_m \tag{I}$$

Dabei sei S_i^m eine obere Schranke von x_i^m. Diese erhält man z.B. mittels

$$S_i^m = \frac{c^m}{t_i^m (1 + z_i^m)} \quad .$$

Die Bedingung (I) entspricht der sogenannten Fix-Cost-Ungleichung, die in vielen Planungsaufgaben auftaucht, siehe z.B. [11, S.9].

Bemerkung:
Es ist bei dieser Formulierung zulässig, dass gleichzeitig $x_i^m = 0$ und $u_i^m = 1$. Für einen minimalen Kapazitätsbedarf wird dieser Fall bei einer sinnvollen Zielfunktion jedoch nicht in einer Optimallösung auftreten oder diese nicht beeinflussen.

Rüstvariablen-Bedingung
Rüstzeiten zwischen zwei Typen i und j sollen genau dann berücksichtigt werden, wenn beide Typen der Maschine zugeordnet werden, d.h. $u_{ij}^m = 1 \Leftrightarrow u_i^m = 1 \wedge u_j^m = 1$. Diese Bedingung lässt sich durch folgende Ungleichungen beschreiben:

$$u_{ij}^m \leq u_i^m \tag{R1}$$
$$u_{ij}^m \leq u_j^m \tag{R2}$$
$$u_i^m + u_j^m \leq 1 + u_{ij}^m \tag{R3}$$
$$u_{ij}^m, u_i^m, u_j^m \in \{0, 1\}, \quad \forall m \in M \backslash Z, i, j \in P_m, i < j$$

Die Transformation eines ursprünglich quadratischen Terms (wie hier der Rüstzeitterm $u_{ij}^m = u_i^m u_j^m$ in der Kapazitätsbedingung) in einen linearen Ausdruck durch das Einführen zusätzlicher Variablen ist eine gebräuchliche Methode in der binären Optimierung und existiert auch für höherdimensionale Polynome von Variablen u_1, .. , u_n. Ausführlicheres hierzu findet man z.B. in [11, S.132]. Da eine Erhöhung der Variablen u_{ij}^m zu einem Kapazitätsbedarf in (K) führt, wird der Fall $u_i^m = 0$ (keine Produktion von Typ i auf m) und gleichzeitig $u_{ij}^m = 1$ (Rüstzeit für Typ i auf m) für gute Lösungen mit geringem Ressourcenbedarf ausgeschlossen. Die Bedingungen (R1) und (R2) sind deshalb für unsere Aufgabe nicht notwendig.

Flusserhaltungs-Bedingung
Die Outputmenge eines Prozesses soll vollständig als Inputmenge in den nächsten Arbeitsgang eingehen. Dabei ist zu beachten, dass von den bearbeiteten Mengen ein gewisser Anteil an Ausschuss wegfällt. Ergebnis nach dem letzten Prozess sollen genau die gewünschten Stückzahlen sein.

$$\sum_{m \in M_{A_k^{(i)}}} a_i^m x_i^m = \sum_{m \in M_{N_k^{(i)}}} x_i^m \qquad \forall i \in P, k = 1, \ldots, n_i - 1$$

$$\sum_{m \in M_{A_{n_i}^{(i)}}} a_i^m x_i^m = d_i \qquad \forall i \in P \qquad \text{(F)}$$

Bemerkungen:
Nicht erfüllbarer Bedarf eines Arbeitsganges wird bei dieser Modellierung über die Variable x_i^m mit $m \in Z$ registriert und mit in die Gesamtbedarfsmenge für den nächsten Produktionsschritt aufgenommen. Denn Ziel ist es, für jeden Arbeitsgang fehlende Kapazität zu identifizieren, unabhängig von Engpässen in anderen Prozessen.

Für die Überschussmengen $x_i^{z_A}$ sollte ebenfalls ein Yield angegeben werden. Sonst kann es für die Minimierung des Gesamtüberbedarfes besser sein, in Arbeitsgängen am Ende des Produktionsprozesses reale Maschinen nicht auszulasten und stattdessen $x_i^{z_A}$ zu erhöhen. Denn die Mehrproduktion für Vorgängerarbeitsgänge schaukelt sich wegen der Yields in den Flusserhaltungsbedingungen auf und führt dort zu einem erhöhten Kapazitätsbedarf, welcher wiederum einen Anstieg der dortigen Zusatzmengen bewirkt. Dieser unerwünschte Effekt der Nichtnutzung vorhandener Ressourcen wird vermieden, wenn für die künstliche Maschine z_A ein kleinerer als der minimale Yield aller realen Maschinen des Arbeitsganges A definiert wird. Befinden sich in dem Arbeitsgangs allerdings alte Maschinen mit hohen Ausschusswerten, sollte abgewogen werden, ob für $a_i^{z_A}$ ein besserer Wert entsprechend einer potentiellen Neumaschine gewählt wird mit der Konsequenz, dass in der Optimallösung alte Maschinen nicht zwangsläufig ausgelastet werden.

2.5 Zielfunktion

Um eine maximale und ressourcensparende Nutzung der vorhandenen Kapazitäten zu erreichen sowie den Umfang der fehlenden Kapazität einzuschätzen, soll die Summe des nicht erfüllbaren Bedarfes aller Arbeitsgänge minimiert werden. Die Zielfunktion sei also

$$f(x,u) := \sum_{\substack{i \in P \\ m \in Z}} x_i^m \quad .$$

Der Zielfunktionswert $f(x,u)$ ist vorsichtig zu interpretieren, da die Höhe der $x_i^{z_A}$ von der Wahl der $a_i^{z_A}$ abhängt. Es lässt sich aber bei einer sinnvollen Wahl gut für die einzelnen Arbeitsgänge abschätzen, ob die fehlende Kapazität über eine Zusatzschicht abgedeckt werden kann oder der Kauf einer neuen Maschine nötig ist.

2.6 Zusammenfassung des Modells

$$\begin{aligned}
&\text{min} && f(x,u) = \sum_{\substack{i \in P \\ m \in Z}} x_i^m && && \\
&\text{subject to} && && && \\
&\sum_{i \in P_m} t_i^m (1 + z_i^m) x_i^m && + \sum_{i \in P_m} \sum_{\substack{j \in P_m \\ j > i}} r_{ij}^m u_{ij}^m \leq c^m && \forall m \in M \backslash Z && \text{(K)} \\
& && x_i^m - S_i^m u_i^m \leq 0 && \forall m \in M \backslash Z, i \in P_m && \text{(I)} \\
& && u_i^m + u_j^m \leq 1 + u_{ij}^m && \forall m \in M \backslash Z, i,j \in P_m, i < j && \text{(R3)} \\
& && \sum_{m \in M_{A_k^{(i)}}} a_i^m x_i^m = \sum_{m \in M_{N_k^{(i)}}} x_i^m && \forall i \in P, k = 1, \ldots, n_i - 1 && \\
& && \sum_{m \in M_{A_{n_i}^{(i)}}} a_i^m x_i^m = d_i && \forall i \in P && \text{(F)} \\
& && x_i^m \geq 0, u_{ij}^m, u_i^m, u_j^m \in \{0,1\} && &&
\end{aligned}$$

Bei diesem Optimierungsproblem handelt es sich um ein gemischt-ganzzahliges, lineares Optimierungsproblem.

2.7 Größenordnung

Das Modell nimmt bei einer gegebenen Anzahl von Typen und Arbeitsgängen einen maximalen Umfang an, wenn alle Typbelegungen erlaubt sind, d.h. $P_m \equiv P$ für alle $m \in M$, und jeder Typ alle Arbeitsgänge durchläuft. Die theoretische Anzahl an Variablen und

Nebenbedingungen sowie die aktuellen Werte aus der Praxis sind in Tabelle 2.1 aufgeführt.

Mengen/Var./NB	max.theoret.Anz.	max.reale Anz.	reale Anz. wg. P_m und r_{ij}^m
Injektortypen	$\|P\|$	30	30
Arbeitsgänge	$\|\mathcal{A}\|$	42	42
reale Maschinen	$\|M \backslash Z\|$	242	242
x_i^m	$\|M\| \cdot \|P\|$	8.520	6.313
u_i^m	$\|M \backslash Z\| \cdot \|P\|$	7.260	5.035
u_{ij}^m	$\frac{1}{2} \cdot \|M \backslash Z\| \cdot \|P\| \cdot (\|P\| - 1)$	105.270	595
Summe Var.		121.050	11.943
(K)	$\|M \backslash Z\|$	242	242
(I)	$\|M \backslash Z\| \cdot \|P\|$	7.260	5.035
(R3)	$\frac{1}{2} \cdot \|M \backslash Z\| \cdot \|P\| \cdot (\|P\| - 1)$	105.270	595
(F)	$\|\mathcal{A}\| \cdot \|P\|$	1.260	1.260
Summe NB		114.032	7.132

Tabelle 2.1: Theoretische und reale Größe des Modells

Den größten Teil des Modells bestimmen die Binärvariablen und die Nebenbedingungen (R3), entsprechend stark werden sie die Lösbarkeit des Problems beeinflussen. Pro nicht freigegebener Typbelegung entfallen das entsprechende x_i^m und u_i^m sowie $|P|-1$ Variablen u_{ij}^m und die zugehörigen Bedingungen (I) und (R3). Mit praktischen Daten wird sich das Modell deutlich einfacher lösen lassen, da in nur wenigen Arbeitsgängen typwechselspezifische Rüstvorgänge stattfinden.

2.8 Komplexität

Wir wollen nun eine Aussage zur Komplexität unseres Problems treffen. Die dafür notwendigen Begriffe werden im Folgenden kurz erklärt. Für eine exakte Definition sei auf entsprechende Literatur verwiesen, [26] empfiehlt z.B. [5].
Entscheidungsprobleme sind Probleme, die entweder mit ja oder nein beantwortet werden können. Jedes Optimierungsproblem lässt sich in eine Klasse von Entscheidungsproblemen umwandeln. Für unsere Kapazitätsplanung lässt sich diese Klasse folgendermaßen definieren:

KEP$_r$ (Kapazitätsentscheidungsproblem) mit $r \in \mathbb{R}_+$:
Existiert für eine gegebene Instanz eine Typbelegung, so dass $f(x, u) \leq r$?

Komplexität bezeichnet die „Kompliziertheit“ eines Problems. Eine Klasse von Problemen in der Komplexitätstheorie ist NP (Abkürzung für nichtdeterministisch polynomiell). Für Entscheidungsprobleme, die in NP liegen, lässt sich in polynomieller Zeit in Abhängigkeit

von der Eingabelänge der Instanz für eine gegebene Lösung bestätigen, ob dies tatsächlich eine Lösung ist.
KEP_r liegt in NP, denn für eine gegebene Lösung sind nur alle Variablen x_i^m mit $m \in Z$ zu addieren, um die Aussage zu verifizieren. Die Bestätigung der Zulässigkeit der „geratenen“ Lösung durch Einsetzen in die Nebenbedingungen ist ebenfalls in polynomieller Zeit möglich.
Ein Problem p heißt NP-schwer, wenn sich jedes Problem aus NP in deterministisch polynomieller Zeit auf p reduzieren lässt. Ein NP-schweres Problem ist also mindestens so schwer wie das schwerste Problem in NP. Wegen der Transitivität von Polynomialzeitreduktion reicht es für den Beweis, dass p NP-schwer ist, zu zeigen, dass sich ein bekanntes NP-schweres Problem auf p reduzieren lässt. NP-schwere Probleme, die selbst in NP liegen, heißen NP-vollständig. Diese Probleme gehören zu den schwersten in NP und es ist ungeklärt, ob für diese Klasse ein polynomieller Algorithmus existiert.
Ein Problem, für das die NP-Vollständigkeit bereits von Karp in [20] nachgewiesen wurde, ist die Knotenfärbung in Graphen:

Gegeben sei ein ungerichteter Graph $G = (V, E)$ ohne Mehrfachkanten. Eine gültige *Knotenfärbung* ordnet jedem Knoten $v \in V$ eine natürliche Zahl bzw. „Farbe“ zu, so dass benachbarte Knoten nicht die gleiche Farbe besitzen. Das entsprechende Entscheidungsproblem fragt, ob es eine zulässige Knotenfärbung mit höchstens k Farben gibt.

Wir werden nun die Knotenfärbung auf das Problem KEP_0 reduzieren. Ist dieses Problem NP-vollständig, sind auch alle anderen Entscheidungsprobleme dieser Klasse und damit das ursprüngliche Optimierungsproblem NP-vollständig.

Behauptung:
Das Problem KEP_0 ist NP-vollständig.

Beweis:
Wir wandeln das Knotenfärbungsproblem für $k \in \mathbb{N}_0$ und einen gegebenen Graphen $G = (V, E)$ folgendermaßen in ein Entscheidungsproblem KEP_0 mit einem Arbeitsgang A und k Maschinen um: die Maschinen repräsentieren verschiedene Farben, die Produkte seien identisch mit den Knoten. Es muss über die Rüstzeiten verhindert werden, dass benachbarte Knoten derselben Maschine bzw. Farbe zugeordnet werden. Da es sich hier um einen einstufigen Produktionsprozess handelt, wird die Menge $\mathcal{N}_i$ nicht benötigt.

Es seien also

$$
\begin{aligned}
\mathcal{A} &:= A \\
M_A &:= \{1, \ldots, k+1\} \\
Z &:= \{k+1\} \\
M &:= M_A \\
P &:= \{v : v \in V\} \\
\mathcal{A}_i &:= (A), v \in V \\
P_m &:= P, m = 1, \ldots, k+1 \\
c_m &:= |V|, m = 1, \ldots, k \\
d_v &:= 1, v \in V \\
t_v^m &:= 1, v \in V, m = 1, \ldots, k \\
z_v^m &:= 0, v \in V, m = 1, \ldots, k \\
r_{vw}^m &:= \begin{cases} |V|+1, (v,w) \in E, \\ 0 \text{ sonst} \end{cases} , m = 1, \ldots, k \\
a_v^m &:= 1, v \in V, m = 1, \ldots, k+1.
\end{aligned}
$$

Werden zwei benachbarte Knoten derselben Maschine zugeordnet, wird wegen der Rüstzeit die Kapazitätsbedingung verletzt. Die übrigen Parameter sind so gewählt, dass jede Maschine mit beliebig vielen nichtbenachbarten Knoten belegt werden kann.Wird nun das Knotenfärbungsproblem mit ja beantwortet, kann auch KEP_0 positiv beantwortet werden. Wir setzen $x_v^m = 1$, wenn m der gewählten Farbe für v entspricht und 0 sonst. Da jedem Knoten eine zulässige Farbe zugewiesen werden konnte, können auch alle Bedarfe erfüllt werden, ohne die Kapazitätsbedingung zu verletzen.
Können wir umgekehrt KEP_0 bejahen, erhalten wir sofort eine gültige k-Färbung für G, indem wir als Farbe für v die zugewiesene Maschine mit minimalem Index wählen.

Kapitel 3

Diskussion des Modells

3.1 Kapazitäts- vs. Fertigungsplanung

Unsere Aufgabe ist die Kapazitäts- und Investitionsplanung, in der die lang- und mittelfristige Verfügbarkeit von Betriebsmitteln gewährleistet wird. Im nächsten Planungsschritt, der Fertigungsplanung, werden Losgrößen, Fertigungstermine und Reihenfolgen festgelegt (das sogenannte Scheduling). Diese Aspekte, insbesondere die Festlegung von Produktionsreihenfolgen, können zu einem abweichenden Kapazitätsbedarf führen, für den bisher mit Hilfe der technischen, organisatorischen und rüstbedingten Verfügbarkeit der Anlagen ein Anteil der Kapazität reserviert wurde.
In der Literatur ist das Problem der Produktionsplanung mit reihenfolgeabhängigen Rüstzeiten bekannt. Vom Grundproblem ist das *Capacitated Lot-Sizing* (CLS, [23]) unserer Aufgabe recht ähnlich: Ziel ist es hier, zu gegebenen Bedarfen für mehrere Perioden die zu produzierenden Mengen eines jeden Produktes für jede Periode zu identifizieren, meistens unter Minimierung von fixen und variablen Lagerhaltungs- und Produktionskosten. Dabei ist die Produktion einstufig. CLS enthält im Allgemeinen Flusserhaltungsbedingungen für die Lagerhaltung, Fixkostenungleichungen (I) und Kapazitätsbeschränkungen (K) ohne Rüstzeiten. Wir können die Flusserhaltung in CLS uminterpretieren, indem wir statt der Perioden die aufeinanderfolgenden Arbeitsgänge als Stufen für die Flusserhaltung verwenden sowie statt Variablen für Lagermengen unsere $x_i^{z_A}$. Bei reihenfolgeabhängigen Rüstzeiten reicht diese einfache Betrachtungsweise wie in CLS nicht aus. So entstand das Simultane Lot-Sizing und Scheduling, indem die Planungsperioden in Mikroperioden geteilt werden und die Produktionsmengen einer Mikroperiode als Fertigungstermin zugeordnet werden. Häufig wird die reihenfolgeabhängige Rüstzeit als Kostenfaktor in die Zielfunktion übernommen. In [12] tritt sie als Term in der Kapazitätsbedingung auf. Diese komplexen Optimierungsprobleme behandeln zwar Rüstzeiten exakt in Abhängigkeit von der gewählten Reihenfolge, werden in der Literatur jedoch überwiegend mittels Näherungsverfahren wie Threshold Accepting oder Simulated Annealing gelöst. Diese aufwändige Betrachtung wollen und müssen wir wie eingangs erklärt umgehen, da uns wichtige Parameter und Zusammenhänge wie Liefertermine, erwartete Verzögerungen wegen feh-

lender Bauteile und anderer Einflussgrößen der Fertigungsplanung unbekannt sind. Im Folgenden wollen wir untersuchen, welche Konsequenzen das Weglassen einer Reihenfolge auf die Genauigkeit der Planung hat.

3.2 Vorteile gegenüber der bisherigen Planung

Unser Ziel ist es, die starre Linienbelegung und Einteilung in Rüst- und Nichtrüstmaschinen aufzuweichen, um die Flexibilität der Linien zu nutzen.
Folgendes Beispiel soll diese Problematik veranschaulichen:
Es sei A ein Arbeitsgang mit 3 Maschinen, welche die gleichen Taktzeiten, Kapazitäten und Rüstzeiten besitzen. In der bisherigen Planung wurden Prioritäten für die Typbelegung festgelegt: Maschine 1 bearbeitet zuerst Typ 1, Restkapazität wird für die Bearbeitung von Typ 2 genutzt. Analog bearbeitet Maschine 2 bevorzugt Typ 3, anschließend Typ 4. Maschine 3 als Rüstlinie bearbeitet den Rest aller Aufträge. Ein Stückzahlszenario kann in diesem Fall zu einer Typbelegung wie in Abbildung 3.1 auf Seite 18 führen. Die Fläche eines Rechteckes symbolisiert die notwendige Bearbeitungszeit des Types auf der jeweiligen Maschine. Hier müsste mindestens dreimal gerüstet werden. In dieser Situation erscheint es jedoch günstiger, Typ 2 komplett auf Linie 1 zu bearbeiten, wie in Abbildung 3.2, dann muss auf der Linie gar nicht gerüstet werden.
Es kann also durchaus von Nutzen sein, die Maschinenbelegungen variabel in Abhängigkeit von der Bedarfssituation zu wählen.

3.3 Genauigkeit der Rüstzeiten

Die zu produzierende Menge x_i^m eines Types auf einer Maschine wird häufig in der Terminplanung in kleinere Produktionsaufträge geteilt, welche nicht unmittelbar hintereinander bearbeitet werden. Weder die Anzahl der so entstehenden Teilaufträge noch die Termine der Produktion sind zum Zeitpunkt der Kapazitätsplanung bekannt oder beeinflussbar.
Im Modell vernachlässigen wir deshalb die zeitliche Festlegung der Produktion der Mengen x_i^m und berechnen nur einen Durchschnittsrüstwert für die gemeinsame Zuordung zweier Typen zu einer Linie innerhalb eines Produktionszeitraumes. Statt einer linearen Abfolge der Produktionsmengen beziehen wir alle paarweisen Rüstzeiten für eine bestimmte Auswahl an zulässigen Typen in den Kapazitätsbedarf ein. Die Rüstzeiten für jede Paarung werden mittels empirischer Durchschnittshäufigkeiten gewichtet.
Diese Modellierung hat den Vorteil, dass das Modell stark reduziert wird und keine Festsetzung von Mikroperioden notwendig ist. Allerdings ist das Ergebnis ungenau, da bei der späteren Umsetzung ein bestimmter Rüstwechsel seltener oder häufiger als im Modell veranschlagt auftreten kann. Dies ist kein Problem, wenn wir für die Differenz zwischen Modellrüstzeit und realer Reihenfolge vernünftige Schranken angeben können, innerhalb derer auf Abweichungen zwischen Modell und Realität reagiert werden kann. Wir können z.B. folgende Forderung an die Modellrüstzeit stellen:

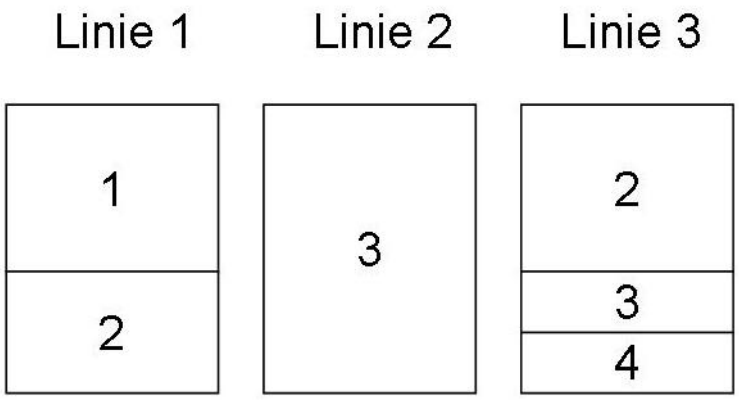

Abbildung 3.1: Typbelegung nach Prioritäten

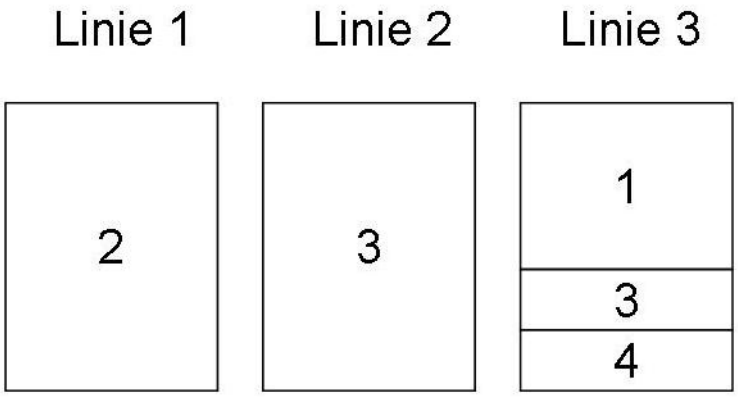

Abbildung 3.2: Günstigere Typbelegung

Die Modellrüstzeit soll die reale Rüstzeit nicht um mehr als 2 Schichten zu je 8 Stunden unterschreiten (möglicher Handlungsspielraum).

Um eine Abschätzung der Abweichung zwischen Modell und Realität zu erhalten, legen wir zwei Prämissen für die Planung fest:

- Die Maschinenzuordnung aus dem Modell wird in die Fertigungsplanung übernommen.
- Die Teilung der zu produzierenden Mengen vom Typ i für einen Monat erfolgt in maximal 4 Teilmengen.

Der erste Punkt erfordert eine Kommunikation mit der für die Fertigungsplanung verantwortlichen Abteilung. Die zweite Annahme resultiert aus den wöchentlichen Abrufterminen für jedes Produkt bei einer Periodenlänge von einem Monat.

Wir wollen nun den Unterschied zwischen maximaler realer Rüstzeit R_{max} und modellierter Rüstzeit R_M auf einer Maschine m bewerten. Es sei $B \subset P_m$ eine Menge von Produkten, die auf Maschine m bearbeitet werden sollen. Die Modellrüstzeit für diese Zuordnungsmenge ist unabhängig von der Reihenfolge

$$R_M(B) = \sum_{\substack{i,j \in B \\ i<j}} h_{ij}^m \hat{r}_{ij}^m,$$

wobei $\hat{r}_{ij}^m$ die Rüstzeit für einen Wechsel von i nach j und h_{ij}^m die im Modell verwendete, konstante Durchschnittshäufigkeit in einem Monat seien.

Wir wollen sicher stellen, dass Mehrrüstzeit mit 2 Schichten abgedeckt werden kann:

$$R_{max}(B) - R_M(B) \leq 2 \text{ Schichten} = 16 \text{ Stunden} =: S \tag{3.1}$$

Bei $|B|$ verschiedenen Typen, die in jeweils höchstens 4 Teilmengen bearbeitet werden, finden maximal $4|B| - 1$ Rüstwechsel statt.
Es seien

$$\hat{r}_{max} := \max_{i,j \in B} \hat{r}_{ij}^m$$
$$r_{min} := \min_{i,j \in B} h_{ij} \hat{r}_{ij}^m,$$

d.h. $\hat{r}_{max}$ ist die maximale Rüstzeit für einen Wechsel auf Linie m und r_{min} die minimale monatliche Rüstzeit. In der Realität findet unter den obigen Prämissen im schlimmsten Fall $(4|B| - 1)$-mal der maximale Rüstwechsel statt. Im Modell wird dagegen für jedes

Paar $(i,j) \in B$ mindestens ein Wert der Höhe r_{min} berechnet. Es gilt somit

$$\begin{aligned}
&R_{max}(B) - R_M(B) \\
\leq &(4|B|-1)\hat{r}_{max} - \sum_{\substack{i,j\in B \\ i<j}} h_{ij}^m \hat{r}_{ij}^m \\
\leq &(4|B|-1)\hat{r}_{max} - \sum_{\substack{i,j\in B \\ i<j}} r_{min} \\
\leq &(4|B|-1)\hat{r}_{max} - \frac{|B|(|B|-1)}{2} r_{min} \\
= &-\frac{1}{2}|B|^2 r_{min} + |B|(4\hat{r}_{max} + \frac{1}{2}r_{min}) - \hat{r}_{max} \\
=: &g(|B|).
\end{aligned}$$

Es ist also für jede Maschine m die Bedingung zu prüfen

$$g(|B|) \leq S \quad , 2 \leq |B| \leq |P_m|. \tag{3.2}$$

Ist diese erfüllt, gilt auch (3.1). Damit kann unter den obigen Prämissen die Abweichung der Modellrüstzeit von der Realität mit zwei Zusatzschichten abgedeckt werden. Trotz der groben Abschätzung ist Bedingung (3.2) bei den uns vorliegenden Daten für alle relevanten Maschinen erfüllt. Die betrachteten Maschinen besitzen nur wenige Typfreigaben, d.h. $|P_m|$ ist klein, oder die vorliegenden Rüstzeiten liegen nicht sehr weit auseinander.
Im Allgemeinen können wir davon ausgehen, dass schlechte Reihenfolgen unwahrscheinlicher sind als günstige. Durch die Verwendung der empirischen mittleren Rüsthäufigkeiten zuzüglich eines Sicherheitsquantiles sollte die Differenz zwischen Real- und Modellrüstzeit im Mittel nicht positiv sein.

3.4 Stillstandzeiten

Stillstandzeit ist die Zeit, in der eine Maschine aus technischen oder organisatorischen Gründen nicht genutzt oder gerüstet werden kann. Sie verringert die zeitliche Verfügbarkeit von Maschinen und damit deren Kapazität. Technische Störungen sind nicht vorhersehbar, während ablaufbedingter Stillstand durch die Planung der Reihenfolge beeinflusst werden kann.
In der Umsetzung des Modellierungsergebnisses dürfen die Planungsperioden zwar verschoben, aber nicht unterbrochen bzw. auseinandergerissen werden.
Wir wollen ein Beispiel untersuchen, in dem bei der Planung der Reihenfolge bei gegebener Maschinenbelegung Probleme auftreten können. Hier nehmen wir erneut an, dass die Maschinenzuordnung aus der Modelloptimierung in die Reihenfolgeplanung übernommen wird. Für die Typbezeichnungen verwenden wir Großbuchstaben. Rüstzeiten werden vernachlässigt und die Maschinenkapazitäten seien alle gleich der Planungsperiode. Außerdem wird angenommen, dass kein Überlappen stattfindet. Beim Überlappen werden

Aufträge in kleinere Teillose zerteilt, welche nach ihrer Bearbeitung an den nächsten Arbeitsgang weitergegeben werden. Ohne Überlappen muss dort auf den kompletten Auftrag gewartet werden.
Reihenfolgebedingter Stillstand tritt auf, wenn zwei Typen auf einer Maschine bearbeitet werden sollen, die im vorigen Arbeitsgang zu sehr unterschiedlichen Zeitpunkten fertiggestellt werden. Man kann Beispiele konstruieren, in denen jede Reihenfolge zu einem Stillstand führt. In 3.3 haben wir eine solche Situation abgebildet. Diese drei Fälle bilden alle wesentlichen Konstellationen ab, andere Reihenfolgen sind wegen der Äquivalenz von A und C bzw. D, E und F im Ergebnis analog.
Dieses Beispiel ist durchaus realistisch, wenn z.B. die Typfreigaben keine andere Belegung in Arbeitsgang 2 erlauben und die Bedarfe von D, E, F und B genügend hoch sind. Wie in Abb. 3.3 zu sehen, entsteht bei jeder Reihenfolge maschineller Stillstand (dunkel gefärbt). In unserem Beispiel entsteht jeweils Stillstand von einem Sechstel einer Periode.
Diese Lücken können theoretisch geschlossen werden, indem der Produktionsstart einiger Typen nach hinten geschoben wird, so dass die Aufträge trotzdem zusammenhängend bearbeitet werden können. Dann müssen allerdings viele Teile zwischengelagert werden. Lagerplatz wird aber schon durch Teile beansprucht, die später mit ihrer Bearbeitung beginnen, weil auf der nächsten Maschine noch ein anderer Auftrag bearbeitet wird.
Liefertermine können außerdem ein Verschieben der Bearbeitungszeiten verhindern. Es kann sogar passieren, dass Liefertermine mit der berechneten Belegung nicht einhaltbar sind.
Insgesamt kann es also in der Fertigungsplanung mit der vorgegebenen Belegung zu Verzögerungen und langen Durchlaufzeiten kommen. Durch Rüst-, Transport- und Liegezeiten kann sich das Ergebnis ändern. Maschinenstillstand kann verkürzt werden durch Überlappen, Planung von Pufferbeständen und Durchführen von Wartungsarbeiten in längeren Pausen. Die bisherige Trennung von Kapazitätsplanung und Scheduling ist jedoch erfahrungsgemäß erfolgreich.

3.5 Unsicherheit von Daten

Fast alle Daten für unser Modell sind Schwankungen und Unsicherheiten unterworfen. Taktzeiten, Yields und Zweitläuferquoten sind stochastische Größen, die nur über die Zeit gemittelt werden können. All diese Daten sind jedoch im Hinblick auf ihre Auswirkungen auf die Kapazität von geringer Bedeutung und haben keinen wesentlichen Einfluss auf die Planung, sondern führen nur zu kleineren Ungenauigkeiten, wenn sie regelmäßig aktualisiert werden.
Im Gegensatz dazu können längerer Maschinenausfall und variierende Stückzahlszenarien das Ergebnis der Planung maßgeblich beeinflussen. Für Maschinen müssen von Zeit zu Zeit Umbau-, Instandhaltungs- und Reparaturmaßnahmen geplant werden, deren Stillstand auch andere Arbeitsgänge beeinflusst: wenn in einem Arbeitsgang eine Maschine stillsteht, kommt es in Nachfolgearbeitsgängen ebenfalls zu Verzögerungen.
Die Stückzahlbedarfe sind auch nicht fixiert, weil ein Kunde seine Nachfrage in vertrag-

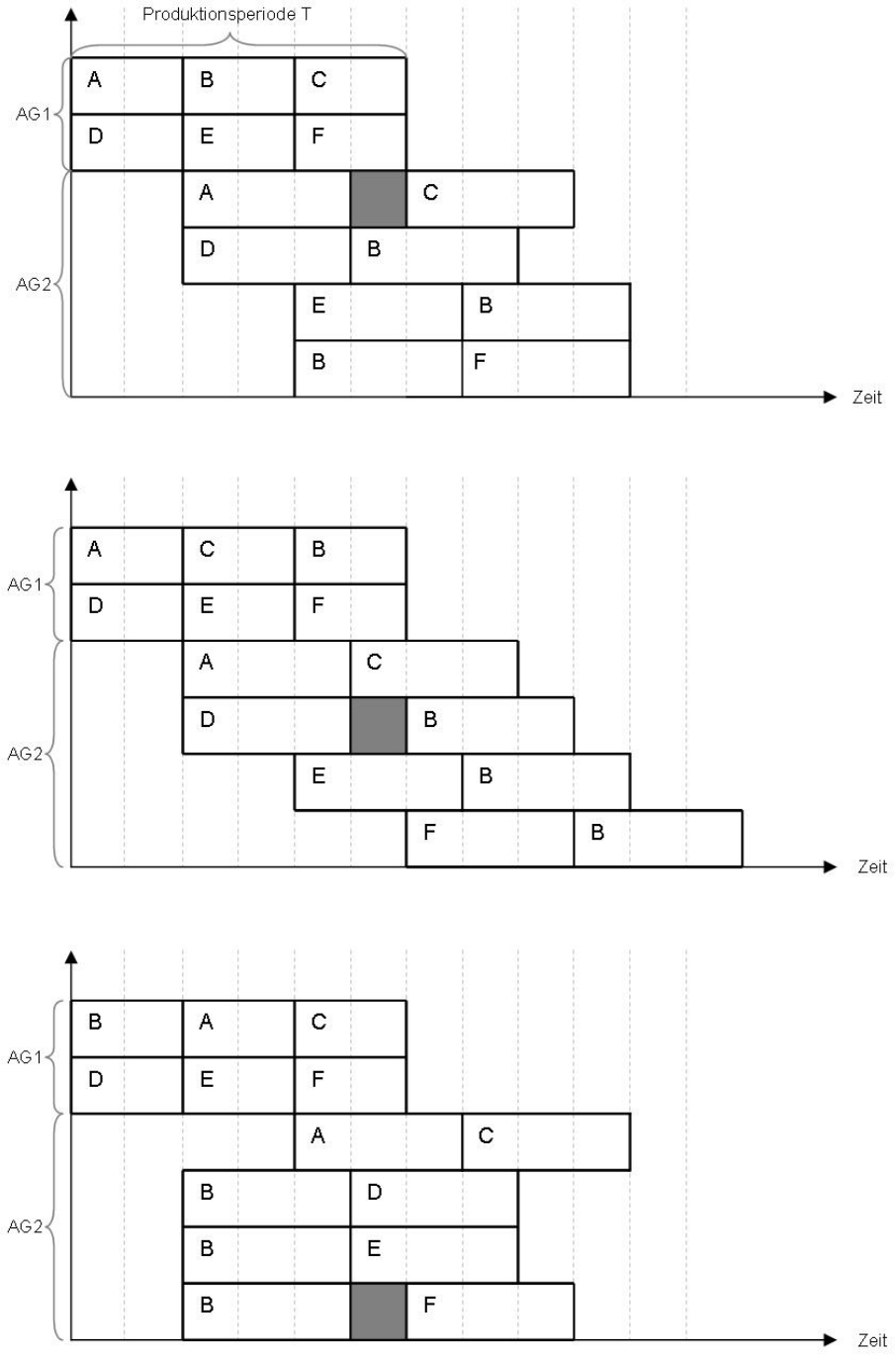

Abbildung 3.3: Ablaufbedingter Maschinenstillstand

lich festgelegten Grenzen herabsetzen oder erhöhen darf. Es kommt recht häufig eine Kundenanfrage, ob ein Mehrbedarf bewerkstelligt werden kann.
Man muss also das Modell mit vielen verschiedenen Maschinenausfall- und Stückzahlszenarien mehrmals lösen, wenn man verschiedene mögliche Situationen vergleichen möchte, dabei kann für jedes Szenario ein anderer Produktionsplan entstehen. Auswege bieten die stochastische und robuste Optimierung, die in Kapitel 6 vorgestellt werden. Wir wollen uns zunächst mit der Lösung des Problemes für feste Kapazitäten und Bedarfe beschäftigen.

Kapitel 4

Exakte Lösungsverfahren

Einen guten Überblick über ganzzahlige Optimierung liefert [26]. Die Ideen und Definitionen in diesem Kapitel sind überwiegend an die dortige Darstellung angelehnt.

4.1 Das Prinzip exakter Lösungsverfahren

Wir betrachten in diesem Kapitel die allgemeine gemischt-ganzzahlige lineare Optimierungsaufgabe

$$\min\{c^T x + d^T u : Ax + Gu \leq b, x \in \mathbb{R}^n_+, u \in \mathbb{Z}^p_+\} \tag{MIP}$$

mit $c \in \mathbb{R}^n, d \in \mathbb{R}^p, A \in \mathbb{R}^{m \times n}, G \in \mathbb{R}^{m \times p}, b \in \mathbb{R}^m$.
Es bezeichne

$$X := \{(x, u) \in \mathbb{R}^n \times \mathbb{Z}^p : Ax + Gu \leq b\}$$

die zulässige Menge von (MIP).
Alle praktisch relevanten exakten Verfahren zur Lösung von (MIP) beruhen auf der iterativen Lösung und Modifikation einer Relaxierung $\min\{\varphi(x, u) : (x, u) \in T \subset \mathbb{R}^{n+p}\}$, also eines einfacheren Problems, für das gilt:

$$X \subseteq T \quad \text{und} \quad \varphi(x, u) \leq c^T x + d^T u \quad \forall (x, u) \in X.$$

Am häufigsten wird die LP-Relaxierung verwendet, die die Ganzzahligkeitsbedingung des Ursprungsproblems weglässt.
Es sei

$$\min\{c^T x + d^T u : Ax + Gu \leq b, x \in \mathbb{R}^n_+, u \in \mathbb{R}^p_+\} \tag{MIP$_{LP}$}$$

die LP-Relaxierung von (MIP). In einem 0-1-Problem wird $u \in \{0, 1\}^p$ ersetzt durch $u \in [0, 1]^p$.
Die Optimallösung der Relaxierung liefert (bei Minimierung) eine untere Schranke für den Optimalwert des Ausgangsproblems, da die zulässige Menge größer ist als die des ganzzahligen Problems. Gleichzeitig definiert der Wert jeder zulässigen Lösung $(x, u) \in X$ eine obere Schranke für den Wert einer Optimallösung.

Im Laufe des Algorithmus wird versucht, durch Verschärfung der Relaxierung die untere Schranke anzuheben und gleichzeitig durch das Finden einer guten zulässigen Lösung die obere Schranke zu verbessern. Zwei bekannte und bewährte Verfahren dieser Art sind Schnittebenenverfahren und die Branch-and-Bound-Methode.

4.2 Gute Formulierungen

Definition:
Ein Polyeder $P \subset \mathbb{R}^{n+p}$ heißt *Formulierung* einer Menge $X \subset \mathbb{R}^n \times \mathbb{Z}^p$, wenn $X = P \cap (\mathbb{R}^n \times \mathbb{Z}^p)$.

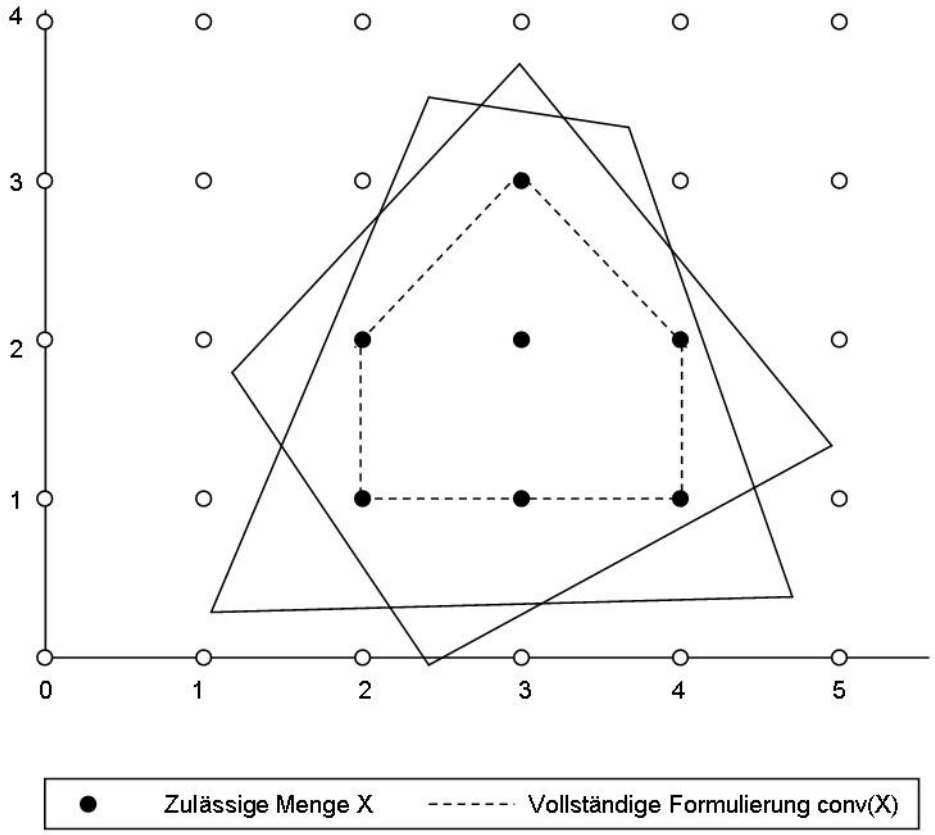

Abbildung 4.1: Verschiedene Formulierungen für eine Menge ganzzahliger Lösungen

Für gemischt-ganzzahlige lineare Optimierungsprobleme sind Formulierungen nicht eindeutig, wie Abb. 4.1 (nach [26, S.15]) veranschaulicht. Die Wahl der Formulierung kann entscheidend für den Erfolg eines Lösungsverfahrens sein. Sind alle Ecken einer Formulierung P in X enthalten, spricht man von einer *vollständigen* Formulierung. Mit dieser

Eigenschaft kann das Problem

$$\min\{c^T x + d^T u : (x, u) \in X\}$$

für beliebige Zielfunktionskoeffizienten (c^T, d^T) durch

$$\min\{c^T x + d^T u : (x, u) \in P\}$$

ersetzt und mit Methoden der linearen Optimierung wie dem Simplex-Verfahren gelöst werden.

Satz: Existenz vollständiger Formulierungen (nach Padberg [16])

Sei X die zulässige Menge von (MIP) mit A, G, b rational und

$$conv(X) = \{y : y = \sum_{i=1}^{t} \lambda_i y_i, \sum_{i=1}^{t} \lambda_i = 1, \lambda_i \geq 0, \text{für alle endlichen Teilmengen} \{y_1, .., y_t\} \text{von } X, t \in \mathbb{N}\}$$

die konvexe Hülle von X.

1. Es ist $conv(X)$ ein Polyeder in $\mathbb{R}^{n+p}$, d.h. es existieren $k \in \mathbb{Z}$, $0 \leq k < \infty$ und rationale Matrizen $H \in \mathbb{R}^{k \times n}$, $I \in \mathbb{R}^{k \times p}$ und $h \in \mathbb{R}^k$, so dass

   $$conv(X) = \{(x, u) \in \mathbb{R}^{n+p} : Hx + Iu \leq h, x \geq 0, u \geq 0\}.$$

2. Es ist $conv(X)$ eine vollständige Formulierung von X, d.h.

   $$\min\{c^T x + d^T u : (x, u) \in X\} = \min\{c^T x + d^T u : (x, u) \in conv(X)\}$$

   für beliebige $(c^T, d^T) \in \mathbb{R}^{n+p}$.

3. Es existiert ein endlicher Algorithmus zur Lösung von (MIP).

Ausführliche Beweise finden sich ebenfalls in [16]. Aussage (3) folgt aus der Reduktion von (MIP) auf ein lineares Programm in (2) und der Endlichkeit des Simplex-Algorithmus. Der Satz liefert lediglich eine Existenzaussage für vollständige Formulierungen wie $conv(X)$, jedoch keine konstruktive Aussage zur Ermittlung der Darstellung in (1). Für gemischt-ganzzahlige Probleme ist die konvexe Hülle der zulässigen Lösungen meist entweder nicht vollständig bekannt oder es sind sehr viele Ungleichungen notwendig, um sie zu beschreiben. Die schärfsten gültigen Ungleichungen sind Facetten von $conv(X)$:

Definition:
Sei P ein Polyeder in $\mathbb{R}^n$ und $\pi x \leq \pi_0$ mit $\pi \in \mathbb{R}^n, \pi_0 \in \mathbb{R}$ gültig für alle $x \in P$.
Die Menge $F = \{x \in P : \pi x = \pi_0\}$ heißt *Seite* von P. Man sagt, die Ungleichung $\pi x \leq \pi_0$ *definiert* F.
Ist F eine Seite von P mit $dim(F) = dim(P) - 1$, heißt F *Facette*.

Für eine exakte Beschreibung der konvexen Hülle aller zulässigen Punkte von (MIP) benötigt man eine definierende Ungleichung für jede Facette. Besitzt $conv(X)$ nicht volle Dimension, existieren außerdem Gleichungen, die für alle $x \in X$ erfüllt werden und die affine Hülle von $conv(X)$ beschreiben.
Da man nicht hoffen kann, alle Facetten von $conv(X)$ zu finden, sucht man stattdessen nach möglichst guten Formulierungen.

Preprocessing

Preprocessing-Techniken, auch Presolve genannt, dienen der Verbesserung der Formulierung eines gemischt-ganzzahligen Optimierungsproblems. Dabei wird untersucht, ob a priori redundante Nebenbedingungen entfernt, Schranken verschärft, Unzulässigkeit entdeckt und Variablen fixiert werden können. Beim Probing werden logische Bedingungen der Art „ $x_i = 0 \Rightarrow x_j = 0$ “ erzeugt und der Formulierung hinzugefügt. Hier soll nur auf ein paar grundlegende Techniken aus [21] eingegangen werden, die in unserem Problem anwendbar sind.

Verbesserung von Schranken

Wegen der Flusserhaltungsbedingungen (F) und $0 \leq a_i^m \leq 1$ gilt:

$$\begin{aligned}
\sum_{m \in M_{N_k^{(i)}}} x_i^m = \sum_{m \in M_{A_k^{(i)}}} a_i^m x_i^m \quad &\leq \sum_{m \in M_{A_k^{(i)}}} x_i^m && \forall i \in P, k = 1, \ldots, n_i - 1 \\
\Rightarrow \qquad d_i &\leq \sum_{m \in M_A} x_i^m && \forall A \in \mathcal{A}_i \\
\Rightarrow \qquad x_i^{z_A} &\geq - \sum_{m \in M_A \setminus \{z_A\}} S_i^m + d_i =: U_i^{z_A} && \forall A \in \mathcal{A}_i
\end{aligned}$$

Dabei sei S_i^m eine obere Schranke für x_i^m. Die untere Schranke von $x_i^{z_A}$ ist also $\max\{0, U_i^{z_A}\}$. Damit lässt sich auch a priori eine untere Schranke für den Zielfunktionswert angeben:

$$f(x,u) \geq \sum_{\substack{i \in P \\ m \in Z}} \max\{0, U_i^m\}$$

Für die meisten Werte wird gelten $U_i^{z_A} < 0$, so dass diese untere Schranke i.d.R. deutlich schlechter ist als die lineare Relaxierung.

Bei der Modellierung haben wir bereits vorgestellt, dass eine obere Schranke für x_i^m durch

$$S_i^m = \frac{c^m}{t_i^m(1+z_i^m)}$$

berechnet werden kann. Bei kleinen Bedarfen d_i kann diese Schranke möglicherweise verschärft werden: es sei

$$\hat{a}_i^{A_l^{(i)}} = \min_{m \in M_{A_l^{(i)}}} \{a_i^m\}$$

der minimale Outputfaktor für Produkt i in Arbeitsgang l und für alle $m \in M_{A_k^{(i)}}$

$$\Lambda_i^m := \prod_{l>k} \hat{a}_i^{A_l^{(i)}}$$

das Produkt der minimalen Yields aller Nachfolgearbeitsgänge von $A_k^{(i)}$. Dann ist

$$\bar{S}_i^m := d_i \cdot \frac{1}{\Lambda_i^m}$$

der maximale Bedarf von Produkt i auf Maschine m. Die folgende Preprocessingtechnik ist nur erfolgreich, wenn $\bar{S}_i^m < S_i^m$.

Verschärfung von Koeffizienten
Wir betracheten die Kapazitätsbedingung für eine Maschine m und eine Variable u_{kl}^m. Angenommen, es gilt

$$\sum_{i \in P_m} t_i^m(1+z_i^m)\bar{S}_i^m + \sum_{\substack{i,j \in P_m \\ j>i \\ \{i,j\} \neq \{k,l\}}} r_{ij}^m < c^m,$$

dann ist mit

$$\delta := c^m - \sum_{i \in P_m} t_i^m(1+z_i^m)\bar{S}_i^m - \sum_{\substack{i,j \in P_m \\ j>i \\ \{i,j\} \neq \{k,l\}}} r_{ij}^m$$

offenbar für alle zulässigen Lösungen sowohl mit $u_{kl}^m = 0$ als auch mit $u_{kl}^m = 1$ die Ungleichung

$$\sum_{i \in P_m} t_i^m(1+z_i^m)x_i^m + \sum_{\substack{i,j \in P_m \\ j>i \\ \{i,j\} \neq \{k,l\}}} r_{ij}^m u_{ij}^m + (r_{kl}^m - \delta)u_{kl}^m \leq c^m - \delta$$

erfüllt. Die Regel kann iterativ mit einer weiteren Variable auf die verschärfte Ungleichung angewendet werden. Sie ist dann hilfreich, wenn auf einer Maschine wenige Typen mit kleinen oberen Schranken $\bar{S}_i^m$ freigegeben sind und hohe Rüstzeiten auftreten.

4.3 Branch-and-Bound

Branch-and-Bound-Verfahren lösen das ganzzahlige Optimierungsproblem mittels kontrollierter, teilweiser Enumeration der ganzzahligen Lösungen. Ausgehend von einer optimalen, nicht-ganzzahligen Lösung einer Relaxierung werden durch Teilung der zulässigen Menge Unterprobleme erzeugt, denen so lange Restriktionen hinzugefügt werden, bis alle Ganzzahligkeitsbedingungen erfüllt sind.
Der Lösungsaufwand wird einerseits durch geschicktes Verzweigen (Branching) nach Teilproblemen und andererseits durch möglichst frühzeitiges Terminieren (Bounding) von nicht Erfolg versprechenden Teilen des so entstehenden Entscheidungsbaumes verringert. Die Qualität der berechneten oberen und unteren Schranken ist ein entscheidender Faktor für die Effizienz eines solchen Verfahrens.
Die wesentlichen Schritte eines B&B-Algorithmus sind:

Initialisierung:
Als Startwert für eine obere Schranke z^o (bei Minimierung) kann der Wert auf unendlich oder den Zielfunktionswert einer zulässigen Lösung, die z.B. durch eine Heuristik ermittelt wird, gesetzt werden. Die Lösung der Relaxierung liefert eine untere Schranke z_u. Die Kandidatenliste besteht zu Beginn nur aus dem Ausgangsproblem.

Knotenwahl
Solang die Kandidatenliste nicht leer ist, muss in diesem Schritt entschieden werden, welches Teilproblem als nächstes bearbeitet wird. Dabei gibt es verschiedene Varianten:
Bei der Tiefensuche werden Knoten gewählt, die am weitesten von der Wurzel (dem Ausgangsproblem) entfernt sind. Das sind die Knoten, in denen schon die meisten Variablen fixiert wurden. Diese Methode findet schnell zulässige Lösungen, die untere Schranke verbessert sich dabei jedoch nur langsam.
Umgekehrt wählt man bei der Breitensuche den Knoten mit dem kürzesten Abstand zur Wurzel. Man findet dadurch zwar nur langsam eine zulässige Lösung, diese ist aber i.d.R. recht gut.
Ein weiteres, häufig verwendetes Kriterium ist Best-lower-bound, bei dem das Problem mit der aktuell besten unteren Schranke gewählt wird.
Das gewählte Problem wird aus der Kandidatenliste entfernt.

Lösen der Relaxierung und Bounding
Das aktuelle Teilproblem wird relaxiert und gelöst. Nun können folgende Fälle auftreten:

1. Ausloten durch obere Schranke:
   Besitzt die Relaxierung keine zulässige Lösung (gleichzusetzen mit einem unendlichen Zielfunktionswert) oder ist die Lösung mindestens so groß wie die aktuelle obere Schranke und damit nicht besser als die bisher beste Lösung, kann die Suche in diesem Teil des Baumes beendet werden.
   Noch schärfer wird das Ausloten, wenn man eine problemspezifische Minimalverbes-

serung Δz einführt. Dann werden nur Teilbäume akzeptiert, deren relaxierte Lösung besser ist als $z^o - \Delta z$. Dann ist jedoch zu beachten, dass die am Ende gefundene Lösung nicht zwangsläufig das tatsächliche Optimum darstellt.

2. Ausloten durch Zulässigkeit:
   Ist die Lösung der Relaxierung zulässig für das Ausgangsproblem mit einem besseren Zielfunktionswert als in der aktuell besten Lösung, ist die Suche in diesem Zweig ebenfalls beendet. Die neue Lösung verbessert die obere Schranke und kann zum Ausloten von Problemen in der Warteliste genutzt werden.

3. Kann der Zweig nicht durch Fall 1 oder 2 ausgelotet werden, erfolgt die Teilung in weitere Unterprobleme **(Branching)**. Bei 0-1-Variablen entstehen zwei Teilprobleme, in denen eine Binärvariable jeweils auf 0 bzw. 1 gesetzt wird. Bei der Wahl der zu fixierenden Variablen versucht man, eine bestmögliche Verbesserung der Schranken zu erreichen. Mögliche Kriterien sind z.B. reduzierte Kosten oder die Stärke des Gebrochenseins. Die beiden neuen Teilprobleme kommen in die Kandidatenliste.

Terminierung
Wenn die Kandidatenliste leer ist, existiert entweder keine zulässige Lösung oder die aktuelle Lösung ist optimal. Es kann aber auch abgebrochen werden, wenn der Unterschied zwischen oberer und unterer Schranke für praktische Zwecke klein genug oder die aktuell beste Lösung besser als ein definierter Zielwert ist.

4.4 Schnittebenenverfahren und Separierungsalgorithmen

4.4.1 Prinzip

Ziel von Schnittebenenverfahren ist es, eine aktuelle, nicht-ganzzahlige Lösung (x^*, u^*) des relaxierten Problems (MIP_{LP}) durch eine zusätzliche Nebenbedingung von der zulässigen Menge X des Ausgangsproblems zu trennen, ohne zulässige Lösungen abzuschneiden.
Gomory hat in den 1950ern einen Algorithmus zum systematischen Finden von Schnittebenen entwickelt und in [6] gezeigt, dass durch das Hinzufügen endlich vieler Schnittebenen die Optimallösung von (MIP) gefunden werden kann. Leider haben sich die Schnittebenen von Gomory als nicht besonders stark erwiesen, da sie sehr langsam konvergieren. In jeder Iteration muss das Separierungsproblem gelöst werden:

Sei $(x^*, u^*) \in \mathbb{R}^{n+p}$ eine Lösung von (MIP_{LP}).
Bestimme, ob $(x^*, u^*) \in X$. Falls $(x^*, u^*) \notin X$, finde $\pi \in \mathbb{R}^{n+p}$, $\pi_0 \in \mathbb{R}$, sodass $\pi(x, u) \leq \pi_0$ $\forall (x, u) \in X$ und $\pi(x^*, u^*) > \pi_0$.

Später wurde von Grötschel, Lovász und Schrijver in [8] und [9] bewiesen, dass das Separierungsproblem genauso schwer ist wie die eigentliche Optimierung: für eine Klasse von Optimierungsproblemen existiert genau dann ein polynomieller Algorithmus, wenn die Familie der Separierungsprobleme dieser Klasse in polynomieller Zeit lösbar ist. Wie in Abschnitt 2.8 gezeigt, ist das betrachtete Problem NP-vollständig. Für diese Klasse sind keine polynomiellen Algorithmen bekannt.
Schnittebenenverfahren gelangten erst zum Erfolg durch die Entwicklung der Polyedertheorie und die Einführung starker, problemspezifischer Schnittebenen.

4.4.2 Spezielle Schnittebenen

Es existieren eine Vielzahl von Separierungsalgorithmen für allgemeine und spezielle gemischt-ganzzahlige Optimierungsaufgaben. Dazu gehören der Separierungsalgorithmus von Gomory und die Rundungsprozeduren von Chvátal und Schrijver, die in [1] und [17] vorgestellt werden. Eine Prozedur für gemischt-ganzzahlige Probleme mit Binärvariablen wird in [13] beschrieben.
Um gute Schnittebenen für unser Problem zu erhalten, wollen wir die Nebenbedingungen (R1)-(R3) untersuchen. Es ist zu beachten, dass die Erkenntnisse jeweils nur auf die Variablen einer Maschine anwendbar sind, denn die Beziehungen gelten nur für $\{u_i^m, u_{ij}^m : i, j \in P_m\}$ mit einem festen $m \in M \backslash Z$. Wir betrachten also im Folgenden ein konstantes m und substituieren der Einfachheit halber $u_i^m =: u_i$, $u_{ij}^m =: u_{ij}$. Es sei $q := |P_m|$, u der q-dimensionale Vektor $[u_i]_{i \in P_m}$ sowie v der $q(q-1)/2$-dimensionale Vektor der Variablen u_{ij}, wobei für die Indizierung die Ordnung $12, 13, .., 1q, 23, ..$ zugrundegelegt wird. Die Positionen (i, j) und (j, i) seien dabei identisch.

Wir bezeichnen die Nebenbedingungen wie folgt:

$$-u_i + u_{ij} \leq 0 \quad \text{(R1)}$$
$$-u_j + u_{ij} \leq 0 \quad \text{(R2)}$$
$$u_i + u_j - u_{ij} \leq 1 \quad \text{(R3)}$$
$$-u_{ij} \leq 0 \quad \text{(R4)}$$
$$u_i, u_{ij} \text{ ganzzahlig} \quad \text{(R5)}$$

Die zulässige Menge sei $X_R := \{(u,v) \in \mathbb{Z}^{q(q+1)/2} : (u,v) \text{ erfüllt} (R1) - (R4)\}$.

Die konvexe Hülle $conv(X_R)$ bildet das sogenannte Boolean Quadric Polytope, welches unter anderem von Padberg in [15] untersucht wurde. Die Bedingungen (R1)-(R4) sind Facetten von $conv(X_R)$.
Padberg gibt weitere Familien von Facetten an und zeigt, dass insgesamt mindestens $O(3^q)$ lineare Ungleichungen für die Beschreibung von $conv(X_R)$ notwendig sind. Ein Beispiel für weitere Facetten von $conv(X_R)$ sind die Dreiecksungleichungen:

$$u_i + u_j + u_k - u_{ij} - u_{ik} - u_{jk} \leq 1 \quad \text{(T1)}$$
$$-u_i + u_{ij} + u_{ik} - u_{jk} \leq 0 \quad \text{(T2)}$$
$$-u_j + u_{ij} - u_{ik} + u_{jk} \leq 0 \quad \text{(T3)}$$
$$-u_k - u_{ij} + u_{ik} + u_{jk} \leq 0 \quad \text{(T4)}$$

Damit werden u.a. folgende gebrochene Lösungen separiert:

T1: $u_i = u_j = u_k = \frac{1}{2}, u_{ij} = u_{ik} = u_{jk} = 0$
T2: $u_i = u_{ij} = u_{ik} = \frac{1}{2}, u_{jk} = 0$
T3: $u_j = u_{ij} = u_{jk} = \frac{1}{2}, u_{ik} = 0$
T4: $u_k = u_{ik} = u_{jk} = \frac{1}{2}, u_{ij} = 0$

Die Dreiecksungleichungen sind für X genau so gültig wie für X_R. Eine gültige Ungleichung sollte nur dann dem Ausgangsproblem hinzugefügt werden, wenn sie von der aktuellen Lösung der Relaxierung verletzt wird, da das Modell sonst sehr groß wird und schon für die Lösung der Relaxierung viel Zeit erforderlich ist. Leider beobachtet man bei unserem Problem, dass die relaxierte Lösung selten eine der Dreiecksungleichungen verletzt, da die oberen Schranken für x_i^m nicht besonders stark sind. Die lineare Relaxierung unseres Problems zeigt folgendes Verhalten: Die Menge x_i^m für $m \in M \backslash Z$ soll möglichst groß werden, was wegen (I) eine Erhöhung von u_i^m notwendig macht. Wegen (R3) führt eine Erhöhung der u_i^m wiederum zu einem Ansteigen der u_{ij}^m, was einen erhöhten Kapazitätsbedarf in (K) zur Folge hat. Durch die Höhe der Schranke S_i^m ist es jedoch möglich, u_i^m so klein

zu halten, dass (R3) auch für $u_{ij}^m = 0$ erfüllt ist. Die relaxierte Lösung führt also zu einer Lösung mit $u_{ij}^m = 0$ und sehr kleinen gebrochenen u_i^m, welche die Bedingungen (T1)-(T4) nicht verletzen. Die Dreiecksungleichungen sind also nur hilfreich, wenn bereits einige u_i^m auf 1 fixiert wurden. Deshalb wäre es sinnvoll diese Ungleichungen als Schnittebenen mit einem Branch-and-Bound-Algorithmus zu verbinden.

4.4.3 Branch-and-Cut

Insgesamt sind Schnittebenenverfahren als alleinige Lösungsverfahren als kritisch anzusehen, da die Algorithmen in der Praxis oft ein schlechtes Laufzeitverhalten zeigen. Deshalb werden sie häufig in eine Branch-and-Bound-Umgebung eingebettet. Schnittebenen können in ein Branch-and-Bound-Schema eingebunden werden, um Relaxierungen zu verschärfen und in den Zweigen schneller zu zulässigen Lösungen zu gelangen. Sie werden dann hinzugefügt, wenn ein Knoten nicht sofort durch Lösen der Relaxierung ausgelotet werden kann. Statt einer Verzweigung in neue Teilprobleme wird durch die Einschränkung des Lösungsraumes versucht, doch noch ein Ausloten zu erreichen.
Die Schnittebenen können entweder global gültig, also für das ursprüngliche Problem zulässig, oder nur lokal für den aktuellen Teilbaum mit seinen Einschränkungen gültig sein. Die Verwendung lediglich lokal gültiger Schnittebenen erfordert einen höheren Speicherbedarf. Der Separierungsalgorithmus wird abgebrochen, wenn der Aufwand für das Finden einer Schnittebene zu hoch ist, keine passende Schnittebene gefunden oder das Problem ausgelotet wurde. Kann der Zweig nicht ausgelotet werden, setzt man mit dem Branching fort.
Branch-and-Cut-Algorithmen besitzen in der Praxis oft vergleichsweise gute Konvergenzeigenschaften. Allerdings erhöht das Hinzufügen von Ungleichungen den Rechenaufwand in den Knoten. Außerdem wird Speicherplatz für die Verwaltung des Schnittebenenpools benötigt.

Kapitel 5

Lösung des Modells und numerische Ergebnisse

5.1 Der Solver lp_solve

Für die Lösung des Modells wurde die Software lp_solve ausgewählt. Diese bringt den Vorteil, dass sie kostenlos im Rahmen der GNU lesser general public license ([4]) heruntergeladen und in ein Excel-Makro eingebettet werden kann, sodass für den Dateninput vorhandene Tabellen direkt genutzt werden können. Grundsätzlich ist lp_solve eine Bibliothek von Routinen, die für verschiedene Sprachen wie C, VB, .NET, Excel, Java etc. verwendet oder über AMPL, MATLAB u.a. aufgerufen werden können. Es ist in ANSI C geschrieben und kann sowohl in Windows als auch in Linux kompiliert werden. Das Modell wird per lp- oder mps-File übermittelt oder direkt in einer Programmierumgebung erstellt.
Der Solver löst gemischt-ganzzahlige lineare Programme mithilfe der Branch-and-Bound-Methode. Der Verlauf des Algorithmus kann über verschiedene Modifizierungen wie die Wahl von Verzweigungs-, Knotenwahl- und Abbruchfunktionen beeinflusst werden. Lp_solve stellt weniger Features als z.B. der kommerzielle Solver CPLEX bereit und kann keinerlei nichtlineare Terme behandeln. Ein großer Nachteil ist das Fehlen von Separierungsalgorithmen bzw. der Möglichkeit, als Benutzer Schnittebenen innerhalb eines Branch-and-Bound-Algorithmus zu implementieren.

5.2 Testinstanzen

Um das Verhalten des Modells mit realen und fiktiven Daten zu untersuchen, wurden verschiedene Klassen von Instanzen des Problemes erzeugt, welche in Tabelle 5.1 zusammengefasst sind. Dabei betrachten wir nur die 6 Arbeitsgänge der Injektormontage mit insgesamt 20 Maschinen, in denen die größte Streuung von Rüstzeiten vorliegt. Alle übrigen Arbeitsgänge sind nicht rüstkritisch und erfordern keine Binärvariablen. Die orig-Dateien arbeiten mit Originaldaten und -parametern und unterscheiden sich lediglich in

Instanzenklasse	Variablen	Nebenb.	Variierte Daten
orig	7.613	950	Bedarf
bel3	7.613	950	Typfreigaben: 30%
bel8	7.613	950	Typfreigaben: 80%
ruest1	7.613	ca. 7.040	Rüstzeit: 0 - 8h
ruest2	7.613	ca. 7.040	Rüstzeit: 0 - 2h
zufall	7.613	950	$t_i^m \in [\max\{0, t_i^m - 5\}, t_i^m + 5]$,$0,8 \leq a_i^m \leq 1$

Tabelle 5.1: Übersicht der Instanzenklassen

den Bedarfszahlen. Die Originaldaten besitzen die Eigenschaft, dass viele Parameter nur maschinen- und nicht typabhängig sind. Um den Einfluss konkreter Parameter zu untersuchen, wurden auf Basis der Originaldaten und dem Bedarf aus orig1 folgende künstliche Instanzen mit zufälligen, gleichverteilten Parametern erzeugt: In den bel-Dateien wurden die Typfreigaben zufällig vergeben, in den ruest-Instanzen fallen für jede Typkombination variierende Rüstzeiten an. Schließlich sind in den zufall-Instanzen sowohl Taktzeiten als auch Yields zufällig gewählt, wobei sich die Taktzeiten im Umfeld der Originaldaten bewegen.
Die jeweilige Anzahl der Variablen und Nebenbedingungen ist ebenfalls in Tabelle 5.1 aufgeführt. Das Vorhandensein von Rüstzeiten erhöht natürlich die Anzahl der Nebenbedingungen, welche von der Menge der Nichtnullrüstzeiten abhängt. Die in allen Instanzen hohe Anzahl von Variablen ist darauf zurückzuführen, dass beim Erzeugen der Instanz die Variablen x_i^m, u_i^m und u_{ij}^m auch für nicht erlaubte Typen angelegt wurden. Die Modelle enthalten also noch eine Vielzahl an überflüssigen Variablen und redundanten Indikatorbedingungen der Form

$$x_i^m \leq 0.$$

Diese werden vor der Optimierung durch Presolve-Verfahren von lp_solve eliminiert.

5.3 Numerische Ergebnisse

Es wurden von jeder Instanzenklasse mehrere Datensätze erzeugt und mit dem Branch-and-Bound-Algrithmus von lp_solve gelöst. Die Berechnung wurde nach spätestens zwei Stunden abgebrochen. Die Berechnungen erfolgten auf einem Rechner mit Intel Pentium M 715 Prozessor (1.5 GHz) mit 500 MB RAM und 60 GB Festplatte. Die Ergebnisse sind in Tabelle 5.2 ersichtlich. Zum Vergleich wurde mit CPLEX die Optimallösung der Modelle berechnet. CPLEX wendet mehr Presolve-Techniken an und fügt dem Problem bei Bedarf automatisch Schnittebenen hinzu. Die Tabelle enthält folgende Angaben:

relax: Optimallösung der linearen Relaxierung, Startwert für untere Schranke z_u
best: beste gefundene Lösung mit lp_solve
gap: relativer Abstand der besten Lösung von lp_solve zur aktuellen unteren Schranke (von lp_solve berechnet), $gap = \frac{best - z_u}{z_u}$

time: benötigte Zeit zum Finden der Optimallösung mit lp_solve in Sekunden (Abbruch nach spätestens zwei Stunden: „timeout“)
opt^c: Optimallösung mit CPLEX
time^c: benötigte Zeit zum Finden der Optimallösung mit CPLEX in Sekunden
cuts^c: Anzahl der von CPLEX hinzugefügten Schnittebenen.

Die Zielfunktionswerte wurden gerundet. An sinnvollen Stellen sind außerdem die Durchschnittswerte der jeweiligen Instanzenklasse angegeben.
Bei der Auswertung ist zu bedenken, dass es sich um eine sehr geringe Menge von Instanzen handelt. Alle Schlussfolgerungen im Bezug auf das Verhalten des Modells stellen demzufolge nur den Versuch einer plausiblen Erklärung dar und müssen nicht generell zutreffen.
Als erstes lässt sich feststellen, dass sich die Modelle mit den gewählten Originaldaten im Mittel in ausreichend kurzer Zeit mit lp_solve lösen lassen. Es gibt jedoch auch den Fall orig1, bei dem die Optimallösung in zwei Stunden nicht gefunden werden konnte. In allen Datensätzen wird jedoch in der vorgegebenen Zeit eine Lösung mit einem Gap unter 1% gefunden. Eine Differenz zur Optimallösung von in diesem Fall ca. 400 Stück ist für die Praxis akzeptabel. CPLEX beschleunigt die Lösung durch die Anwendung von Schnittebenen.
Vergleichen wir orig1 mit den bel-Instanzen, stellen wir fest, dass sich die künstlichen Datensätze schneller lösen lassen. Außerdem wird überall ein viel schlechterer Zielfunktionswert erzielt. Bei einer zufälligen Verteilung der Typfreigaben können manche Typen in Arbeitsgängen auf keiner Maschine bearbeitet werden, weshalb dort der komplette Auftrag trotz evtl. freier Kapazitäten unerfüllt bleibt. Selbst bei 80% erlaubter Zuordnungen kann die Verteilung sehr ungünstig sein. Vermutlich wird dadurch aber das Finden der Lösung beschleunigt, da viele Typen schon eindeutig einer Linie oder den Zusatzmaschinen zugeordnet werden können.
Die ruest-Instanzen stellen die schwierigsten Probleme dar. Lp_solve schafft es nicht, in zwei Stunden eine vernünftige Lösung zu finden. CPLEX erreicht durch den effektiven Einsatz von Schnittebenen eine Optimallösung innerhalb einer Stunde. Die ruest1-Modelle mit hohen Rüstzeiten liefern in lp_solve eine Lösung mit sehr hohem Gap, da die Relaxierung eine schlechtere untere Schranke für die Optimallösung darstellt. CPLEX kann diese Probleme schneller lösen als jene mit wenig differierenden Rüstzeiten, da sich hier schneller Zweige vom Lösungsbaum separieren lassen.
Wie zu erwarten verschlechtert sich der Optimalwert mit steigenden Rüstzeiten, da mehr Kapazität für Rüstvorgänge beansprucht wird. Die lineare Relaxierung liefert in allen ruest-Datensätzen den gleichen Wert wie für orig1. Dies untermauert die in Abschnitt 4.4.2 erwähnte Eigenschaft des Modells, dass die Rüstvariablen u_{ij}^m in der relaxierten Lösung häufig den Wert Null annehmen.
Der Vergleich von orig1 mit den zufall-Instanzen zeigt, dass das Finden der Optimallösung bei typunabhängigen Yields und Taktzeiten wie in den Originaldatensätzen im Mittel länger dauert, da viele identische Lösungen existieren und deshalb nur wenige Teile des

Instanz	**relax**	**best**	**gap [%]**	**time [s]**	**opt^c**	**time^c [s]**	**cuts^c**
orig0	0	0	0,0	1,43	0	0,02	0
orig1	227.150	227.782	0,2	timeout	227.396	0,30	28
orig2	353.612	353.612	0,0	11,40	353.612	0,09	16
orig3	3.356.825	3.356.825	0,0	1,76	3.356.825	0,02	0
orig4	1.458.565	1.458.565	0,0	1,60	1.458.565	0,02	0
orig5	3.779.772	3.779.772	0,0	1,56	3.779.772	0,02	0
orig6	3.187.942	3.187.942	0,0	1,14	3.187.942	0,02	0
∅ orig			0,03	1.031,27		0,07	6
bel30	2.307.602	2.307.602	0,0	1,17	2.307.602	0,03	4
bel31	2.917.625	2.917.625	0,0	1,12	2.917.625	0,01	2
bel32	2.354.373	2.354.373	0,0	1,42	2.354.373	0,02	3
∅ bel3			0,0	1,24		0,02	2
bel80	1.562.293	1.562.293	0,0	1,7	1.562.293	0,06	5
bel81	1.732.816	1.732.816	0,0	2,03	1.732.816	0,02	0
bel82	1.561.986	1.561.986	0,0	1,9	1.561.986	0,02	0
∅ bel8			0,0	1,88		0,03	2
ruest10	227.150	457.402	98,0	timeout	368.013	45,84	81
ruest11	227.150	455.235	96,4	timeout	342.521	37,92	60
ruest12	227.150	469.256	106,2	timeout	372.020	47,62	60
ruest13	227.150	472.223	106,7	timeout	365.193	21,86	68
ruest14	227.150	452.057	97,3	timeout	343.207	14,17	90
∅ ruest1			100,9	timeout		33,48	72
ruest20	227.150	316.446	36,4	timeout	270.603	266,87	105
ruest21	227.150	315.137	37,3	timeout	274.895	620,90	99
ruest22	227.150	319.763	38,0	timeout	275.104	110,09	101
ruest23	227.150	318.756	40,1	timeout	275.632	2.807,59	98
ruest24	227.150	302.509	32,1	timeout	267.071	27,00	87
∅ ruest2			36,8	timeout		766,49	98
zufall0	2.616.540	2.625.496	0,3	956,07	2.625.496	0,11	27
zufall1	2.218.733	2.240.375	0,6	timeout	2.229.189	0,14	42
zufall2	1.927.693	1.953.288	0,1	444,18	1.953.288	0,12	34
zufall3	2.321.238	2.327.805	0,1	87,67	2.327.805	0,10	32
zufall4	2.533.779	2.540.072	0,2	633,12	2.540.072	0,13	39
∅ zufall			0,3	1.864,21		0,12	35

Tabelle 5.2: Numerische Ergebnisse

Branch-and-Bound-Baumes ausgelotet werden. Bei weniger symmetrischen Zeiten kann die Optimallösung oft schneller gefunden werden. Der schlechte Zielfunktionswert in den zufall-Instanzen rührt einerseits daher, dass in den realen Daten der Yield im Mittel ca. 96% statt 90 % beträgt. Andererseits wurden die Taktzeiten zufällig aus dem Intervall $[t_i^m - 5,\, t_i^m + 5]$ gewählt. Entstanden dabei negative Werte, wurde die Taktzeit auf Null gesetzt und die Typbelegung verboten. Damit sind in Arbeitsgängen mit Taktzeiten unter 5 Sekunden häufiger Typen nicht freigegeben.
Zusammenfassend ist festzuhalten, dass lp_solve für reale Datensätze ausreichend gute Lösungen in akzeptabler Zeit liefert. Bei Erhöhung der Rüstkomplexität stößt es jedoch im Vergleich zu CPLEX schnell an seine Grenzen.

Kapitel 6

Erweiterungsmöglichkeiten

6.1 Semidefinite Relaxierung

Wir haben für die Bedingung $u_{ij}^m = u_i^m u_j^m$ die lineare Relaxierung durch die Nebenbedingungen (R1)-(R3) durchgeführt. Diese kann durch zusätzliche Nebenbedingungen wie die Dreiecksungleichungen verschärft werden. Allerdings haben wir festgestellt, dass diese Formulierung nicht besonders stark ist. Eine andere Möglichkeit ist die semidefinite Relaxierung, welche von [14] entwickelt wurde. Die folgenden Aussagen und Definitionen sind [7] entlehnt.

Definition:
Eine symmetrische Matrix $Q \in \mathbb{R}^{n \times n}$ ist *positiv semidefinit*, wenn gilt

$$x^T Q x \geq 0 \quad \forall x \in \mathbb{R}^n. \tag{i}$$

Man schreibt in diesem Fall $Q \succeq 0$. Eigenschaft (i) ist äquivalent zu den Bedingungen

$$\text{Alle Eigenwerte von } Q \text{ sind nicht-negativ.} \tag{ii}$$
$$\text{Es existiert } R \in \mathbb{R}^{n \times n}, \text{ so dass } Q = R^T R. \tag{iii}$$

Wir betrachten wieder eine konkrete Maschine mit $q := |P_m|$, $u := [u_i]_{i \in P_m} \in \mathbb{R}^q$ und verzichten auf den Index m. Es sei U die Matrix mit Einträgen u_{ij}, wobei $u_{ii} := u_i$. Für eine zulässige ganzzahlige Lösung gilt dann $U := uu^T$, d.h. U ist positiv semidefinit nach (iii). Diese Bedingung kann noch verschärft werden mit

$$Y := \begin{pmatrix} 1 \\ u \end{pmatrix} \begin{pmatrix} 1 \\ u \end{pmatrix}^T = \begin{pmatrix} 1 & u^T \\ u & U \end{pmatrix}.$$

Diese Matrix ist in einer zulässigen Lösung ebenfalls positiv semidefinit und symmetrisch. Darüber hinaus besitzt sie wegen $u_i u_i = u_i$ die Eigenschaft, dass Diagonale und erste Zeile

identisch sind. Geben wir dieser Zeile den Index 0, erhalten wir die Relaxierung

$$\begin{aligned} &Y_{0i} = Y_{ii} && \forall 1 \leq i \leq q \\ &Y \succeq 0. \end{aligned}$$

Diese Formulierung impliziert automatisch $0 \leq u_i \leq 1$. Die Semidefinite Relaxierung kann mit Innere-Punkte-Methoden gelöst werden und liefert oft eine stärkere Schranke als die LP-Relaxierung.

6.2 Robuste und stochastische Optimierung

Die stochastische und die robuste Optimierung stellen zwei Möglichkeiten dar, mit Unsicherheit von Daten umzugehen. Stochastische Ansätze sind z.B. in [10] zu finden. Die Ausführungen zum Prinizip robuster Optimierung entstammen [3].
In unserer Investitionsplanung stellen häufig mehrere Kunden eine Anfrage, mit welchem Investitionsaufwand eine Volumenerhöhung ihres Produktes im Vergleich zur bisher angeforderten Menge erfüllt werden kann. Nicht jede Zusatzanfrage wird später auch tatsächlich umgesetzt. Man könnte nun nach einer Lösung suchen, die diese verschiedenen Bedarfsszenarien berücksichtigt. In der stochastischen Optimierung minimiert man anhand einer bekannten Verteilung der unsicheren Daten den Erwartungswert der Zielfunktion, so dass die Nebenbedingungen mit hoher Sicherheit erfüllt werden. Voraussetzung hierfür ist die Kenntnis der Wahrscheinlichkeiten für die Umsetzung der einzelnen Kundenanfragen. Die stochastische Betrachtungsweise ist sehr optimistisch, eine Lösung verletzt dabei mit hoher Wahrscheinlichkeit eine Nebenbedingung. In der robusten Optimierung sucht man dagegen nach einer Lösung, die mit Sicherheit eine Menge von möglichen Szenarien erfüllt. Diese Betrachtungsweise kann sehr pessimistisch sein, wenn man die Menge der Fälle nicht einschränkt. Man wird z.B. nicht annehmen, dass alle Kundenanfragen zu Stückzahlerhöhungen zugleich umgesetzt werden.
Für unsere Situation passt eher robuste Optimierung, denn wir kennen keine Wahrscheinlichkeiten für die Umsetzung einer Anfrage. Wir können jedoch eine beschränkte Menge $\mathcal{S}$ von Szenarien definieren. Eine robust zulässige Lösung des Ausgangsproblems ist eine Lösung, die für beliebige Daten aus $\mathcal{S}$ zulässig ist. Betrachten wir das allgemeine Problem (MIP) und als unsichere Daten den Vektor b, dann ergibt sich folgendes robustes Optimierungsproblem:

$$\min\{c^T x + d^T u : Ax + Gu \leq b, x \in \mathbb{R}^n_+, u \in \mathbb{Z}^p_+, \forall b \in \mathcal{S}\}.$$

Diese Modellierung setzt folgende Forderungen um:

1. Alle Entscheidungen sind zu treffen, bevor die unsicheren Daten ihre Werte annehmen.

2. Alle Nebenbedingungen dürfen auf keinen Fall verletzt werden.

3. Die Daten sind unsicher, aber beschränkt innerhalb einer bekannten Menge.

Diese Voraussetzungen können bei Bedarf aufgeweicht werden.

In unserem Problem sind die einzigen sofort zu treffenden Entscheidungen die Investitionsmaßnahmen. Die Produktionsmengen können auch erst dann angepasst werden, wenn der Produktionsbedarf endgültig feststeht. Bedingung 1 muss also nicht für alle Variablen erfüllt sein, die Entscheidungsvariablen x_i^m können für jedes Bedarfsszenario unterschiedlich sein. Wir definieren als Szenarienmenge

$$\mathcal{S} = d_1, \ldots, d_S$$

eine Menge von S Bedarfsvektoren $d_s = [d_{i,s}]_{i=1}^{|P|}$. Wir möchten den Kapazitätsbedarf der Arbeitsgänge für alle Szenarien minimieren, wobei die Zusammensetzung des Zusatzbedarfes je nach Szenario beliebig sein darf, die Gesamtzusatzmenge je Arbeitsgang jedoch szenarienunabhängig optimiert werden soll:

$$\text{min} \qquad f_1(x,u,s) = \sum_{m\in Z} \max_{s\in\mathcal{S}} \sum_{i\in P} x_{i,s}^m$$

subject to

$$\begin{aligned}
\sum_{i\in P_m} t_i^m(1+z_i^m)x_{i,s}^m \;+\sum_{i\in P_m}\sum_{\substack{j\in P_m\\ j>i}} r_{ij}^m u_{ij,s}^m &\le c^m && \forall m\in M\backslash Z, s\in\mathcal{S}\\
x_{i,s}^m - u_{i,s}^m d_i &\le 0 && \forall m\in M\backslash Z, i\in P_m, s\in\mathcal{S}\\
u_{i,s}^m + u_{j,s}^m &\le 1 + u_{ij,s}^m && \forall m\in M\backslash Z, i,j\in P_m, i<j, s\in\mathcal{S}\\
\sum_{m\in M_{A_k^{(i)}}} a_i^m x_{i,s}^m &= \sum_{m\in M_{N_k^{(i)}}} x_{i,s}^m && \forall i\in P, k=1,\ldots,n_i-1, s\in\mathcal{S}\\
\sum_{m\in M_{A_{n_i}^{(i)}}} a_i^m x_{i,s}^m &= d_{i,s} && \forall i\in P, s\in\mathcal{S}\\
x_{i,s}^m \ge 0, u_{ij,s}^m, u_{i,s}^m, u_{j,s}^m &\in \{0,1\}
\end{aligned}$$

Möchte man den Zusatzbedarf zusätzlich typabhängig unter Kontrolle haben, kann man als Zielfunktion auch

$$f_2(x,u,s) := \sum_{\substack{i\in P\\ m\in Z}} \max_{s\in\mathcal{S}} x_{i,s}^m$$

definieren.

Eine einfache Aufweichung der Nichtverletzung von Nebenbedingungen können wir folgendermaßen vornehmen: wir teilen die Menge der Bedarfsszenarien in zwei disjunkte Mengen

$$\mathcal{S}_1 := \{d_s\in\mathcal{S} : \sum_{i\in P} d_{i,s} \le N\} \quad \text{und} \quad \mathcal{S}_2 := \{d_s\in\mathcal{S} : \sum_{i\in P} d_{i,s} > N\},$$

also geringe und hohe Anfragen. Die Schranke N legt einen Normalbereich fest, innerhalb dessen sich die niedrigen (und am ehesten umsetzbaren) Anfragen bewegen. Für die hohen Anfragen aus $\mathcal{S}_2$ erlauben wir, dass die Kapazität der Maschinen erhöht werden darf. Wir erhalten als Kapazitätsbedingung:

$$\sum_{i \in P_m} t_i^m (1 + z_i^m) x_{i,s}^m + \sum_{i \in P_m} \sum_{\substack{j \in P_m \\ j > i}} r_{ij}^m u_{ij,s}^m \leq c_s^m \quad \forall m \in M \backslash Z, s \in \mathcal{S}$$

$$c_s^m = c^m, \quad d_s \in \mathcal{S}_1$$
$$c_s^m = c^m + \tau, \quad d_s \in \mathcal{S}_2,$$

wobei τ die erlaubte Kapazitätserhöhung von z.B. zwei Schichten sei. In [3] wird beschrieben, wie man die relative Verletzung einer Nebenbedingung in Abhängigkeit vom Abstand der Daten zu einem konvexen Normalbereich modellieren kann.

Kapitel 7

Fazit

Mit dieser Arbeit wurde die Grundlage für eine zentrale und automatisierte Kapazitätsplanung unter den geforderten Zielstellungen geschaffen.

In dem entwickelten Modell werden der Produktionsfluss und der Kapazitätsbedarf für ein gegebenes Produktionsprogramm abgebildet, ohne eine Feinterminierung vorzunehmen.

Die Typfreigaben werden im Vergleich zum bisherigen Vorgehen flexibler genutzt.

Typwechselspezifische Rüstzeiten werden mittels empirischer Wechselhäufigkeiten in Abhängigkeit von der Typbelegung der jeweiligen Maschine berücksichtigt.

Im Allgemeinen gehört die bearbeitete Aufgabe zur Klasse der NP-vollständigen Probleme. Zur Lösung des Problems mit realen Daten ist ein einfaches Branch-and-Bound-Verfahren ausreichend.

Die Daten der einzelnen Arbeitsbereiche wurden in standardisierten Tabellen erfasst und können dort von den verantwortlichen Technologen aktualisiert werden.

Mit Hilfe von lp_solve kann das entwickelte mathematische Modell aufgestellt und gelöst werden. So ist eine schnelle Bewertung von Stückzahlszenarien und Identifikation von Engpässen möglich.

Abbildungsverzeichnis

1.1 Prozessfluss Injektorfertigung 2
1.2 Arbeitsstationen einer Vormontagelinie 2

3.1 Typbelegung nach Prioritäten 18
3.2 Günstigere Typbelegung 18
3.3 Ablaufbedingter Maschinenstillstand 22

4.1 Verschiedene Formulierungen für eine Menge ganzzahliger Lösungen 25

Tabellenverzeichnis

2.1 Theoretische und reale Größe des Modells 13

5.1 Übersicht der Instanzenklassen 35
5.2 Numerische Ergebnisse 37

Literaturverzeichnis

[1] K. Aardal and S. van Hoesel. Polyhedral Techniques in Combinatorial Optimization. *Statistica Neerlandica*, 1995.

[2] F. Barahona, S. Bermon, O. Günlük, and S. Hood. Robust Capacity Planning in Semiconductor Manufacturing. *Journal of Global Optimization*, 32(3):385–400, 2005.

[3] A. Ben-Tal and A. Nemirovski. Selected topics in robust convex optimization. *Mathematical Programming, Series B*, 112(1):125–158, 2008.

[4] M. Berkelaar(Entwickler). http://lpsolve.sourceforge.net/5.5.

[5] C.H.Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[6] R. E. Gomory. An Algorithm for Integer Solutions to Linear Programs. In R. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, 1963.

[7] D. J. Grainger and A. N. Letchford. *On Semidefinite Programming Relaxations of Combinatorial Optimization Problems.* Lancaster University, 2007.

[8] M. Grötschel, L.Lovász, and A.Schrijver. The Ellipsoid Method and its Consequences in Combinatorial Optimization. *Combinatorica 1*, pages 169–197, 1981.

[9] M. Grötschel, L.Lovász, and A.Schrijver. Corrigendum to our Paper „The Ellipsoid Method and its Consequences in Combinatorial Optimization“. *Combinatorica 4*, pages 291–295, 1984.

[10] J.R.Birge. Stochastic programming computation and applications. *INFORMS J. Comput.*, 9(2):111–133, 1997.

[11] J. Kallrath. *Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis - mit Fallstudien aus Chemie, Energiewirtschaft, Metallgewerbe, Produktion und Logistik.* Vieweg Verlag, 2002.

[12] H. Meyr. Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operations Research*, 139:277–292, 2002.

[13] G. Nemhauser and L. Wolsey. A recursive procedure to generate all cuts for 0-1-mixed integer programs. *Mathematical Programming, Series A*, pages 379–391, 1990.

[14] N.Z.Shor. Quadratic Optimization Problems. *Soviet Journal of Computer and System Sciences*, 25:1–11, 1987. Originally published in Tekhnicheskaya Kibernetika, No. 1,1987, pp. 128-139.

[15] M. W. Padberg. The Boolean Quadric Polytope: some charakteristics, facets and relatives. *Mathematical Programming*, 45(1):139–172, 1989.

[16] M. W. Padberg. *Linear optimization and extensions.* Springer Verlag, 1995.

[17] M. W. Padberg. Classical Cuts for Mixed-Integer Programming. *Annals of Operations Research 139*, pages 321–352, 2005.

[18] M. W. Padberg and M. P. Rijal. *Location, scheduling, design and integer programming.* Kluwer Academic Publishers, 1996.

[19] G. Piroch. *Standortoptimierung der Ersatzteillager am Beispiel einer Ziegeleifirma unter dem Gesichtspunkt der Kostenminimierung.* TU Chemnitz, 1997.

[20] R.M.Karp. Reducibility among Combinatorial Problems. In R.E.Miller and J.W.Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[21] M. W. P. Savelsbergh. Preprocessing and Probing Techniques for Mixed Integer Programming Problems. *ORSA Journal on Computing*, 6:445–454, 1994.

[22] J. F. Shapiro. *Mathematical programming: structures and algorithms.* John Wiley & Sons, 1979.

[23] A. Staggemeier and A. Clark. A survey of Lot-sizing and Scheduling Models. *23rd Annual Symposium of the Brazilian Operational Research Society(SOBRAPO)*, 2001.

[24] D. Tenfelde-Pohl. *Facilities layout problems: polyhedral structure, multiple objectives and robustness.* Shaker Verlag, 2002.

[25] A. Tschakert. *Langfristige Schulungsplanung für Piloten mittels ganzzahliger Optimierung.* TU Chemnitz, 2004.

[26] L. A. Wolsey. *Integer Programming.* Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, Chichester, 1998.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig und nur unter der Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Chemnitz, 1. Juni 2008 Stefanie Weißbach