Jens Buchholz

Optimale Maschinenbelegung in der mechanischen Fertigung einer Großwerft

Cuvillier Verlag Göttingen

Optimale Maschinenbelegung in der mechanischen Fertigung einer Großwerft

Dem Fachbereich für Mathematik
der Universität Duisburg-Essen
(Campus Duisburg)

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

eingereichte Dissertation von

Jens Buchholz

aus Duisburg

Datum der Einreichung: 15. Dezember 2004

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.
1. Aufl. - Göttingen : Cuvillier, 2005
Zugl.: Duisburg-Essen, Univ., Diss., 2004
ISBN 3-86537-452-2

Von dem Fachbereich Mathematik der Universität Duisburg-Essen (Campus Duisburg) genehmigte Dissertation.
Referent: Prof. Dr. Rüdiger Schultz
Korreferent: Prof. Dr.-Ing. Eike Lehmann
Tag der mündlichen Prüfung: 23. März 2005

Nonnenstieg 8, 37075 Göttingen
Telefon: 0551-54724-0
Telefax: 0551-54724-21
www.cuvillier.de

1. Auflage, 2005
Gedruckt auf säurefreiem Papier

ISBN 3-86537-452-2

Inhaltsverzeichnis

Danksagung

Mein erster und besonderer Dank geht an meinen Betreuer Prof. Dr. Rüdiger Schultz. Er ermöglichte mir die Mitarbeit an interessanten Aufgaben und gab mir stets Impulse für meine Tätigkeit. Seine Unterstützung und sein Engagement verhalfen mir, nicht nur meine mathematischen Kenntnisse zu nutzen und deutlich auszubauen, sondern auch reichhaltige Erfahrungen auf anderen Feldern, zum Beispiel in der Präsentation der Ergebnisse auf Workshops, Tagungen und Konferenzen, zu machen.

Außerdem möchte ich Prof. DR.-Ing. Jürgen Ritterhoff, als ehemaligen Leiter der Forschungs- und Entwicklungsabteilung bei HDW, für seine Unterstützung und sein Engagement danken. Er sorgte stets für eine reibungslose Koordination zwischen dem Institut für Mathematik an der Universität Duisburg-Essen (Campus Duisburg) und des Projektpartners, der Howaldtswerke Deutsche-Werft AG. Mein weiterer Dank auf Seiten des Projektpartners gilt Dirk Hauser, dem Leiter der mechanischen Fertigung. Er erklärte sich bereit das Forschungsprojekt zu unterstützen und brachte bei Diskussionen stets inspirierende Ideen ein. Außerdem danke ich Ingo Pedersen für die prompte Bereitstellung notwendiger Daten und die fachlichen Unterstützung bei Fragen zu diesen Daten.

Den Mitgliedern der Arbeitsgruppe "Diskrete Mathematik und Optimierung" des Institutes für Mathematik an der Universität Duisburg-Essen (Campus Duisburg) danke ich für die anregenden Diskussionen, die zu Weiterentwicklungen des vorliegenden Problems führten. Insbesondere möchte ich dabei Dr. Ralf Gollmer danken. Er hatte stets ein offenes Ohr bei Implemetierungsproblemen und sorgte stets für neue, leistungsfähige Hard- und Software.

Meinen ehemailgen Kommilitonen Tobias Heidenreich und Guido Wolf, sowie meiner Mutter und meiner Freundin Tanja Simon danke ich für das sorgfältige Lesen dieser Arbeit und die Anregungen zu Verbesserungen.

Schließlich gilt mein Dank Prof. Dr.-Ing. Eike Lehmann für seine Bereitschaft als zweiter Referent dieser Arbeit zur Verfügung zu stehen. Des weiteren sorgte er für anregenden Ideen während diverser Vorträge beim Projektpartner.

Duisburg, im Dezember 2004 Jens Buchholz

Kapitel 1

Einleitung

Die mathematische Optimierung bietet ein breites Instrumentarium zur Lösung von Planungsproblemen aus der industriellen Praxis. Die vorliegende Arbeit befasst sich mit der Entwicklung mathematischer Modelle und Lösungsverfahren zur Optimierung der Maschinenbelegung in der mechanischen Fertigung einer Großwerft. Sie entstand im Rahmen des Kooperationsprojektes "Diskrete Optimierung werfttechnischer Fragestellungen" zwischen der Howaldtswerke Deutsche-Werft AG in Kiel und dem Institut für Mathematik an der Universität Duisburg-Essen, Campus Duisburg.

Die Arbeit ist in acht Kapitel unterteilt. Im zweiten Kapitel wird zunächst eine Einordnung in den werftweiten Produktionsprozess gegeben. Da das behandelte Optimierungsproblem in den mathematischen Bereich der Scheduling-Probleme fällt, wird in Kapitel drei ein Überblick über Scheduling-Probleme in der Literatur gegeben. Im vierten Kapitel wird das Anwendungsproblem näher beleuchtet. Insbesondere wird dabei auf die bisherige Vorgehensweise sowie die auftretenden Nebenbedingungen eingegangen. Im fünften Kapitel wird eine Formulierung des Anwendungsproblems als lineares, gemischt-ganzzahliges Optimierungsproblem vorgestellt. Anschließend wird im sechsten Kapitel die Umsetzung des Modells in Cplex sowie in zwei verschiedenen Heuristiken erläutert. Das siebte Kapitel beschäftigt sich mit den Anwendungsvorteilen und zeigt dabei insbesondere auf, dass mit den neuen Algorithmen auch neue Einsatzmöglichkeiten für die Werft entstanden sind. Das neunte Kapitel schließlich behandelt ausgesuchte Rechenergebnisse, welche vor allem darauf abzielen, einige Anwendungsvorteile zu bestätigen.

Kapitel 2

Einordnung in den werfttechnischen Kontext

Der Bau eines Schiffes stellt auch heute noch hohe handwerkliche und technische Ansprüche an eine Werft. Diese ergeben sich vor allem durch die speziellen Eigenschaften, die ein Schiff zu erfüllen hat, so dass fast jedes Schiff ein Unikat darstellt (im Gegensatz zum Automobilbau, in dem Serien hergestellt werden). Als Beispiel für eine solche Werft soll die unseres Projektpartners, die Howaldtswerke Deutsche-Werft AG in Kiel, zu Grunde gelegt werden. Diese Werft ist zum einen auf den Bau ziviler Überwasserfahrzeuge ausgerichtet und zum anderen auch in der Lage, militärische Über- und Unterwasserfahrzeuge zu bauen. Dieses Kapitel soll den Bauablauf eines Schiffes dem Leser näherbringen. Dazu werden in den folgenden drei Abschnitten zunächst der organisatorische Ablauf, dann der Produktionsablauf und schließlich das Qualitätsmanagement vorgestellt. Da in diesen drei Punkten zwischen zivilem und militärischem Schiffbau in weiten Teilen Übereinstimmungen anzutreffen sind, gehen wir von dem Bau eines zivilen Schiffes aus. An den entsprechenden Stellen wird auf die Unterschiede zwischen zivilem und militärischem Schiffbau eingegangen.

2.1 Organisatorischer Ablauf

Zunächst wird ein Vertrag zwischen der Werft und dem Auftraggeber ausgehandelt. Dieser liefert einerseits die Eckdaten (vgl. Abbildung 2.1) für den zeitlichen Ablauf der Produktion, andererseits werden in diesem Vertrag die für den Auftraggeber wichtigen Eigenschaften des Neubaus festgehalten. Zu diesen Eigenschaften zählen unter anderem:

- Hauptabmessungen des Schiffes
- Ladevolumen und Art der Ladung
- Zulieferfirmen für verschiedene Komponenten, wie etwa Motoren

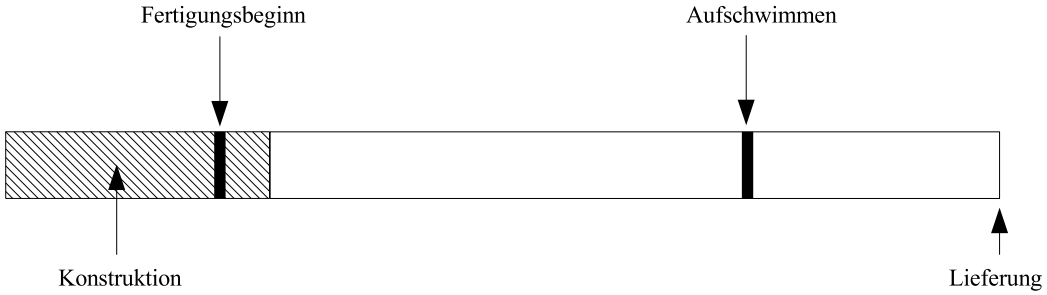

Abbildung 2.1: Zeitlicher Ablauf eines Neubaus mit Eckdaten

Es sind prinzipiell weitere Forderungen seitens des Auftraggebers denkbar, die obige Auflistung stellt jedoch die üblichen Forderungen dar. Diese Bauspezifikationen werden von der Konstruktion umgesetzt und in Bauunterlagen festgehalten. Diese dienen der Arbeitsvorbereitung dazu, Fertigungsaufträge zu erstellen, mit einer groben Terminierung zu versehen und an die jeweiligen Betriebe, in denen die Komponente bearbeitet werden soll, weiterzuleiten. Im Bereich des zivilen Schiffbaus sind in der Regel die folgenden Betriebe betroffen:

1. Objektleitung:
   Für jeden Neubau wird zunächst eine Objektleitung gebildet. Diese ist für die termingerechte Fertigstellung zuständig. Außerdem ist sie die Schnittstelle zwischen den einzelnen Betrieben, die an der Fertigung beteiligt sind und somit oberste Koordinierungsinstanz. Zusätzlich ist sie für den Qualitätsnachweis gegenüber dem Kunden verantwortlich.

2. Stahlschiffbau:
   Im Stahlschiffbau werden die Stahlarbeiten am Schiff durchgeführt. Eine detaillierte Beschreibung dieser Arbeiten erfolgt im Abschnitt über den Produktionsablauf.

3. Rohrfertigung:
   Die Rohrfertigung ist für die Produktion (sofern nicht eingekauft) und Installation der im Schiff benötigten Rohre zuständig. Die Rohre werden zum

Transport von Brennstoffen, Schmiermitteln, Wasser (sowohl Trink- als auch Brauchwasser) und ähnlichem verwendet. Je nach Größe einer Schiffes können so mehrere Kilometer an Rohrleitungen zusammen kommen.

4. Elektroausrüstung:
   Die Elektroausrüstung ist für die elektrische Installation an Bord eines Schiffes verantwortlich. Es muss zum einen die Anbringung von Schalttafeln und zum anderen deren Verkabelung untereinander bzw. mit den entsprechenden Geräten gewährleistet werden. Ähnlich wie in der Rohrfertigung, müssen je nach Größe eines Schiffes mehrere Kilometer an Kabeln verlegt werden.

5. Einzelteilfertigung:
   Die Einzelteilfertigung stellt eine wichtige Ausrüstungskomponente im Produktionsprozess dar. In diesem Bereich werden verschiedene Einzelkomponenten hergestellt, welche für die Ausrüstung eines Schiffes benötigt werden und nicht in einen der obigen Bereiche fallen. Dazu gehören unter anderem:

   - Im Bereich der mechanischen Großteilbearbeitung:
     Abgasventile; Getriebegehäuse; Kühlergehäuse; druckfeste Lukendeckel; Schweißteilbearbeitung von Fundamenten, Schottwänden und ähnlichem; Torpedorohre; Ruderwellen; Antriebsgehäuse; Flansche; Gewinderinge für Lukendeckel; Lagerbuchsen; ...
   - Im Bereich der mechanischen Kleinteilbearbeitung:
     Ventilspindeln; Steuerwellen; Lagergehäuse; Kegelräder; Flansche; Scheiben; Gewindemuttern; Kegelventile; Passbolzen; Kolben; Kolbenstangen; Zylinderlaufbuchsen; Dichtringe für Abgasventile; ...

6. Kranbetrieb:
   Der Kranbetrieb ist für den Transport einzelner Komponenten und Sektionen sowohl innerhalb einer Produktionshalle als auch auf dem Werksgelände verantwortlich. Neben den zwei Hauptkrananlagen mit einer maximalen Hebekraft von 450t bzw. 1000t (vgl. Abbildung 2.2) stehen weitere kleine Kräne sowie Transportfahrzeuge für Schwertransporte zur Verfügung.

7. Materialwirtschaft:
   Die Materialwirtschaft ist für die Bereitstellung und rechtzeitige Auslieferung von angelieferten Bauteilen, Rohmaterialien und Zwischenprodukten zuständig. Dazu stehen einige Transporteinheiten, wie etwa Gabelstapler und Ähnliches zur Verfügung. Für größere bzw. schwerere Elemente erfolgt der Transport in

Absprache mit dem Kranbetrieb. Die Anlieferung der Materialien erfolgt in der Regel zu zentralen Stellen der jeweiligen Betriebe, von wo aus der Betrieb die Materialien weiterverteilt.

Abbildung 2.2: Übersicht über das Werftgelände bei HDW in Kiel

Für die verarbeitenden Betriebe (Nr. 2 - 4) lässt sich die grobe Terminierung in drei Phasen unterteilen:

- Phase 1: Materialbereitstellung
- Phase 2: Fertigung
- Phase 3: Kontrolle

Die Einzelteilfertigung ist zwar auch ein verarbeitender Betrieb, jedoch musste auf Grund des sehr hohen Arbeitsaufkommens in diesem Betrieb eine weitere Phase zwischen Materialbereitstellung und Fertigung eingefügt werden. In dieser Phase soll entschieden werden, ob die entsprechende Arbeit selber durchgeführt werden soll oder an ein Fremdunternehmen vergeben wird. Eine Gegenüberstellung dieser Phasenmodelle ist in Abbildung 2.3 dargestellt. Die zentrale Rolle der Einzelteilfertigung

im Bereich der Neubauten, Umbauten und Reparaturen sowie bei Auslandsprojekten führt zu dem sehr hohen Arbeitsvolumen in diesem Betrieb. Da die Einzelteilfertigung, vor allem der Bereich der mechanischen Fertigung für die zeitintensive Ausrüstung eines Schiffes wichtig ist, ist es in der Vergangenheit immer wieder zu Problemen bei der Terminwahrung während der Ausrüstungsphase gekommen. Aus diesem Grund soll der Arbeitsablauf im Bereich der mechanischen Fertigung mit Hilfe mathematischer Methoden und Algorithmen modelliert und optimiert werden. Dies ist der Gegenstand dieser Arbeit. Die derzeitige Arbeitsweise der mechanischen Fertigung wird in Kapitel 4 näher erläutert.

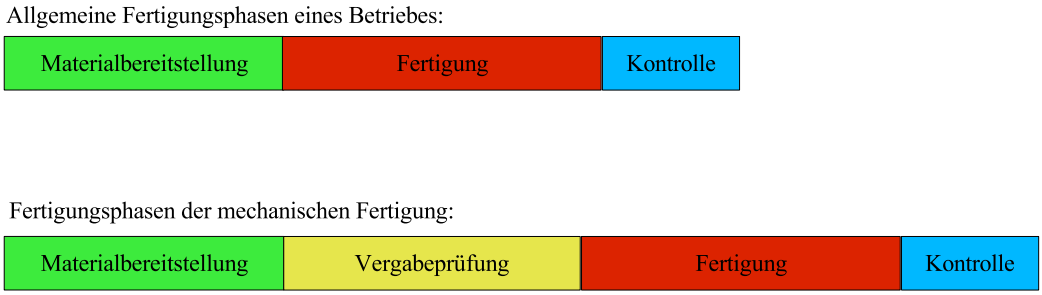

Abbildung 2.3: Gegenüberstellung der allgemeinen Fertigungsphasen eines Betriebes und der Fertigungsphasen der mechanischen Einzelteilfertigung

2.2 Produktionsablauf

Der Bauvorgang eines Schiffes erfolgt nach dem Baukastenprinzip. Dabei werden zunächst aus Einzelteilen Komponenten gefertigt, diese zu Gruppen zusammengefügt und diese wiederum zu noch größeren Objekten. Zum Schluss werden die Teile zum eigentlichen Endprodukt zusammengefügt. Dieses Prinzip wollen wir nun für den Schiffbau am Beispiel des Stahlschiffbaus verdeutlichen:

Die Werft bekommt Stahlplatten in der Regel über das Schienennetz angeliefert. Aus diesen Stahlplatten werden einzelne Elemente herausgeschnitten und zu sogenannten Untergruppen zusammengefügt. Jede Untergruppe besteht in der Regel aus ein bis drei Stahlelementen. Dieser Arbeitsschritt erfolgt in den Hallen 354 und 355 in Abbildung 2.2 je nach Art der Untergruppe. Jede Untergruppe wird markiert, so dass bereits zu diesem frühen Stadium feststeht, in welchem Bereich des Schif-

fes diese Untergruppe eingebaut wird. Im nächsten Bearbeitungsschritt werden die einzelnen Untergruppen in den Hallen 361 und 365 zu sogenannten Gruppen und weiter zu den jeweiligen Sektionen verarbeitet. Einzelne Sektionen werden wiederum zu Blöcken zusammengestellt und verschweißt. Ein Block stellt im Prinzip einen vertikalen Querschnitt des Schiffes dar (vgl. Abbildung 2.4). Bis zu diesem Zeitpunkt erfolgt die Fertigung in entsprechenden Hallen, um den Vorteil einer witterungsunabhängigen Produktionsanlage ausnutzen zu können. Außerdem sind zu diesem Zeitpunkt die Krananlagen noch in der Lage, die Blöcke in das Dock zu transportieren, wo sie schließlich zum eigentlichen Schiff verschweißt werden. Die Ausrüstung der einzelnen Sektionen mit Rohren, Kabeln, etc. erfolgt zum Teil bereits in den jeweiligen Fertigungshallen der Sektionen. Für eine intensive Ausrüstung einzelner Sektionen (zum Beispiel der Brückensektion) ist Halle 366 vorgesehen. Die Ausrüstung eines Schiffes bereits mit der Fertigung der Sektionen zu beginnen, hat den Vorteil, dass gewisse Bereiche eines Schiffes noch leicht zugänglich sind. Somit ist die Materialzufuhr und auch die Ausrüstung selber deutlich einfacher. Außerdem kann in den Hallen eine witterungsunabhängige Ausrüstung erfolgen. Allerdings muss eine sorgfältige Planung der Ausrüstungsarbeiten erfolgen. Dabei muss insbesondere darauf geachtet werden, welche Folgearbeiten an der Sektion noch durchzuführen sind, da diese die bereits eingebaute Ausrüstung unter Umständen beschädigen können. Trotzdem lohnt sich dieser Planungsaufwand, denn die Ausrüstung eines Schiffes im Dock ist um ein Vielfaches aufwändiger und somit kostenintensiver als die Ausrüstung einzelner Sektionen. Neben dem Stahlschiffbau kommt das Baukastenprinzip in allen anderen Fertigungsbereichen der Werft ebenfalls zum Einsatz. Allerdings werden dort in der Regel deutlich weniger Produktionsschritte auf Grund der Struktur der zu fertigenden Komponenten benötigt.

2.3 Qualitätsmanagement

Das Qualitätsmanagement stellt neben der Geheimhaltung den größten Unterschied zwischen militärischem und zivilem Schiffbau dar. Während im zivilen Schiffbau dem Kunden in der Regel nur wenige Qualitätsnachweise vorgelegt werden müssen, müssen im militärischen Schiffbau, insbesondere bei der Fertigung von U-Booten, über eine weitreichendere Zahl von Komponenten detaillierte Qualitätsnachweise erbracht werden. Die Qualitätssicherung erfolgt dabei in drei Stufen:

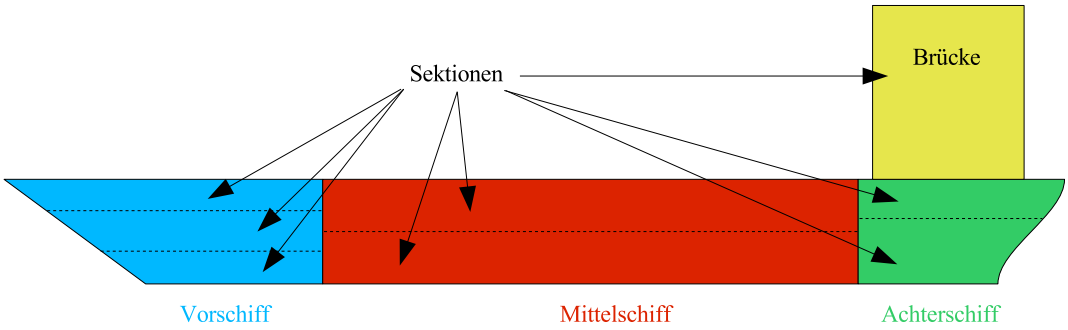

Abbildung 2.4: Beispiel eines Containerschiffes unterteilt in 4 Blöcke mit insgesamt 8 Sektionen

1. Qualitätssicherung Werkstatt (FAT):
   Aus dem Phasenmodell (Abbildung 2.3) ist erkennbar, dass zunächst jeder Fertigungsbetrieb für die Qualität seiner Komponenten verantwortlich ist.

2. Qualitätssicherung Hafen (HAT):
   Beim Einbau einer Komponente in den Schiffsrumpf wird diese umgehend auf ihre Funktionsfähigkeit geprüft. Es kann hier zu Problemen in der Wechselwirkung mit anderen Komponenten bzw. dem Schiffsrumpf kommen, obwohl die Komponenten die Werkstattprüfung mängelfrei überstanden haben.

3. Qualitätssicherung See (SAT):
   Zum Schluss des Bauprojektes erfolgt eine ausgiebige Seeerprobung des Schiffes. Dabei wird zum einen die Funktionsfähigkeit des gesamten Schiffes geprüft, und zum anderen muss nachgewiesen werden, dass das Schiff die Bauspezifikationen des Auftraggebers erfüllt. Probleme können an dieser Stelle neben den Wechselwirkungen mit anderen Komponenten unter anderem durch den Seegang auftreten.

Sollten Probleme bei Punkt 2 oder 3 auftreten, so obliegt es - im Falle der einwandfreien Qualitätsprüfung in der Werkstatt - der Objektleitung in Zusammenarbeit mit den entsprechenden Betrieben, eine Lösung für das Problem zu finden. Mit der Übergabe des Schiffes an den Auftraggeber erhält dieser auch die Protokolle sämtlicher Qualitätsprüfungen.

Kapitel 3

Scheduling in der Literatur

3.1 Definition "Scheduling"

Scheduling-Probleme sind in der Literatur wohl bekannt und schon ausführlich behandelt worden. Dabei wurde eine große Bandbreite von Problemtypen untersucht. Bevor wir uns mit den einzelnen Typen auseinandersetzen wollen, werden wir uns zunächst damit beschäftigen, was unter dem Begriff "Scheduling" zu verstehen ist. Der Begriff ist schon von einigen Autoren definiert worden. Wir wollen die Definition von Pinedo (1995), siehe [**?**, **?**, 8] zugrunde legen:

Definition (Scheduling)
Scheduling beinhaltet die Zuordnung von begrenzten Resourcen (Maschinen) zu Aufgaben (Operationen) über einem Zeithorizont. Es ist ein entscheidungsfindender Prozess, der als Ziel die Optimierung eines oder mehrerer Kriterien hat.

Diese Definition wird in einem erweiterten Zusammenhang benutzt. So werden in der Regel Operationen zu Jobs zusammengefasst. Bestehen diese Jobs nur aus einer Operation, so spricht man von einem *Mono-Operationen-Problem*, sonst von einem *Multi-Operationen-Problem*. Oftmals werden an die Operationen eines Jobs Reihenfolgebedingungen gestellt. In diesem Fall definieren diese Bedingungen das sogenannte *Routing* des jeweiligen Jobs. Die Ressourcenseite lässt sich ebenfalls genauer unterteilen. Ressourcen können erneuerbar oder verbrauchbar sein. Bei den erneuerbaren Ressourcen können wir weiter zwischen Ressourcen unterscheiden, die nur eine Operation zur gleichen Zeit bearbeiten können und solchen Ressourcen, die mehrere Operationen gleichzeitig bearbeiten können. Bei einigen Problemtypen gehen wir davon aus, dass zu Beginn bereits feststeht, welche Ressource für welche Operation benötigt wird. Ist dies hingegen ebenfalls Gegenstand der Optimierung,

so tritt ein zusätzliches Zuordnungsproblem auf, bei dem die Ressource, welche eine Operation bearbeiten soll, spezifiziert werden muss.

3.2 Charakterisierung von Scheduling-Problemen

Scheduling-Probleme werden üblicherweise durch drei charakteristische Eigenschaften voneinander abgegrenzt, vgl. etwa [2]. Das erste Charakteristikum beschreibt die Resourcen bzw. Maschinen. Im nächsten Charakteristikum wird dann auf die Nebenbedingungen eingegangen. Anschließend werden die Optimalitätskriterien klassifiziert. Wir wollen nun in den folgenden Abschnitten auf die einzelnen Charakteristika eingehen.

3.2.1 Ressourcen und Maschinen

Im Allgemeinen werden die Begriffe Ressourcen und Maschinen in der Literatur synonym verwendet. Daher werden wir uns im weiteren Verlauf nur noch mit Maschinen befassen. Die Maschinen eines Scheduling-Problems lassen sich einer Kategorie zuordnen. Diese Kategorien können nochmals zusammengefasst werden, so dass folgende Einteilung üblicherweise vorgenommen wird:

1. Scheduling ohne Zuordnung
Bei dieser Problemklasse werden die Startzeiten für die einzelnen Operationen gesucht. Dabei können verschiedene Typen des Maschinenparks auftreten:

1. 1-Maschinen-Fall:
   Für die Bearbeitung der Operationen steht nur eine einzige Maschine zu Verfügung. Dieser denkbar einfachste Fall stellt ein grundlegendes Modell dar und ermöglicht in einigen Fällen die Lösung von komplexeren Problemen. Die Jobs müssen in diesem Fall notwendigerweise Mono-Operationen sein. Als Anwendungsbeispiel wäre die Abwicklung mehrerer Programme auf einem Computer mit einem Prozessor denkbar.

2. Flow-Shop:
   In diesem Fall stehen für die Bearbeitung mehrere Maschinen zur Verfügung. Charakteristisch ist hier, dass die Bearbeitungsreihenfolge der Jobs auf den Maschinen gleich ist, d.h. das routing der Jobs ist identisch. Weiterhin muss festgelegt werden, ob es möglich ist, dass sich Jobs in der Bearbeitung gegenseitig überholen oder nicht.

3. Job-Shop:
   Beim Job-Shop stehen für die Bearbeitung ebenfalls mehrere Maschinen zur Verfügung. Im Gegensatz zum Flow-Shop ist das Routing der Jobs allerdings nicht identisch, d.h. jeder Job kann ein eigenes Routing haben.

4. Open-Shop:
   Auch beim Open-Shop besteht der Maschinenpark aus mehreren Maschinen. Allerdings ist in diesem Fall kein routing für die Jobs vorgegeben, d.h. die Bearbeitungsreihenfolge der Operationen eines Jobs ist frei wählbar.

5. Gemischter Shop:
   Dieser Fall stellt eine Mischung aus Job-Shop und Open-Shop dar.

2. Scheduling mit Zuordnung und Maschinengruppen

Der Maschinenpark lässt sich in diesem Fall in verschiedene Maschinengruppen unterteilen. Jede Maschine kann dabei eindeutig einer Gruppe zugeordnet werden. Operationen müssen auf einer festgelegten Maschinengruppe bearbeitet werden. Somit entsteht ein zweigeteiltes Problem. Auf der einen Seite muss eine Maschine jeder Operation zugeordnet werden, und auf der anderen Seite muss die zeitliche Reihenfolge der Operationen auf jeder Maschine bestimmt werden. Für jede Gruppe können wir zwischen verschiedenen Konfigurationen differenzieren:

1. Identische Maschinen:
   Die Operationen benötigen auf jeder Maschine die gleiche Bearbeitungszeit.

2. Uniforme Maschinen:
   Die Bearbeitungszeit einer Operation O_i auf der Maschine M_k entspricht hier $p_{i,k} = \frac{q_i}{v_k}$. Dabei stellt q_i z.B. eine Anzahl von Komponenten dar, die für die Operation O_i bearbeitet werden müssen und v_k stellt die Anzahl der Komponenten dar, die mit der Maschine M_k pro Zeiteinheit bearbeitet werden können. Die Bearbeitungszeit ist somit abhängig von der Geschwindigkeit der Maschine.

3. Unabhängige Maschinen:
   Die Bearbeitungszeit der Operation O_i auf Maschine M_k ist hier $p_{i,k}$. Im Gegensatz zu den uniformen Maschinen ist die Bearbeitungszeit nicht abhängig von der Geschwindigkeit der Maschine, sondern von der Maschine selbst. Es muss somit für jede Maschine eine Bearbeitungszeit in den Daten der jeweiligen Operation definiert sein.

3. Allgemeines Scheduling mit Zuordnung

Diese Problemklasse stellt die allgemeinste Art von Scheduling-Problemen dar. Wir gehen dabei davon aus, dass jede Operation auf einer festgelegten Menge von Maschinen bearbeitet werden kann. Weitere Annahmen an diese Menge werden nicht getroffen, d.h. die Maschinen müssen sich nicht generell in verschiedene Gruppen unterteilen lassen können. Auch hier können wir weitere Fälle unterschieden:

1. Alle Jobs sind Mono-Operationen. Wir werden also mit einem Problem konfrontiert, welches in der Literatur als "Multi Purpose Machine Scheduling Problem" behandelt wird (vgl. dazu [2]).

2. Die Jobs müssen in einer vorgegebenen Bearbeitungsreihenfolge abgearbeitet werden. Hierbei ist es zum Teil sehr schwierig, zwischen Flow-Shop und Job-Shop zu unterscheiden. Diese Problemklasse wird in der Literatur als "Shops mit allgemeiner Zuordnung" bezeichnet.

3. Die Jobs müssen kein festgelegtes routing besitzen. Dies wird auch als "Open-Shop mit allgemeiner Zuordnung" bezeichnet.

3.2.2 Nebenbedingungen

Zusätzlich zu den Bedingungen an das Scheduling-Problem, die sich implizit durch die Job- bzw. Maschinenparkcharakteristika ergeben, werden häufig weitere Nebenbedingungen durch das Problem vorgegeben. Wir wollen nun eine Aufstellung der am häufigsten verwendeten Nebenbedingungen angeben:

- Unterbrechnung der Bearbeitung:
  In einigen Fällen ist es möglich, die Bearbeitung einer Operation zu unterbrechen, um sie zu einem späteren Zeitpunkt wieder aufzunehmen. Dabei kann in einigen Fällen auch die Maschine gewechselt werden.

- Aufteilung der Bearbeitung:
  Die Bearbeitungszeit für eine Operation muss nicht in einem Stück durchgeführt werden, sondern kann in kleinere Schritte unterteilt werden. Diese können dann auch zeitgleich auf parallelen Maschinen durchgeführt werden.

- Reihenfolgebedingung an Jobs:
  Manchmal werden durch die Problemstellung Reihenfolgebedingungen an Jobs gestellt. Dies ist nicht zu verwechseln mit dem routing eines Jobs. Dazu betrachten wir folgendes kleine Beispiel: Wir nehmen an, dass wir ein Haus bauen

wollen. Dabei fallen die beiden Jobs "Dach errichten" und "Innenausbau" an. Der Job "Dach errichten" muss natürlich beendet sein, bevor wir mit dem Job "Innenausbau" beginnen können. Daher würde sich hier eine Reihenfolgebedingung ergeben. Der Job "Dachausbau" könnte sich in die Operationen "Dachstuhl errichten" und "Dachdecken" unterteilen. Dabei ist klar, dass erst die Operation "Dachstuhl errichten" abgeschlossen sein muss, bevor wir mit der Operation "Dachdecken" beginnen können. Dies ergibt das routing des Jobs.

- Bündelung von Jobs:
  Jobs können in einigen Anwendungen gebündelt werden, um sie in einem Fertigungsprozess herzustellen.

- No-wait-Probleme:
  Bei den sogenannten No-wait-Problemen darf zwischen der Bearbeitung der Operationen eines Jobs kein Zeitfenster liegen, d.h. ein Job muss ohne Unterbrechung bearbeitet werden.

- Vertauschung von Operationen:
  Bei gewissen Anwendungen spielt das routing eines Jobs keine Rolle, so dass die zugehörigen Operationen vertauscht werden können.

- Identische Fertigstellungstermine:
  Alle Jobs haben den gleichen Fertigstellungtermin. Dies ist z.B. beim Hausbau der Fall, da der Bauunternehmer in der Regel erst Geld bekommt, wenn das Haus fertig ist. Analog können auch identische Starttermine auftreten.

- Identische Bearbeitungszeiten:
  In diesem Fall werden die Bearbeitungszeiten meistens auf eine Zeiteinheit normiert. Analog können auch identische Rüstzeiten durch das Problem gegeben sein.

- Minimaler zeitlicher Abstand zwischen Jobs:
  Bei einigen Anwendungen wird gefordert, dass zwischen der Bearbeitung von zwei Jobs ein vorgegebener zeitlicher Abstand liegen muss.

- Ausgleich von Maschinen:
  Darunter verstehen wir die Forderung, dass alle Maschinen ungefähr zur gleichen Zeit mit der Bearbeitung fertig sein sollen. Dies macht dann Sinn, wenn die Umstellung der Maschinen zeitgleich erfolgen muss, also eine Umstellung einzelner Maschinen nicht möglich ist.

- Blockung von Maschinen:
  Von einer Blockung von Maschinen sprechen wir, wenn zwischen verschiedenen Maschinen nur eine begrenzte Lagerkapazität für Zwischenprodukte gegeben ist. Ist diese Lagerkapazität erschöpft und die weiterverarbeitende Maschine noch beschäftigt, so darf die vorhergehende Maschine keine weiteren Arbeiten durchführen, sie ist also geblockt.

- Rückführung von Jobs:
  Unter der Rückführung von Jobs verstehen wir die unter Umständen mehrmalige Bearbeitung des gleichen Jobs. Dies könnte z.B. durch eine fehlerhafte Produktion im vorangegangenen Schritt notwendig sein.

- Verfügbarkeit von Maschinen:
  Maschinen stehen oftmals nicht ständig zur Verfügung, wenn das Bedienpersonal durch Urlaub, Krankheit, Schichten, etc. nicht zur Verfügung steht.

3.2.3 Optimierungskriterien

Bei Scheduling-Problemen sind eine Vielzahl von Optimierungskriterien vorstellbar. Es ist daher kaum möglich eine Liste aller möglichen Optimierungskriterien anzugeben. Vielmehr liegt es in den Anforderungen des Anwenders, welches Kriterium zum Einsatz kommen soll. Prinzipiell kann jede Nebenbedingung als Optimierungskriterium genutzt werden und umgekehrt. Man muss sich nur im Klaren sein, dass zu starke Nebenbedingungen zur Unlösbarkeit des Problems führen können. Insgesamt lassen sich die Optimierungskriterien in zwei Klassen unterteilen:

1. Minimax-Kriterien:
Bei den Minimax-Kriterien geht es darum, das Maximum einer Funktion zu minimieren. Das wohl häufigste Optimierungskriterium fällt in diese Kategorie, der sogenannte Makespan. Dabei geht es darum, die Zeit zwischen dem Start der ersten Operation und dem Ende der letzen Operation zu minimieren. Weitere Beispiele stellen die Minimierung der maximalen Verweildauer bzw. die Minimierung der maximalen Verspätung dar.

2. Minisum-Kriterien:
Bei den Minisum-Kriterien soll eine Summe von Funktionen minimiert werden. Als Beispiele können die Minimierung der durchschnittlichen, bzw. gesamten Fertigungszeit aller Jobs angegeben werden. Diese Fertigungszeiten können weiterhin nach Jobs

gewichtet werden. Weiterhin stellt die Anzahl der verspäteten Jobs ein Optimierungskriterium dieser Klasse dar.

Auf den ersten Blick wird man feststellen, dass nicht jedes Kriterium in eine dieser Klassen hineinpasst. Jedoch lässt es sich äquivalent umformen, so dass das umgeformte Kriterium dieser Klassifizierung entspricht. Festzuhalten bleibt noch, dass Minisum-Kriterien in der Regel aufwändiger sind als Minimax-Kriterien [8].

3.3 Komplexität und Modelle

3.3.1 Allgemeine Begriffe aus der Komplexitätstheorie

Ein weiterer wichtiger Begriff in der Scheduling-Theorie ist die sogenannte Komplexität. Dies liegt daran, dass sowohl einfache als auch schwierige Scheduling-Probleme bekannt sind. Zunächst müssen wir klären, was mit der Komplexität eines Problems und in diesem Zusammenhang mit einem Problem an sich gemeint ist.

Definition (Problem)
Unter einem Problem versteht man eine allgemeine Fragestellung, bei der mehrere Parameter offen gelassen sind und für die eine Antwort (auch Lösung genannt) gesucht wird. Ein Problem ist definiert, wenn alle Parameter beschrieben sind und zusätzlich angegeben wird, welche Eigenschaften eine Antwort haben soll (vgl. [23]).

Bevor wir uns mit der Komplexität eines Problems beschäftigen, müssen wir uns nun überlegen, wie ein Problem gelöst wird. Im Allgemeinen sagt man, dass ein Algorithmus ein Problem Π löst, wenn er für alle Probleminstanzen $p \in \Pi$ (d.h. alle möglichen Parametereinstellungen eines Problems) eine Antwort findet. Ziel beim Entwurf von Algorithmen ist es, "effiziente" Algorithmen zu entwickeln. Dabei werden Algorithmen hinsichtlich ihrer Speicher- und Zeitkomplexität untersucht. Es ist dabei klar, dass die Laufzeit eines Algorithmus von der Größe der Probleminstanz, d.h. der Eingabedaten, abhängig ist. Für mathematische Eingabestrukturen wird daher ein binäres Kodierungsschema unterstellt. Sind die Eingabestrukturen auf diese Art und Weise einmal festgelegt, kann die Komplexität auf einer geeigneten Rechnerstruktur untersucht werden. Dazu wird in der Regel eine sogenannte Turing-Maschine unterstellt (vgl. z. B. [2], [8] oder [23]). Der Hintergrund dieser Rechnerstrukturen ist in

[24] dargestellt und soll an dieser Stelle nicht weiter vertieft werden. Der Speicherbedarf eines Algorithmus bzgl. einer Probleminstanz $\mathcal{I} \in \Pi$ besteht zum einen aus der Inputlänge $< \mathcal{I} >$ von $\mathcal{I}$, sowie des zusätzlichen Bedarfs an Speicher bzgl. des obigen Kodierungsschema, der während der Ausführung benötigt wird. Die Laufzeit eines Algorithmus wird dagegen über die Anzahl der elementaren Operationen bestimmt, welche bis zur Beendingung des Algorithmus benötigt werden. Zu den elementaren Operationen zählen dabei (vgl. [23]):

- Lesen, Schreiben und Löschen
- Addieren, Subtrahieren, Multiplizieren, Dividieren und Vergleichen

von Zahlen. Zur genauen Berechnung der Laufzeit muss jede elementare Operation mit der entspechenden Kodierunglänge der involvierten Zahlen multipliziert werden. Eine Abschätzung der Laufzeit erfolgt in der Regel dadurch, dass die Anzahl der elementaren Operationen mit der größten Kodierungslänge der im Algorithmus auftretenden Zahlen multipliziert wird. Wir müssen uns sowohl mit der theoretischen als auch mit der praktischen Komplexität von Algorithmen beschäftigen:

- Theoretische Komplexität:
  Gegeben sei ein Problem Π. Die theoretische Komplexität bezeichnet die Schwierigkeit des Problems Π in Abhängigkeit von der Kodierungslänge der entsprechenden Probleminstanz $p \in P$. Es wird dabei außer Acht gelassen, ob die wirklich schwierigen Probleme in der Praxis auftauchen.
- Praktische Komplexität:
  Bei der Untersuchung der praktischen Komplexität werden eine Reihe von Probleminstanzen $p_i \in \Pi$ ($i \in \{1, \ldots, N\}$) hinsichtlich ihres Lösungsaufwandes analysiert. Auf Grund dieser Stichprobe können dann Aussagen über den durchschnittlichen, bzw. minimalen oder maximalen Lösungsaufwand für Probleminstanzen $p \in \Pi$ angegeben werden.

Diese beiden Ansätze bieten jeweils eine Reihe von Vor- Bzw. Nachteilen. Bei der theoretischen Komplexität erhält man eine unwiderlegbare Aussage zur Komplexität der Probleminstanzen von Π bei der praktischen Komplexität dagegen eher vage Aussagen. Diese können aber je nach Umfang und Art der Stichprobe für Anwendungszwecke ausreichen. Man muss sich jedoch stets darüber im Klaren sein, dass es Probleminstanzen geben kann, die eine wesentlich größere Komplexität aufweisen. Ein gutes Beispiel für die Unterscheidung dieser Komplexitätsbegriffe stellt

die Simplexmethode zur Lösung von linearen Optimierungsproblemen dar. Es ist nachgewiesen, dass diese Methode ineffizient sein kann. Allerdings wird sie in der Praxis trotzdem als Standardalgorithmus zur Lösung angewendet. Im weiteren Verlauf werden wir uns nur noch mit der theoretischen Komplexität beschäftigen, da dies Gegenstand vieler aktueller Forschungsvorhaben ist.

Um zwischen "einfachen" und "schwierigen" Problemen unterscheiden zu können, werden wir uns zunächst auf Entscheidungsprobleme zurückziehen. Ein Entscheidungsproblem ist ein Problem, das nur die Antworten JA oder NEIN besitzt. Damit lassen sich zwei Klassen definieren (vgl. [23]):

Definition (Klasse $\mathcal{P}$)
Die Klasse $\mathcal{P}$ besteht aus allen Entscheidungsproblemen, für die ein Algorithmus für eine Turing-Maschine existiert, wobei dieser Algorithmus in Abhängigkeit von der Länge der Kodierung in polynomial vielen Schritten zu einer JA-NEIN-Antwort führt.

In der Klasse $\mathcal{P}$ sind also alle Entscheidungprobleme enthalten, zu denen ein Algorithmus existiert, welcher eine polynomiale Laufzeit aufweist, d.h. effizient ist. Neben dieser Klasse von Problemen wird auch die Klasse $\mathcal{NP}$ betrachtet:

Definition (Klasse $\mathcal{NP}$)
Ein Entscheidungsproblem Π gehört zur Klasse $\mathcal{NP}$, wenn es die folgenden Eigenschaften hat:

(a) Für jedes Problembeispiel $p \in \Pi$, für das die Antwort JA lautet, gibt es mindestens ein Objekt Λ, mit dessen Hilfe die Korrektheit der JA-Antwort überprüft werden kann.

(b) Es gibt einen Algorithmus, der Problembeispiele $p \in \Pi$ und Zusatzobjekte Λ als Input akzeptiert und der in einer Laufzeit, die polynomial in $< p >$ ist, überprüft, ob Λ ein Objekt ist, auf Grund dessen Existenz eine JA-Antwort für p gegeben werden muss.

Diese Definition bedeutet nichts anderes, als dass ein Entscheidungsproblem zur Klasse $\mathcal{NP}$ gehört, wenn eine positive Antwort einer Probleminstanz in polynomialer Zeit überprüft werden kann. Es wird dabei nicht geklärt, wie es zu dieser

positiven Antwort gekommen ist. Das Symbol $\mathcal{NP}$ ist dabei vom Begriff nichtdeterministischer polynomialer Algorithmus abgleitet. Bei diesen Algorithmen ist der erste Schritt nichtdeterministisch ("Raten" des Zusatzobjektes). Anschließend laufen sie wie polynomiale Algorithmen ab. Offensichtlich gilt $\mathcal{P} \subseteq \mathcal{NP}$, da Algorithmen für die Probleme in $\mathcal{P}$ existieren, welche ohne Zusatzobjekt in polynomialer Zeit eine JA- oder NEIN-Antwort liefern. Trotz gewaltiger Forschungsanstrengungen in den vergangenen Jahrzehnten ist die Frage der Gleichheit noch immer ungelöst.

Definition (polynomiale Transformation)
Gegeben seien zwei Entscheidungsprobleme Π und Π' und $p \in \Pi$ eine beliebig Probleminstanz. Eine polynomiale Transformation von Π in Π' ist ein polynomialer Algorithmus, der $p \in \Pi$ in eine Probleminstanz $p' \in \Pi'$ überführt, so dass gilt: Die Antwort auf p ist genau dann JA, wenn die Antwort auf p' JA ist.

Mit Hilfe dieser Definition können wir besonders schwierige Entscheidungsprobleme in der Klasse $\mathcal{NP}$ beschreiben:

Definition ($\mathcal{NP}$-vollständig)
Wir nennen ein Entscheidungsproblem Π $\mathcal{NP}$-vollständig, wenn $\Pi \in \mathcal{NP}$ gilt und falls jedes andere Problem aus $\mathcal{NP}$ polynomial in Π transformiert werden kann.

Mit dieser Definition stellen die $\mathcal{NP}$-vollständigen Probleme die schwierigsten Probleme in der Klasse $\mathcal{NP}$ dar. Existiert nämlich zu einem $\mathcal{NP}$-vollständigen Problem ein Algorithmus mit polynomialer Laufzeit, so existiert zu jedem Problem aus der Klasse $\mathcal{NP}$ ein solcher Algorithmus.

Wir wollen uns nun der Komplexität von allgemeinen Optimierungsproblemen zuwenden:

Ist ein Maximierungsproblem (Minimierungsproblem) P gegeben, so legen wir zusätzlich eine Schranke B fest und fragen uns:

Gibt es für $p \in P$ eine Lösung, deren Wert nicht kleiner (nicht größer) als B ist?

Offensichtlich hängen das ursprüngliche Optimierungsproblem und die obige Fragestellung eng miteinander zusammen. Daher ist folgende Definition sinnvoll:

Definition ($\mathcal{NP}$-schwer)
Wir nennen ein Optimierungsproblem P $\mathcal{NP}$-schwer, wenn das wie oben zugeordnete Entscheidungsproblem polynomial auf ein $\mathcal{NP}$-vollständiges Problem transformiert werden kann.

Mit Hilfe dieser Definition können besonders schwierige Optimierungsprobleme identifiziert werden.

3.3.2 Komplexität bei Scheduling-Problemen: Fallbeispiele

Nachdem im vorigen Abschnitt die wichtigsten Begriffe der Komplexitätstheorie zusammengestellt wurden, wollen wir diese nun anhand einiger Beispiele, welche bei Scheduling-Problemen auftreten, näher erläutern.

Die Klasse $\mathcal{P}$:
Das typische Beispiel für ein Problem der Klasse $\mathcal{P}$ ist das Suchen eines Elementes in einer unsortierten Liste. Dabei gehen wir davon aus, dass das zu suchende Element auch tatsächlich zur Liste gehört. Gesucht ist die Position innerhalb der Liste, an der das Element zu finden ist. Ein Algorithmus könnte dafür wie folgt aussehen (vgl. auch [23]):

Such-Algorithmus:

(A) Geben sei eine Liste L, so wie ein Element $e \in L$. Gesucht ist die Position von e in L.

(B) Setze $i = 1$ (die Listenelemente werden mit 1 beginnend durchnummeriert).

(C) Wenn $L[i] = e$, dann STOP. Das Element e ist an der Position i.

(D) Setze $i = i + 1$ und gehe zu Schritt (C).

Der Aufwand dieses Algorithmus kann leicht abgeschätzt werden, da im Extremfall die gesamte Liste durchsucht werden muss. Damit ist der Aufwand des Algorithmus polynomial abhängig von $< L >$, der Kodierungslänge der Liste. Da dies ein Algorithmus für das gestellte Problem ist, ist der Nachweis erbracht, dass dieses Problem zur Klasse $\mathcal{P}$ gehört.

Ein weiteres Beispiel für ein Problem der Klasse $\mathcal{P}$ ist das folgende Scheduling-Problem:
Gegeben sei ein Mono-Operationen-Problem mit n Jobs, welche auf einer einzigen Maschine durchgeführt werden müssen. Es sei $p_i > 0$ die Bearbeitungszeit von Job i und $w_i \geq 0$ ein Gewichtungsfaktor. Gesucht ist ein Schedule, für den $\sum_{i=1}^{n} w_i C_i$ minimiert wird. Dabei ist C_i die Fertigstellungszeit. Ein entsprechender Algorithmus lautet (vgl. [2]):

Algorithmus $1||\sum w_i C_i$*:*

(A) Nummeriere die Jobs so, dass gilt: $\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq \ldots \geq \frac{w_n}{p_n}$

(B) Setze $C_0 := 0$.

(C) $FOR\ i = 1\ TO\ n\ DO\ C_i := C_{i-1} + p_i$

In Schritt (A) wird für das Sortieren der Jobs ein Aufwand benötigt, der gegen $O(n \log n)$ abgeschätzt werden kann. Der Aufwand für Schritt (C) kann gegen $O(n)$ abgeschätzt werden, so dass der Gesamtaufwand des Algorithmus gegen $O(n \log n)$ abgeschätzt werden kann. Damit besitzt dieser Algorithmus einen polynomialen Aufwand. Für den Nachweis der Optimalität verweise ich auf [2].

Ein $\mathcal{NP}$-schweres Problem:
Um ein Beispiel für ein $\mathcal{NP}$-schweres Scheduling-Problem zu erhalten, betrachten wir den Fall, dass der Maschinenpark aus parallelen Maschinen besteht. Ziel ist es, den Makespan zu minimieren. Dieses Problem wird in der Literatur auch mit $Pm||C_{\max}$ bezeichnet. In der Praxis ist dieses Problem sehr interessant, da es zu einer ausbalancierten Auslastung der Maschinen führt. Es ist gezeigt worden, dass bereits $P2||C_{\max}$, also der Fall mit zwei Maschinen, ein $\mathcal{NP}$-schweres Problem darstellt (siehe [23]). In [2] sind weitere Beispiele für $\mathcal{NP}$-schwere Scheduling-Probleme aufgeführt, sortiert nach der Art des Scheduling-Problems (Job-Shop-Scheduling-, Flow-Shop-Scheduling-, Gemischter-Shop-Scheduling-Problem). Auf den Nachweis, dass das obige Problem ein $\mathcal{NP}$-schweres Problem ist, soll an dieser Stelle verzichtet werden, da dazu tiefere Kenntnisse der Komplexitätstheorie notwendig sind.

3.3.3 Modelle für Scheduling Probleme

Je nach Komplexität eines Scheduling Problems werden unterschiedliche Lösungsansätze bevorzugt. Da eine Vielzahl von Problemen $\mathcal{NP}$-schwer ist, werden in der Regel Heuristiken bevorzugt. Daher sind die Darstellungen von Scheduling Problemen als lineare, gemischt-ganzzahlige Optimierungsprobleme in der Literatur spärlich gesät. In [1] findet sich ein Artikel von Shapiro, welcher ein solches Modell vorstellt.

3.4 Lösungsverfahren

Prinzipiell lassen sich Scheduling Probleme als lineare, gemischt-ganzzahlige Optimierungsprobleme formulieren und mit einem entsprechenden Standardalgorithmus (etwa Branch-&-Bound-Verfahren) lösen. Auf Grund der Komplexität dieser Probleme werden oftmals Heuristiken bevorzugt, da diese deutlich schneller gute Lösungen liefern. Dabei werden unterschiedliche Arten von Heuristiken je nach Problemtyp angewendet. Eine detailierte Beschreibung der hier vorgestellten Algorithmen ist zum Beispiel in [4] angegeben.

Local-Search:
In letzter Zeit haben sich vor allem Algorithmen durchgesetzt, welche nach dem Prinzip der lokalen Suche(auch neighbourhood-search genannt) funktionieren. Diese Algorithmen sind iterative Algorithmen, welche nach dem folgenden Prinzip funktionieren: Ausgehend von einer zulässigen Lösung wird in einer entsprechend definierten Umgebung um diese nach besseren Lösungen gesucht. Existiert keine bessere Lösung in dieser Umgebung, so endet der Algorithmus. Ansonsten wird im nächsten Iterationschritt um diese bessere Lösung herum weitergesucht.

Local-Search-Algorithmus:

(A) Finde eine zulässige Lösung i und initialisiere $i^* = i$. Setze $k = 0$.

(B) Setze $k = k + 1$. Bestimme eine Umgebung $N(i, k)$

(C) Bestimme die beste Lösung $j \in N(i, k) \cup \{i\}$ und setze $i^* = j$.

(D) Wenn keine Abbruchbedingung erfüllt ist, dann gehe zu Schritt (B). Sonst STOP, i^* ist die beste gefundene Lösung.

Das Problem dieser Algorithmen besteht überwiegend darin, wie die Umgebung definiert werden muss, da zum einen die Umgebungen klein genug sein müssen um schnell die beste Lösung darin zu bestimmen, zum anderen müssen die Umgebungen aber auch groß genug sein, um Fortschritte zu erzielen. Als Beispiel für eine Umgebung eines zulässigen Schedules könnte die Menge aller derjenigen Schedules gewählt werden, die durch das Vertauschen von zwei unterschiedlichen Operationen entstehen. Dies ist auch ein Beispiel dafür, dass $i \notin N(i,k)$ gelten kann.

Tabu-Search:
Algorithmen die in die Klasse der Tabu-Search-Algorithmen fallen, beruhen auf der Annahme, dass die Suchrichtung und damit der Suchraum mit entsprechenden Mitteln eingegrenzt werden kann. Dazu werden in der Regel zwei Listen geführt. In der einen Liste werden sogenannte Tabu-Bedingungen aufgestellt. Diese Bedingungen geben an, welche Arten von zulässigen Lösungen nicht hinsichtlich Optimalität untersucht werden sollen. In der anderen Liste werden sogenannte Aspirations-Bedingungen geführt. Diese stellen Bedingungen dar, welche eine gute Lösung erfüllen soll. Der Algorithmus läuft dann wie folgt ab:

Tabu-Search-Algorithmus:

(A) Bestimme ein zulässige Lösung i. Setze $i^* = i$ und $k = 0$ (i^* ist die beste gefundene Lösung und k der Iterationsindex). Definiere weiterhin Tabu- und Aspirations-Bedingungen.

(B) Setze $k = k + 1$ und bestimme eine Teilmenge V aller Lösungen in einer Umgebung $N(i,k)$ um i, so dass jedes Element dieser Umgebung entweder eine Tabu-Bedingung verletzt oder eine Aspirations-Bedingung erfüllt.

(C) Bestimme die beste Lösung j aus der Menge $\{i\} \cup V$ und setze $i^* = j$.

(D) Datiere die Tabu- und Aspirationsbedingungen auf.

(E) Wenn keine Abbruchbedingung erfüllt ist, dann gehe zu Schritt (B).
Sonst STOP, i^* ist die beste gefundene Lösung.

Der Trick bei diesen Algorithmen liegt darin, die Tabu- und Aspirations-Bedingungen geschickt zu wählen. Dadurch wird der Suchraum eingegrenzt, ohne optimale Lösungen zu verbieten.

Genetische-Algorithmen:
Genetische Algorithmen arbeiten nach dem Prinzip der natürlichen Evolution "Der Stärkere überlebt!". Zu Beginn wird eine Anzahl von zulässigen Lösungen (die so genannte Population) benötigt. Diese können durch andere Heuristiken oder zufällig ermittelt werden. Während der Iteration werden dann aus je zwei Lösungen mindestens eine weitere Lösung auf geeignete Art und Weise generiert und gegebenenfalls mutiert. Anschließend wird mit den besten Elementen der Population eine neue Population erzeugt. Auf diese Art und Weise wird die Population in jedem Iterationsschritt niemals verschlechtert, in der Regel sogar verbessert.

Genetischer-Algorithmus:

(A) Initialisiere eine Menge (sogenannte Population) P von zulässigen Lösungen. Setze i^* gleich dem besten Element der Menge P, sowie $k = 0$.

(B) Setze $k = k + 1$. Wähle $i_1, i_2 \in P$, $i_1 \neq i_2$, aus und erzeuge eine neue zulässige Lösung $j(i_1, i_2)$.

(C) Setze $P = (P \cup \{j\}) \setminus \{k\}$, wobei k die schlechteste Lösung in der Menge $P \cup \{j\}$ darstellt.

(D) Datiere i^* auf.

(E) Wenn keine Abbruchbedingung erfüllt ist, dann gehe zu Schritt (B). Sonst STOP, i^* ist die beste gefundene Lösung.

Für unsere Problemstellung wurde ein genetischer Algorithmus an das Problem angepasst (siehe Abschnitt 6.1.2). Dort werden wir näher auf die Vorteile dieser Klasse von Algorithmen hinsichtlich der gegebenen Problemstellung eingehen. Außerdem werden wir dort auf die genaue Implementierung der einzelnen Schritte näher erläutern.

Es gibt noch eine Reihe von weiteren Algorithmenklassen wie etwa Simulated Annealing, welche ebenfalls für Scheduling-Probleme angewendet werden, doch spielen diese eine eher untergeordnete Rolle.

Kapitel 4

Anwendung

Im Gegensatz zu den akademischen Problemstellungen in der Literatur bieten anwendungsorientierte Fälle in der Regel einige Vor- und Nachteile. Die Probleme, die sich bei Anwendungen stellen, bestehen darin, dass sich normalerweise kein Modell, welches in der Literatur eingehend behandelt wurde, direkt auf die Praxis übertragen lässt. Somit müssen oftmals für jeden Anwendungskontext Teile der Modellierung oder des Algorithmus angepasst bzw. vollständig ersetzt werden. Außerdem werden aus der Praxis in der Regel weitere Nebenbedingungen eingebracht, die z.B. auf rechtliche oder betriebsbedingte Forderungen zurückzuführen sind. Diese sind ebenfalls im Anwendungsfall mit einzubeziehen. Allerdings bieten Anwendungen auch einige Vorteile. So werden häufig bereits zulässige Lösungen generiert, die als Startlösung für bekannte Algorithmen dienen können. Weiterhin stellt sich die Datenlage oftmals so dar, dass exakte Algorithmen angewendet werden können, obwohl nachweislich Fälle eintreten könnten, in denen diese Algorithmen einen zu großen Aufwand benötigen würden.

Im nun folgenden Kapitel werden wir auf die Charakteristika unseres Anwendungsfalles eingehen. Daran werden dann sehr schnell die Unterschiede zu den Problemformulierungen in der Literatur deutlich. Außerdem wollen wir uns mit der bisherigen Lösungsmethode sowie der Datenerhebung vertraut machen, um die Vor- und Nachteile der bisherigen Methode herauszustellen.

4.1 Charakterisierung in der Praxis

4.1.1 Jobs und Operationen

Jeder Job stellt ein Werkstück dar, welches eine Reihe von Operationen in einer vordefinierten Reihenfolge zu durchlaufen hat. Mit anderen Worten, das Werkstück muss auf einer Reihe von Maschinen in einer festen Reihenfolge bearbeitet werden. In einem Rohdatensatz sind nicht nur die Bearbeitungsschritte definiert, für die ein Maschineneinsatz notwendig ist, sondern auch Bearbeitungsschritte wie Materialbeschaffung und Kontrolle. Diese Bearbeitungsschritte können vernachlässigt werden, da für diese keine Maschine notwendig ist, so dass jeder Job aus ein bis drei Operationen besteht. An jeden Job sind frühestmögliche Start- und spätestmögliche Fertigstellungstermine gebunden, die sich aus der Materialverfügbarkeit bzw. der weiteren Verwendung des Werkstückes ergeben. Weiterhin sind die Bearbeitungs- und Rüstzeiten für jede Operation hinreichend genau bekannt. Außerdem ist hinterlegt, welcher Maschinentyp für eine Operation notwendig ist.

4.1.2 Maschinenpark

Der Maschinenpark in unserem Anwendungsfall lässt sich zunächst in zwei Bereiche unterteilen. Da wäre zum einen die Kleinmechanik. In diesem Bereich werden relativ viele Werkstücke gefertigt, die jedoch nur eine verhältnismäßig kurze Bearbeitungszeit in Anspruch nehmen. Zum anderen wäre da die Großmechanik. Dort werden zwar nur relativ wenige Werkstücke gefertigt, dafür beanspruchen diese einen relativ langen Bearbeitungszeitraum. Beide Bereiche arbeiten unabhängig voneinander, so dass das Problem in zwei Teilprobleme zerfällt. In beiden Bereichen lassen sich die dort eingesetzten Maschinen in verschiedene Maschinengruppen unterteilen, so dass jede Maschinengruppe aus technisch identischen Maschinen besteht. An die Operationen ist nur die Forderung gestellt, dass sie auf einer bestimmten Maschinengruppe gefertigt werden müssen. Die Wahl der Maschine innerhalb der Maschinengruppe ist frei. Es ist klar, dass jede Maschine nur jeweils ein Werkstück zur gleichen Zeit bearbeiten kann. Die Anzahl der Maschinen variiert von Maschinengruppe zu Maschinengruppe. Außerdem muss ein Schichtplan berücksichtigt werden, in dem jede Maschinengruppe unterschiedliche Schichten fahren kann. Innerhalb einer Maschinengruppe ist der Schichtplan jedoch für jede Maschine gleich.

4.1.3 Zielfunktion

Die Aufgabe besteht darin, einen Maschinenbelegungsplan zu erstellen, der die folgenden Nebenbedingungen erfüllt:

1. Einhaltung der Schichtpläne
2. Einhaltung der Start- und Fertigstellungstermine

Da die zweite Nebenbedingung nicht unbedingt eingehalten werden kann, besteht die Möglichkeit, Jobs an andere Unternehmen zu vergeben. Es können aus betrieblichen Gründen nur vollständige Jobs, die noch nicht bearbeitet wurden, vergeben werden. Eine Vergabe von einzelnen Operationen ist nicht möglich. Da die Fertigstellungstermine nicht unbedingt starr sind, lassen sich diese gegebenenfalls ein wenig verschieben, d.h. eine zulässige Lösung kann Jobs enthalten, die ein wenig verspätet fertiggestellt werden. Diese maximal mögliche Verspätung ist vom Anwender zu Beginn festzulegen und kann zwischen einem und drei Tagen liegen. Somit ist ein Maschinenbelegungsplan gesucht, der die obigen Nebenbedingungen erfüllt und möglichst wenige Vergaben bzw. verspätete Jobs enthält. Dabei muss das Optimum nicht erreicht werden. Vielmehr ist der Praxispartner daran interessiert, eine gute Lösung zu erhalten, d.h. eine Lösung, die mit einem Gütezertifikat versehen werden kann. Als Betrachtungshorizont soll ein Maschinenbelegungsplan für zwei bis vier Wochen erstellt werden. Dies entspricht in etwa einem Datenvolumen von insgesamt 1200 bis 2100 Operationen. Dieses Datenvolumen verteilt sich dann in etwa zu zwei Dritteln auf die Klein- und zu einem Drittel auf die Großmechanik. Ein längerer Zeitraum als vier Wochen ist vorerst nicht geplant, da eine zufrieden stellende Planungssicherheit nach vier Wochen nicht mehr gegeben ist.

4.2 Bisherige Vorgehensweise

4.2.1 Erhebung der Daten

Der Datenaustausch zwischen den einzelnen Bereichen auf der Werft erfolgt über ein SAP-System. Sobald von der Konstruktion eine Zeichnung und eine zugehörige Stückliste angefertigt wurde, wird in diesem SAP-System ein Planauftrag angelegt. Dieser muss einer bestimmten Baugruppe oder einem Arbeitspaket eines Projektes zugeordnet sein. Durch den im Projekt hinterlegten Netzplan wird jedem Planauftrag eine Grobterminierung zugeordnet. Durch die Grobterminierung wird von der

Arbeitsvorbereitung ein entsprechender Arbeitsplan angelegt. Dabei wird geprüft, ob die notwendigen Materialien zum vorgegebenen Zeitpunkt verfügbar sind. Falls sie nicht verfügbar sein sollten und auch nicht rechtzeitig beschafft werden können, so erfolgt eine Umterminierung. Sind diese Vorbereitungen abgeschlossen, so wird der Planauftrag in einen Fertigungsauftrag (=Job) umgewandelt und steht somit der Fertigung zur Verfügung. Der Bestand an Fertigungsaufträgen wird stündlich zwischen dem SAP-System und dem Fertigungsleitstand abgeglichen. Im Fertigungsleitstand erfolgt die Planung der Fertigungstermine. Aufgrund dieser Termine rufen die Materialkoordinatoren das notwendige Material vom Lager ab. Sollten sich bei dem gesamten Vorgang irgendwelche Änderungen ergeben, z.B. Änderungen im Arbeitsplan, Materialverfügbarkeiten etc., so erfolgt die Kommunikation der betroffenen Bereiche stets über das SAP-System.

4.2.2 Erstellung von Maschinenbelegungsplänen

Die Maschinenbelegungspläne werden automatisch im Fertigungsleitstand generiert. Dazu werden noch nicht eingetaktete Jobs aus dem SAP-System gemäß ihrer Terminierung im Arbeitsplan in den bestehenden Belegungsplan integriert, d.h. die nächste freie, ausreichend große Zeitspanne auf der entsprechenden Maschine wird benutzt. Dabei werden die bereits eingetakteten Aufträge nicht verändert. Außerdem wird nicht berücksichtigt, ob Aufträge zu spät fertiggestellt werden. Dies erfolgt in einer manuellen Nachbearbeitung. Dabei sind drei verschiedene Aktionen möglich:

1. Vertauschung von Operationen:
   Es können zwei beliebige Operationen getauscht werden, sofern sich keine von beiden bereits in der Bearbeitung befindet. Dabei müssen diese Operationen weiterhin auf der gleichen Maschinengruppe bearbeitet werden. Außerdem müssen die anderen Rahmenbedingungen, insbesondere die Terminierung, eingehalten werden.

2. Fremdvergabe:
   Ein Job, dessen Bearbeitung noch nicht begonnen wurde, kann an ein anderes Unternehmen vergeben werden. Dies ist zwar mit Kosten verbunden, jedoch können so freie Kapazitäten für andere Jobs gewonnen werden.

3. Tauschen der Maschinengruppe:
   In gewissen Fällen müssen Operationen nicht unbedingt von der vorgegebenen Maschinengruppe bearbeitet werden. Statt dessen können auch ein bis zwei andere Maschinengruppen benutzt werden. Dadurch erhöht sich zwar in der

Regel die Bearbeitungszeit, allerdings bietet sich dadurch der Vorteil, dass freie Kapazitäten auf anderen Maschinengruppen genutzt werden können. Da diese Möglichkeit nur für wenige Operationen in Frage kommt und dieses Vorgehen nur als Notlösung angesehen wird, werden wir diesen Aspekt in der Optimierung nicht aufgreifen. Ein Einbeziehen dieses Sachverhaltes ist aber prinzipiell möglich.

Diese Aktionen können natürlich in beliebiger Reihenfolge und beliebig oft wiederholt werden. Die manuelle Nachbearbeitung erfolgte bisher in der Regel einmal pro Woche bei der Beratung der entsprechenden Mitarbeiter. Dabei konnten auch kurzfristige Änderungen berücksichtigt werden, wie z.B. geplante Ausfälle von Maschinen zwecks Wartungsarbeiten etc..

4.3 Vor- und Nachteile

Der bisherige Lösungsansatz zeichnet sich durch seine hohe Flexibilität aus. D.h., sollte es zu einer unvorhergesehenen Änderung der Rahmenbedingungen kommen, so lassen sich die veränderten Rahmenbedingungen sehr leicht berücksichtigen, da der Maschinenbelegungsplan zum größten Teil manuell erstellt wird. Typischerweise ergeben sich Änderungen der Rahmenbedingungen durch unvorhergesehene Ausfälle von Maschinen oder fehlendes Material. In unserem Anwendungsfall hingegen kommen unvorhergesehene Maschinenausfälle eher selten vor. Auch das Problem der Materialversorgung tritt in unserem Fall selten auf, da mit der Erstellung des Arbeitsplanes die notwendigen Materialien angefordert werden und im Materiallager bereitgestellt werden. Dagegen ist es normal, dass während der Ausrüstungsphase eines Schiffes gewisse Zusatzarbeiten anfallen, die sofort erledigt werden müssen. Auf dieses Problem werden wir in Kapitel 7 näher eingehen. Des Weiteren bietet dieser Lösungsansatz den Vorteil, dass auf die Erfahrung von Mitarbeitern zurückgegriffen werden kann, die sich schon einige Zeit mit der Erstellung der Maschinenbelegungspläne beschäftigt haben. An dieser Stelle kommen wir allerdings auch zu den Nachteilen. Die Erfahrung der Mitarbeiter geht bei einem Mitarbeiterwechsel in der Regel verloren, so dass ein neuer Mitarbeiter erst einen Lernprozess durchlaufen muss, in dem er die nötigen Erfahrungen sammeln kann. Außerdem hat sich in der Vergangenheit des Öfteren herausgestellt, dass die Erstellung eines zulässigen Maschinenbelegungsplanes sehr zeitaufwändig ist. Daher ist es auch nicht möglich, mit der bisherigen Methode einen Maschinenbelegungsplan zu erstellen, der erst in

ein bis zwei Monaten zum Tragen kommt.

Es ist also ein Lösungsansatz gesucht, der schnell eine zulässige Lösung bietet, die nach Möglichkeit mit einem Gütezertifikat versehen ist. Diese Lösung soll dann Grundlage für eine Besprechung der zuständigen Mitarbeiter sein, um einen Maschinenbelegungsplan zu erstellen. Im Optimalfall wären somit nur geringe Änderungen am automatischen Belegungsplan notwendig, die in einem überschaubaren zeitlichen Rahmen erarbeitet werden können. Außerdem ließen sich so mehrere Lösungsmöglichkeiten zu unterschiedlichen Rahmenbedingungen für die Besprechung generieren. Aus diesen können dann Schlussfolgerungen für einen guten Belegungsplan gezogen werden. Außerdem können so weitergehende Entscheidungen, wie z.B. Investitionsentscheidungen, unterstützt werden.

Kapitel 5

Modellierung

In unserem Anwendungskontext treten einige Nicht-Standard-Elemente für Job-Shop-Scheduling Probleme auf. Daher werden wir diese zunächst näher untersuchen, bevor wir mit der eigentlichen Modellierung unseres Problems beginnen.

5.1 Identische Maschinen

Da in unserem Problem Maschinengruppen mit technisch identischen Maschinen auftreten, müssen wir uns Gedanken darüber machen, wie die physische Reihenfolgeplanung auf einer solchen Maschinengruppe zu erfolgen hat. Dazu betrachten wir eine Maschinengruppe, die aus drei Maschinen bestehen soll. Auf diesen Maschinen seien sechs Operationen einzuplanen. Ein möglicher Belegungsplan für diese Maschinengruppe ist in Abbildung 5.1 dargestellt.

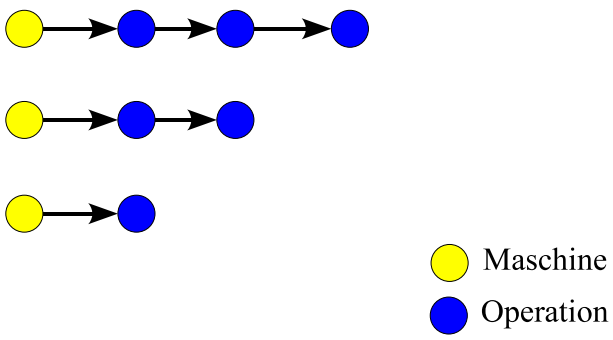

Abbildung 5.1: Belegungsplan einer Maschinengruppe mit drei identischen Maschinen und sechs Operationen

Betrachten wir die Maschinenbelegung genauer, so stellen wir fest, dass sich das Pro-

blem der Maschinenbelegung für eine Maschinengruppe mit identischen Maschinen auf das wohlbekannte Problem der Rundreise in einem Graphen übertragen lässt [1].

Definition Rundreise
Gegeben sei ein Graph $G = (V, E)$ mit einer Knotenmenge V (engl. vertices) und einer Kantenmenge E (engl. edges). Eine Rundreise ist eine Menge $R \subseteq E$, so dass gilt:

R1: Zu jedem Knoten $i \in V$ existiert eine Kante $e_{ji} \in R$, die in den Knoten i hineinführt.

R2: Zu jedem Knoten $j \in V$ existiert eine Kante $eji \in R$, die aus dem Knoten j herausführt.

R3: R enthält keine Teilzyklen, sogenannte Subtouren.

Anmerkung:
Die Bezeichnung e_{ij} für eine Kante impliziert die Richtung der Kante vom Knoten i zu dem Knoten j.

Wir definieren für unser Problem einen Graphen auf folgende Art und Weise: Jede Maschine und jede Operation wird als Knoten eines gerichteten Graphen interpretiert. Weiterhin sei der Graph vollständig, d.h. zwischen je zwei Knoten existiert sowohl eine Vorwärts- als auch eine Rückwärtskante. Gesucht ist in diesem Graphen eine Rundreise. Eine mögliche Lösung unseres Rundreiseproblems für die angegebene Maschinenbelegung ist in Abbildung 5.2 angegeben. Wir können beweisen, dass sich jeder Belegungsplan zu einer entsprechenden Rundreise umformulieren lässt:

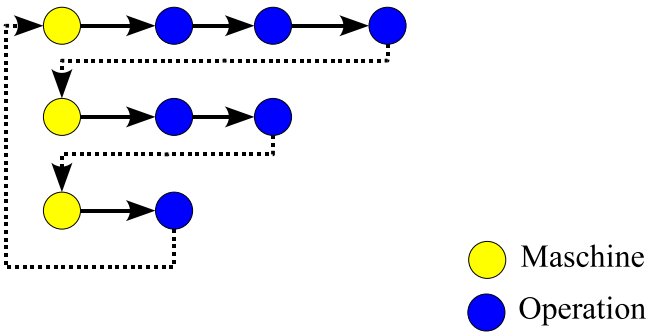

Abbildung 5.2: Darstellung der Maschinenbelegung als Rundreise

Lemma 1
Gegeben sei ein Maschinenbelegungsplan für eine Maschinengruppe mit identischen Maschinen. Dann existiert eine Rundreise auf einem entsprechenden Graphen, so dass die Reihenfolge der Maschinenbelegung der Reihenfolge in der Rundreise entspricht.

Beweis:
Jeder Maschinenbelegungsplan läst sich in folgender Art und Weise darstellen:

$$\begin{array}{lll} M_1 \rightarrow & O_{11} \rightarrow \ldots & \rightarrow O_{1n_1} \\ M_2 \rightarrow & O_{21} \rightarrow \ldots & \rightarrow O_{2n_2} \\ & \vdots & \\ M_l \rightarrow & O_{l1} \rightarrow \ldots & \rightarrow O_{ln_l} \end{array}$$

Wir definieren nun einen Graphen. Dafür generieren wir einen Knoten v_{M_i} für alle $i \in \{1, \ldots, l\}$. Diese Knoten entsprechen den Maschinen. Weiterhin generieren wir für jede Operation O_{ij}, $i \in \{1, \ldots, l\}$ und $j \in \{1, \ldots, n_i\}$, einen Knoten $v_{O_{ij}}$. Diese Knoten entsprechen den Operationen. Wir bilden nun in diesem Graphen einen Weg auf folgende Art und Weise:

1. Aus dem Maschinenbelegungsplan übernehmen wir die Kanten $v_{O_{ij}}v_{O_{ij+1}}$ für alle $i \in \{1, \ldots, l\}$ und alle $j \in \{1, \ldots, n_i - 1\}$. Weiterhin übernehmen wir die Kanten $v_{M_i}v_{O_{i1}}$.

2. Zusätzlich fügen wir die Kanten $v_{o_i n_i}v_{M_{i+1}}$ für alle $i \in \{1, \ldots l - 1\}$ zu dem Weg hinzu.

3. Schließlich fügen wir noch die Kanten $v_{O_{ln_l}}v_{M_1}$ zu dem Weg hinzu.

4. Ist für ein $j \in \{1, \ldots, l\}$ $n_j = 0$, d.h. auf der Maschine j wird keine Operation durchgeführt, so nehmen wir die Kante $v_{M_j M_{j+1}}$ in die Rundreise auf.

Offensichtlich definiert der so erstellte Weg eine Rundreise auf dem Graphen und der Beweis ist erbracht. □

Umgekehrt lässt sich auch folgendes Lemma beweisen:

Lemma 2a
Gegeben sei ein Graph, dessen Knoten aus den Maschinen einer Maschinengruppe und den darauf durchzuführenden Operationen gebildet wurde. Ist dann für diesen Graphen eine Rundreise gegeben, so existiert eine entsprechende Maschinenbelegung.

Beweis:
Wir beginnen die Rundreise mit einem Knoten, der zu einer Maschine gehört. Dann lässt sich die Reihenfolge der Knoten wie folgt darstellen:

$$M_1 O_{11} \dots O_{1n_1} M_2 O_{21} \dots O_{2n_2} \dots M_l O_{l1} \dots O_{ln_l} M_1 \dots$$

Wir brechen diesen Kreislauf auf, indem wir jede Kante weglassen, die in einen Knoten führt, welcher einer Maschine entspricht. Dadurch erhalten wir folgende Sequenzen:

$$\begin{array}{c} M_1 O_{11} \dots O_{1n_1} \\ M_2 O_{21} \dots O_{2n_2} \\ \vdots \\ M_l O_{l1} \dots O_{ln_l} \end{array}$$

Dabei können gewisse Abfolgen $O_{i1} \dots O_{in_i}$ leer sein. Dies bedeutet, dass auf der Maschine M_i keine Operation durchgeführt wird. Jede Sequenz lässt sich nun auf eine physische Reihenfolge von Operationen auf der entsprechenden Maschine abbilden.□

Somit ist erwiesen, dass sich die physische Reihenfolge der Operationen auf identischen Maschinen auf eine Rundreise in einem entsprechenden Graphen abbilden lässt und umgekehrt.

Bei Rundreiseproblemen ergeben sich im Allgemeinen Schwierigkeiten aus Subtouren. Diese Schwierigkeiten ergeben sich aus der Anzahl an Nebenbedingungen, die notwendig sind, um eine Rundreise mathematisch zu beschreiben:

Forderung:	Beschreibung	Anzahl Nebenbedingungen
$R1$	Jeder Knoten hat genau einen Vorgänger	card (V)
$R2$	Jeder Knoten hat genau einen Nachfolger	card (V)
$R3$	Keine Subtouren	$\sum_{i=2}^{\text{card }(V)} \binom{\text{card }(V)!}{i!}$

Zur Veranschaulichung zeigt die folgende Tabelle, wie sich die Anzahl der Nebenbedingungen zur Formulierung der entsprechenden Eingenschaften (R1 - R3) in

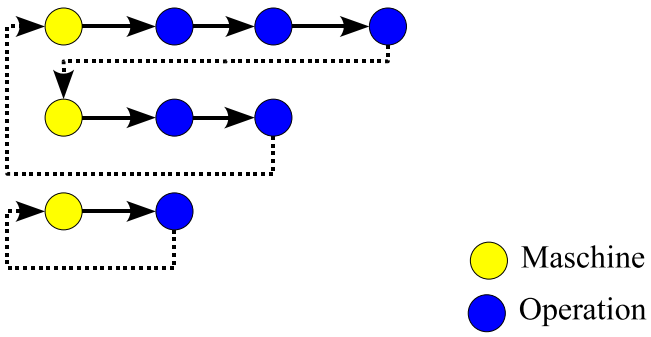

Abbildung 5.3: "Rundreise" mit einer Subtour

Abhängigkeit von der Anzahl der Knoten entwickelt:

card(V)	$R1$	$R2$	$R3$
2	2	2	0
3	3	3	3
4	4	4	10
5	5	5	25
6	6	6	56
7	7	7	119
8	8	8	246
9	9	9	501
10	10	10	1012
11	11	11	2035
12	12	12	4082
13	13	13	8177
14	14	14	16368
15	15	15	32751
16	16	16	65518
17	17	17	131053
18	18	18	262124
19	19	19	524267
20	20	20	1048554

Betrachten wir nun die zu Anfang angegebene Maschinenbelegung, so erfüllt auch die "Rundreise" aus Abbildung 5.3 die Forderung, dass jeder Knoten genau einmal betreten und auch genau einmal verlassen wird. Offensichtlich lässt sich also auch aus einer solchen "Rundreise" eine Maschinenbelegung gewinnen. Daher können wir Lemma 2a umformulieren und erhalten:

Lemma 2b
Gegeben sei ein Graph, dessen Knoten aus den Maschinen einer Maschinengruppe und den darauf durchzuführenden Operationen gebildet wurde. Weiterhin sei eine Lösung des Rundreiseproblems auf diesem Graphen gegeben, wobei die Lösung Subtouren enthalten darf. Enthält dann jede Subtour mindestens einen Knoten, welcher einer Machine entspricht, so existiert auch zu dieser Lösung des Rundreiseproblems eine entsprechende Maschinenbelegung.

Beweis:
Der Beweis erfolgt analog zu dem von Lemma 2a, wobei wir jede Subtour als eigenständige Rundreise auf einem Teilgraphen betrachten. □

Wir behandeln nun den Fall, dass wir eine Subtour erhalten, die keine Maschine enthält. Dazu lässt sich folgendes Lemma festhalten:

Lemma 3
Für das Problem der Maschinenbelegung in einer Maschinengruppe mit identischen Maschinen kann das Teilproblem der Rundreiseplanung nur aus Subtouren bestehen, welche mindestens eine Maschine enthalten.

Beweis:
Angenommen, wir haben eine Subtour, welche nur aus Knoten besteht, die zu Operationen gehören. Dann lässt sich diese wie folgt darstellen:

$$(*) \quad O_1, O_2, \ldots, O_n, O_1 \ldots$$

Aus dem Gesamtproblem wissen wir weiterhin, dass Nebenbedingungen existieren, welche besagen, dass eine neue Operation erst beginnen darf, wenn die vorherige Operation gestartet und bearbeitet wurde. Dies bedeutet für den Fall $(*)$:

$$(**) \quad \begin{array}{c} t_1 + b_1 \leq t_2 \\ t_2 + b_2 \leq t_3 \\ \vdots \\ t_n + b_n \leq t_1 \end{array}$$

Dabei sei t_i die Startzeit von Operation i und b_i ihre zugehörige Bearbeitungszeit. Addieren wir die Nebenbedingungen aus $(**)$ auf, so erhalten wir als weitere zulässi-

ge Ungleichung:

$$\sum_{i=1}^{n} t_i + \sum_{i=1}^{n} b_i \leq \sum_{i=1}^{n} t_i$$
$$\Leftrightarrow \quad \sum_{i=1}^{n} b_i \leq 0$$

Da für unseren Anwendungsfall alle Bearbeitungszeiten größer als Null sind, stellt die letzte Zeile einen Widerspruch dar und der Beweis ist erbracht. □

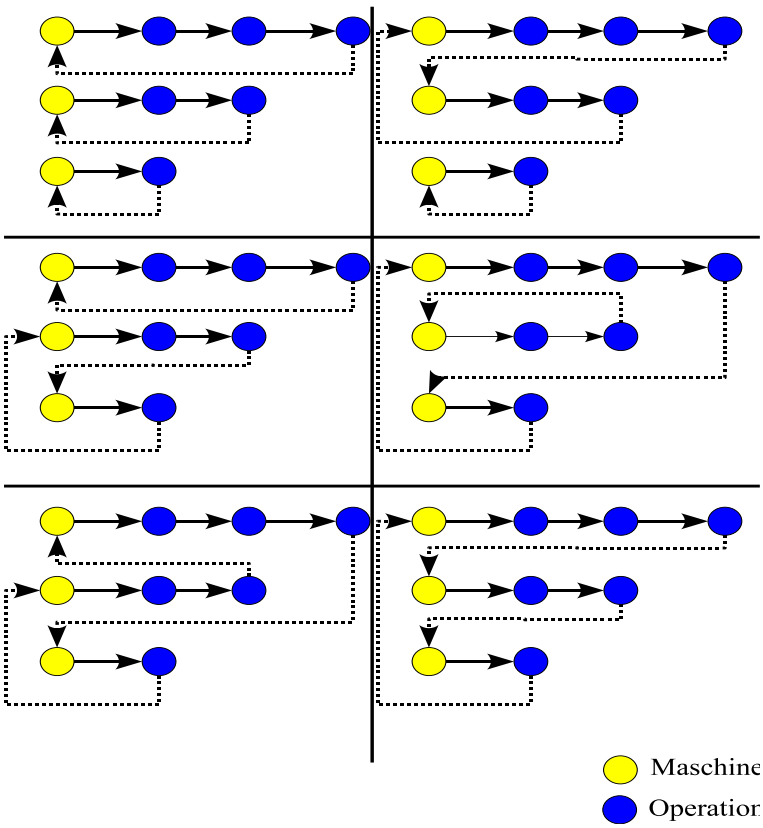

Abbildung 5.4: Rundreisen mit Subtouren für die Maschinenbelegung aus der Abbildung 5.1.

Damit ist gezeigt, dass sich die physische Reihenfolge der Operationen auf identischen Maschinen über eine Rundreise formulieren lässt. Weiterhin wissen wir, dass Subtourelimination, welche beim Rundreiseproblem sehr aufwändig ist, für unseren Fall keine Rolle spielt. Allerdings wissen wir auch, dass mehrere Rundreisen der gleichen Maschinenbelegung entsprechen können. Betrachten wir den Maschinenbelegungsplan aus Abbildung 5.1, so kann dieser durch insgesamt sechs Lösungen des Rundreiseproblems mit möglichen Subtouren gelöst werden. Die Lösungen sind in Abbildung 5.4 aufgelistet. Diese Art der Symmetrie ist für einen Algorithmus sehr schlecht. Unter Umständen muss in unserem Fall eine Vielzahl von Rundreisen

untersucht werden, bevor wir feststellen, dass alle die gleiche Maschinenbelegung beschreiben. Aus diesem Grund werden wir uns nun genauer mit der Vorgänger-Nachfolger-Beziehung für den Fall der Maschinenbelegung beschäftigen:

Offensichtlich ist, dass jede Operation genau einen Vorgänger haben muss. Dieser Vorgänger kann entweder eine andere Operation oder eine Maschine sein. Des Weiteren ist klar, dass jede Operation höchstens einen Nachfolger haben darf, wobei für diesen gefordert werden kann, dass es sich um eine andere Operation und nicht um eine Maschine handeln muss. Für Maschinen lassen sich diese Forderungen dagegen nicht übertragen. Dort müssen wir fordern, dass keine Maschine einen Vorgänger haben darf. Weiterhin fordern wir, dass jede Maschine höchstens einen Nachfolger besitzen darf, wobei dieser unbedingt eine Operation sein muss. Durch diese Charakterisierung der Vorgänger-Nachfolger-Beziehung werden Unterschiede zwischen Knoten gemacht, die einer Maschine entsprechen und solchen Knoten, die einer Operation entsprechen. Dies ist aufwändiger, hat dafür aber zur Folge, dass aus dem Rundreiseproblem ein Routenplanungsproblem wird, bei dem, ausgehend von jedem Maschinenknoten, ein Weg zu suchen ist, so dass jeder Knoten, der einer Operation entspricht, auf genau einem dieser Wege liegt. Die Lösung besteht also aus genau so vielen Wegen wie Maschinen in der jeweiligen Maschinengruppe existieren. Diese Lösung ist eindeutig, abgesehen von der Möglichkeit, Routen zwischen Maschinen zu tauschen. Im Falle mehrerer Maschinengruppen modellieren wir die physische Reihenfolge über genau so viele Graphen, wie Maschinengruppen vorhanden sind. Die Modellierung der physischen Reihenfolge erfolgt dabei unabhängig voneinander. Die optimale Lösung kann jedoch nicht unabhängig voneinander ermittelt werden, sofern das Gesamtproblem kein Mono-Operationen-Problem ist.

5.2 Rüstzeiten

Da in unserem Anwendungsfall im Bearbeitungsvorgang Rüstzeiten vorkommen, müssen wir zunächst klären, welche Art von Umrüstung der Maschinen gemeint ist. Prinzipiell lassen sich nämlich zwei Arten von Rüstvorgängen betrachten:

1. Die Umrüstung einer Maschine ist nicht abhängig von der Art des zu fertigenden Bauteils, d.h. die Umrüstung kann bereits erfolgen, während das Bauteil noch von einer anderen Maschine bearbeitet wird. Diese Art von Umrüstungen trifft man sehr häufig in der chemischen Industrie an, wenn es um die Reini-

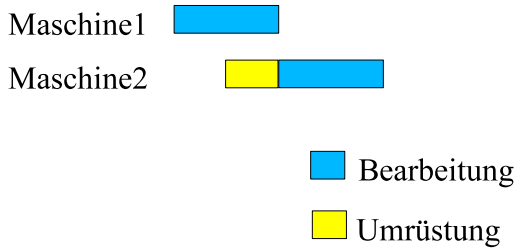

Abbildung 5.5: Bauteilunabhängige Umrüstung, wie sie oftmals in der chemischen Industrie zu finden ist.

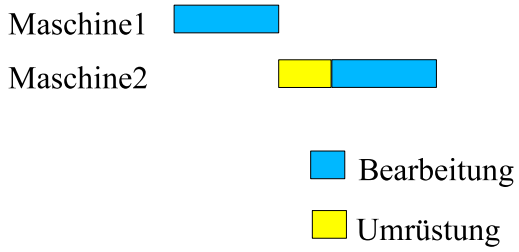

Abbildung 5.6: Bauteilabhängige Umrüstung, wie sie in unserem Anwendungskontext auftritt.

gung von Maschinen bei einem Produktionswechsel geht. Die Abbildung 5.5 stellt diesen Zusammenhang graphisch dar.

2. Die Umrüstung der Maschine ist abhängig von der Art des zu fertigenden Bauteils, d.h. die Umrüstung kann erst begonnen werden, wenn alle vorherigen Arbeiten an dem Bauteil abgeschlossen sind. Diese Art der Umrüstung trifft man vor allem dann an, wenn zur Bearbeitung des Bauteils spezielle Vorrichtungen an der Maschine angebracht werden müssen, wie z. B. Halterungen, in die das Bauteil eingepasst werden muss. Zur Veranschaulichung dient die Abbildung 5.6.

Für unseren Fall müssen wir ein Modell entwickeln, welches den zweiten Fall abbildet. Dazu werden wir die Rüstzeit als weitere Bearbeitungszeit auffassen. Sobald die Bearbeitung einer Operation auf einer Maschine möglich ist, werden wir prüfen, ob zunächst eine Umrüstung zu erfolgen hat. Anschließend kann dann die eigentliche

Bearbeitung erfolgen. Wenn wir nun weiterhin fordern, dass die Bearbeitung einer Operation erst starten kann, wenn die vorherige Operation beendet wurde, erhalten wir ein Modell, welches die gewünschte Umrüstung enthält.

5.3 Schichtpläne

Ein weiteres Kernelement des Modells stellen Schichtpläne dar. Um diese zu berücksichtigen, werden wir erst einige Annahmen treffen, bevor zwei verschiedene Ansätze von Schichtplänen betrachtet werden.

5.3.1 Annahmen

Für den zugrunde liegenden Schichtplan gehen wir von folgenden drei Annahmen aus:

1. Die Arbeitszeit ist für alle Maschinen einer Maschinengruppe gleich, d.h. innerhalb einer Maschinengruppe ist die verfügbare Arbeitszeit jeder Maschine pro Tag gleich.

2. Weiterhin gehen wir davon aus, dass der Arbeitsbeginn für alle Maschinen im gesamten Maschinenpark an jedem Tag gleich ist.

3. Zusätzlich fordern wir, dass die Schichten innerhalb eines Tages zusammenhängend sind, d.h., betrachten wir den Schichtplan nur für einen Tag, so darf im Schichtplan keine Unterbrechnung auftreten.

Die erste Forderung stellt eine Vereinfachung des Modells dar. Zwar kann man prinzipiell für jede Maschine einen eigenen Schichtplan hinterlegen und diesen auch im Modell berücksichtigen, allerdings müssen dazu weitere ganzzahlige Variablen im Modell eingeführt werden. Dies beeinflusst die Rechenzeit negativ. Da weiterhin in Gesprächen mit dem Projektpartner festgestellt wurde, dass normalerweise alle Maschinen einer Gruppe dem gleichen Schichtplan unterworfen sind, können wir davon ausgehen, dass diese Forderung stets erfüllt ist. Die zweite Forderung erleichtert die Modellierung der Schichtpläne. Ist ein Job gegeben, welcher aus mindestens zwei Operationen besteht, die auf unterschiedlichen Maschinengruppen bearbeitet werden müssen, so kann es beim Wechsel zwischen zwei Operationen zu zusätzlichen

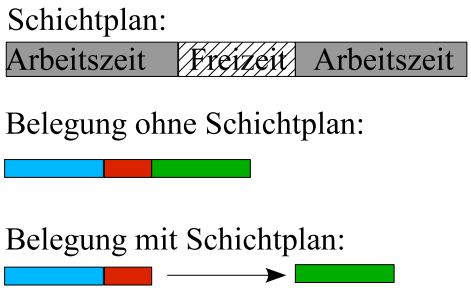

Abbildung 5.7: Belegung einer Maschine bei strikter Schichteinhaltung

Wartezeiten kommen, wenn die betreffenden Maschinen einen unterschiedlichen Arbeitsbeginn haben. Zur weiteren Vereinfachung können wir den zeitlichen Arbeitsbeginn jeder Maschine auf Mitternacht legen, da dies zum tatsächlichen Arbeitsbeginn nur eine Verschiebung um eine konstante Größe darstellt. Die letzte Forderung dient ebenfalls zur vereinfachten Modellierung der Schichtpläne, denn auf diese Weise können wir die Schichten an einem Tag zu einer Schicht zusammenfassen. Ein Wechsel zwischen den Schichten eines Tages muss nicht explizit modelliert werden. Dies ist zwar möglich, allerdings nur unter Zuhilfenahme einer Vielzahl von ganzzahligen Variablen, die sich negativ auf die Rechenzeit auswirken.

5.3.2 Modellierung mit strikter Schichteinhaltung

Schichtpläne können berücksichtigt werden, indem gefordert wird, dass die Bearbeitung mit dem Ende einer Schicht abgeschlossen sein muss, sofern kein nahtloser Übergang zur nächsten Schicht besteht. Dies treffen wir insbesondere dann an, wenn die Bearbeitung nicht unterbrochen werden kann. Würde die Bearbeitung einer Operation vielleicht nur eine kurze Zeitspanne über das Schichtende hinausgehen, so müsste die Bearbeitung dieser Operation auf den Beginn der nächsten Schicht verschoben werden (vgl. Abbildung 5.7). Der so gewonnene Freiraum in der vorherigen Schicht kann zwar unter Umständen für andere Fertigungsvorgänge genutzt werden, im Extremfall werden die Maschinen jedoch nicht bis zum Ende einer Schicht genutzt.

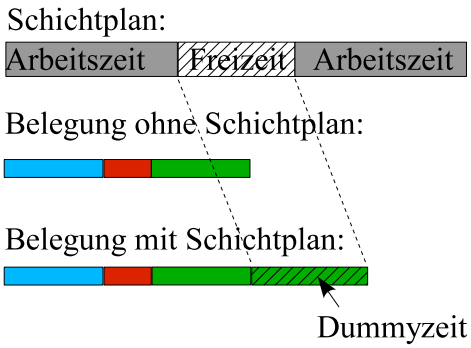

Abbildung 5.8: Belegung einer Maschine bei schichtübergreifender Fertigung

5.3.3 Schichtübergreifende Modellierung

Da in unserem Anwendungsfall Fertigungsvorgänge auftreten, die über mehre Schichten erfolgen, ist ein Modell zu erstellen, welches die Schichtpläne so abbildet, dass die Bearbeitung am Ende einer Schicht unterbrochen und mit dem Beginn der nächsten Schicht fortgesetzt wird. Dazu werden wir eine sogenannte Dummyzeit einführen, die genau der Zeitspanne zwischen den beiden Schichten entspricht. Erstreckt sich die Bearbeitung einer Operation über eine Schicht hinaus, so addieren wir die Dummyzeit zur Bearbeitungszeit hinzu. Dadurch lässt sich abbilden, dass die Bearbeitung mit dem Ende der ersten Schicht unterbrochen und mit dem Beginn der nächsten Schicht fortgesetzt wird (vgl. Abbildung 5.8).

5.4 Modelle

Wir wollen uns nun der expliziten Modellierung des Job-Shop-Scheduling-Problems annehmen. Dazu werden wir im ersten Abschnitt unseren Anwendungsfall an die Standardzielfunktion anpassen. Im zweiten Abschnitt wird dann der Anwendungsfall mit der für den Industriepartner relevanten Zielfunktion modelliert. Eine Übersicht aller Bezeichnungen, Daten, Variablen, Restriktionen und Modelle kann im Anhang nachgeschlagen werden.

5.4.1 Zielfunktion 'Makespan'

Bevor wir mit der Modellierung beginnen können, müssen wir einige Bezeichnungen einführen. Wir bezeichnen mit J die Menge aller Jobs und mit O die Menge aller Operationen. In der Menge O lassen sich die Operationen so durchnummerieren, dass die Operationen eines Jobs fortlaufend nummeriert sind und das Routing des Jobs durch die Nummerierung ausgedrückt wird, d.h die Operationen des Jobs l seien $O_{n_{l-1}+1}, \dots O_{n_l}$. Weiterhin sei M die Menge aller Maschinen, die sich in G Maschinengruppen unterteilen lässt. Wir verstehen zusätzlich unter der Menge M_g die Menge aller Maschinen und Operationen, welche zur Maschinengruppe $g \in \{1, \dots, G\}$ gehören bzw. von dieser Maschinengruppe bearbeitet werden müssen. Zum Schluss definiere die Menge M_g^O die Menge aller Operationen, welche auf der Maschinengruppe g bearbeitet werden muss.

Nebenbedingungen auf Maschinengruppenebene:
Wir haben gesehen, dass die physische Reihenfolge der Operationen auf den Maschinen einer Maschinengruppe durch modifizierte Rundreisenebenbedingungen ausgedrückt werden kann. Dazu führen wir eine Variable $x_{ij}^g \in \{0, 1\}$ ein. Diese ist 1, wenn der Knoten j direkter Nachfolger von Knoten i ist und sonst 0. Die jeweilige Maschinengruppe wird durch den Index g ausgedrückt. Zur Erinnerung, jeder Knoten stellt entweder eine Operation oder eine Maschine der jeweiligen Maschinengruppe dar. Als Nebenbedingungen wird dann gefordert, dass jede Operation genau einen Vorgänger haben muss. Dies lässt sich wie folgt ausdrücken:

$$(M1-1) \qquad \sum_{j \in M_g \setminus \{i\}} x_{ji}^g = 1 \quad \forall i \in M_g^O, g \in \{1, \dots, G\}$$

Mit dieser Nebenbedingung ist sichergestellt, dass jede Operation genau einen Vorgänger - eine andere Operation oder eine Maschine - hat. Weiterhin haben wir gefordert, dass jede Operation und Maschine höchstens einen Nachfolger haben darf und dieser eine Operation sein muss. Dies ist durch die Nebenbedingung

$$(M1-2) \qquad \sum_{j \in M_g^O \setminus \{i\}} x_{ij}^g \leq 1 \quad \forall i \in M_g, g \in \{1, \dots, G\}$$

gegeben. Da der Lösungsalgorithmus in einem preproccessing automatisch nicht benötigte Variablen eliminiert, brauchen wir nicht explizit zu fordern, dass $x_{ij}^g = 0$ ist für alle $i \in O \cap M_g$ und alle $j \in M \cap M_g$. Neben der physischen Reihenfolge

müssen wir natürlich zusätzlich die zeitliche Reihenfolge auf den Maschinen berücksichtigen. Dies geschieht mit folgender Nebenbedingung:

$$(M1{-}3) \qquad t_i \geq t_j + b_j + \sum_{h \in M_g} r_{hj} \cdot x_{hj}^g + w_j - (1 - x_{ji}^g) \cdot M \quad \forall i, j \in M_g^O, g \in \{1, \ldots, G\}$$

Dabei bezeichnet t_i die Startzeit von Operation i, b_i die Bearbeitungszeit von Operation i und r_{hj} die Rüstzeit zwischen den beiden Operationen h und j. Mit w_j ist die Dummyzeit der Operation j bezeichnet, die sich aus dem Schichtplan ergibt. Wir werden bei den Nebenbedingungen auf Jobebene sehen, welche Bedeutung w_j hat. M ist eine Konstante. Ist Operation i der direkte Nachfolger von Operation j, so fällt der Teil $(1 - x_{ji}^g) \cdot M$ weg, da $x_{ji}^g = 1$ ist. Dann besagt die Nebenbedingung gerade, dass die Startzeit von Operation i größer oder gleich der Summe aus Startzeit von Operation j, der entsprechenden Bearbeitungs- und Dummyzeit sowie einer möglichen Rüstzeit sein muss. Ist dagegen Operation i nicht direkter Nachfolger von Operation j, d.h. $x_{ji}^g = 0$, so wird die Nebenbedingung allgemeingültig, wenn M groß genug gewählt wurde. Die Nebenbedingung reduziert sich dann auf $t_i \geq 0$.

Nebenbedingungen auf Jobebene:

Zunächst einmal wollen wir modellieren, dass die Operationen eines Jobs in einer vorgegebenen Reihenfolge bearbeitet werden müssen. Dabei machen wir uns die Nummerierung der einzelnen Operationen zu Nutze und können die entsprechende Nebenbedingung wie folgt aufstellen:

$$(M1-4) \qquad t_{i+1} \geq t_i + b_i + \sum_{j \in M_{g(i)} \setminus \{i\}} r_{ji} \cdot x_{ji}^{g(i)} + w_i \quad \begin{array}{l} \forall i \in \{n_{l-1}+1, \ldots, n_l - 1\} \\ \forall l \in \{1, \ldots, |J|\} \end{array}$$

Der Ausdruck $g(i)$ bezeichnet keine Funktion, sondern stellt nur die Schreibweise für die zur Operation i zugehörige Maschinengruppe g dar. Ähnliche Schreibweisen werden wir analog für den zugehörigen Job und ähnlichem verwenden. Weiterhin muss gewährleistet sein, dass jeder Job erst nach seinem frühestmöglichen Starttermin zur Bearbeitung freigegeben ist. Bezeichnen wir mit T_l^S diesen Termin, so können wir die Restriktion wie folgt schreiben:

$$(M1-5) \qquad t_{n_{l-1}+1} \geq T_l^S \quad \forall l \in \{1, \ldots, |J|\}$$

Da wir in diesem Modell den Makespan, also die Bearbeitungszeit für alle Operatio-

nen, minimieren wollen, vernachlässigen wir die spätesten Fertigstellungtermine der Jobs. Würden wir sie dennoch einbeziehen, so könnte das Problem unlösbar werden. Um den Schichtplan mit einzubeziehen, müssen wir drei weitere Restriktionen erstellen. Zunächst einmal müssen wir ermitteln, zu welcher Tageszeit jede Operation startet. Dazu führen wir eine ganzzahlige Variable d_i ein, die die Anzahl der Tage in der Startzeit ermittelt:

$$(M1-7) \qquad \tfrac{t_i}{24} \geq d_i \geq \tfrac{t_i}{24} - \tfrac{2399}{2400} \quad \forall i \in O$$

Da die Startzeit in Stunden mit einer Genauigkeit von zwei Nachkommastellen angegeben wird, erhalten wir so die gewünschten Informationen. Die linke Seite liefert uns einen Wert, welcher mindestens so groß ist, wie die Anzahl der vollen Tage in der Startzeit. Da wir von diesem Wert auf der rechten Seite fast einen Tag ($\frac{2399}{2400}$) abziehen, liegt zwischen diesen beiden Ausdrücken genau eine ganze Zahl. Mit Hilfe dieser Information können wir nun feststellen, ob die eine Operation eine Schicht überschreitet oder nicht. Die Variable v_i soll 1 sein, wenn die Operation i eine Schicht überschreitet und 0 sonst. Die entsprechende Nebenbedingung, um dies auszudrücken, lautet dann:

$$(M1-8) \qquad t_i - 24 \cdot d_i + b_i + \sum_{j \in M_{g(i)}} r_{ji} \cdot x_{ji}^{g(i)} - 24 \cdot v_i \leq S_g \quad \forall i \in O$$

Der Ausdruck $t_i - 24 \cdot d_i + b_i + \sum_{j \in M_{g(i)}} r_{ji} \cdot x_{ji}^{g(i)}$ liefert uns die Uhrzeit, wann die Operation fertiggestellt wird. Ist dieser Wert kleiner oder gleich der Schichtlänge, so liegt keine Schichtüberschreitung vor, d.h. die Variable v_i kann den Wert 0 annehmen. Ist die Uhrzeit dagegen größer als die Schichtlänge, so muss die Variable v_i den Wert 1 annehmen, damit die Nebenbedingung gültig bleibt. Ist eine Schichtüberschreitung gegeben, also $v_i = 1$, so müssen wir noch die entsprechende Dummyzeit ermitteln. Diese soll in der Variablen $w_i \geq 0$ durch die Nebenbedingung

$$(M1-9) \qquad w_i = (24 - S_{g(i)}) \cdot v_i \quad \forall i \in O$$

ausgedrückt werden. Wir sehen, dass die Variable w_i auch entfallen kann, wenn sie in allen anderen Nebenbedingungen durch den Term $(24 - S_{g(i)}) \cdot v_i$ ersetzt wird.

Zielfunktion:
Zum Schluss müssen wir noch die Zielfunktion angegeben. Beim Makespan-Kriterium soll die maximale Fertigstellungszeit der Jobs minimiert werden. Diese Fertigstellungszeit wird standardmäßig in der Variablen $C_{\max} \geq 0$ angegeben. Wir können sie mit der folgenden Nebenbedingung ermitteln:

$$(M1-0) \qquad C_{\max} \geq t_{n_l} + b_{n_l} + \sum_{j \in M_{g(n_l)}} r_{jn_l} \cdot x_{jn_l}^{g(n_l)} + w_{n_l} \quad \forall l \in \{1, \ldots, |J|\}$$

Die zugehörige Zielfunktion lautet dann:

$$(M1-Z) \qquad C_{\max} \rightarrow \text{Min !}$$

Auf diese Art und Weise haben wir ein lineares, gemischt-ganzzahliges Optimierungproblem erstellt, welches in der Lage ist, für einen gegebenen Datensatz den optimalen Makespan zu ermitteln.

5.4.2 Zielfunktion 'Vergaben'

Ausgehend von dem Modell zur Minimierung des Makespan können wir nun ein Modell zur Minimierung der Anzahl der vergebenen Jobs entwickeln. Dazu müssen die bereits aufgestellten Nebenbedingungen zum Teil modifiziert bzw. neue Nebenbedingungen aufgestellt werden. Die bereits eingeführten Bezeichnungen werden wir weiterhin verwenden. Ziel ist es, ein Modell zu entwickeln, dessen Lösung stets einen Maschinenbelegungsplan ermittelt, in dem jeder Job im Rahmen seiner frühesten Start- und spätesten Endtermine gefertigt wird. Jeder Job, der nicht rechtzeitig gefertigt werden kann, wird an ein Fremdunternehmen vergeben.

Nebenbedingungen auf Maschinengruppenebene:
Die physische Reihenfolge der Operationen auf den Maschinen haben wir über modifizierte Rundreisebedingungen charakterisiert. Diese bleiben auch für dieses Modell prinzipiell bestehen. Allerdings brauchen wir sie nur für solche Operationen betrachten, bei denen der zugehörige Job nicht vergeben wurde. Um zu erkennen, ob ein Job vergeben wird, führen wir eine weitere binäre Variable y_l ein. Es sei $y_l = 1$, wenn der Job l durchgeführt wird und sonst gelte $y_l = 0$. Die modifizierten Rundreisebedingungen lassen sich dann wie folgt formulieren:

$$(M2-1) \qquad \sum_{j \in M_g \setminus \{i\}} x_{ji}^g = y_{l(i)} \quad \forall i \in M_g^O, g \in \{1, \ldots, G\}$$

bzw.

$$(M2-2) \qquad \sum_{j \in M_g^O \setminus \{i\}} x_{ij}^g \leq y_{l(i)} \quad \forall i \in M_g, g \in \{1, \ldots, G\}$$

Dabei sei $y_{l(i)} := 1$, falls $i \in M$, also einer Maschine entspricht. Weiterhin können wir die Nebenbedingung, welche die zeitliche Reihenfolge auf den Maschinen beschreibt, übernehmen.

$$(M2-3) \qquad t_i \geq t_j + b_j + \sum_{h \in M_g} r_{hj} \cdot x_{hj}^g + w_j - (1 - x_{ji}^g) \cdot M \quad \forall i, j \in M_g^O, g \in \{1, \ldots, G\}$$

Sollen die Operationen i und j beide durchgeführt werden, so ändert sich an der Nebenbedingung nichts. Andernfalls sind zwei Fälle möglich:

1. Der Job von Operation i wird nicht durchgeführt:
   Dann ist $y_{l(i)} = 0$ und durch die Nebenbedingung $(M2-1)$ folgt, dass $x_{ji} = 0$ ist für alle $j \in M_g$. Dadurch wird die Nebenbedingung allgemeingültig.

2. Der Job von Operation j wird nicht durchgeführt:
   Damit ist $y_{l(j)} = 0$ und mit Nebenbedingung $(M2-2)$ folgt,dass $x_{ji} = 0$ gelten muss, so dass die Nebenbedingung ebenfalls allgemeingültig wird.

Damit haben wir die Nebenbedingung auf Maschinenebene für unser zweites Modell definiert.

Nebenbedingungen auf Jobebene:
Es ist offensichtlich, dass sich die Nebenbedingungen auf Jobebene am stärksten durch das geänderte Optimierungskriterium von den Nebenbedingungen aus dem ersten Modell unterscheiden werden. Betrachten wir die Restriktion für die zeitliche Reihenfolge der Operationen innerhalb eines Jobs, so ist klar, dass diese Nebenbedingung nur auftreten braucht, wenn der zugehörige Job auch wirklich durchgeführt werden soll. D.h., die entsprechende Nebenbedingung lautet:

$$(M2-4) \qquad t_{i+1} \geq t_i + b_i + \sum_{j \in M_{g(i)} \setminus \{i\}} r_{ji} \cdot x_{ji}^{g(i)} + w_i - (1 - y_{l(i)}) \cdot M$$
$$\forall i \in \{n_{l-1}+1, \ldots, n_l - 1\}, \forall l \in \{1, \ldots, |J|\}$$

Dabei erfüllt der Term $(1 - y_{l(i)}) \cdot M$ die gleiche Aufgabe wie der Term $(1 - x_{ji}^{g}) \cdot M$ aus Nebenbedingung $(M1-3)$. D.h., wird der zugehörige Job vergeben $(y_{l(i)} = 0)$, so wird diese Nebenbedingung durch die Konstante M allgemeingültig. Ansonsten entfällt dieser Term, da $y_{l(i)} = 1$ ist. Die Restriktion zur Einhaltung des frühestmöglichen Starttermins lautet in diesem Modell

$$(M2-5) \qquad t_{n_{l-1}+1} \geq T_l^S \cdot y_l \quad \forall l \in \{1, \ldots, |J|\},$$

da der frühestmögliche Starttermin nur eingehalten werden muss, wenn der Job auch durchgeführt werden soll. Analog gehen wir bei der Restriktion für den spätesten Fertigstellungtermin vor, so dass wir als Nebenbedingung

$$(M2-6) \qquad T_l^E - \left(t_{n_l} + b_{n_l} + \sum_{j \in M_{g(n_l)}} r_{jn_l} \cdot x_{jn_l}^{g(n_l)} + w_{n_l} \right) + (1 - y_l) \cdot M \geq 0$$
$$\forall l \in \{1, \ldots, |J|\}$$

erhalten. Wird der Job l nicht durchgeführt, d.h. $y_l = 0$, so wird diese Ungleichung durch M erneut allgemeingültig. Andernfalls muss die vorgegebene Fertigstellungzeit (T_l^E) größer oder gleich der errechneten Fertigstellungzeit (Ausdruck innerhalb der großen Klammer) sein. Um den Schichtplan zu berücksichtigen, können wir die entsprechenden Nebenbedingungen aus dem Modell zur Minimierung des Makespan übernehmen:

$$(M2-7) \qquad \tfrac{t_i}{24} \geq d_i \geq \tfrac{t_i}{24} - \tfrac{2399}{2400} \quad \forall i \in O$$

$$(M2-8) \qquad t_i - 24 \cdot d_i + b_i + \sum_{j \in M_{g(i)}} r_{ji} \cdot x_{ji}^{g(i)} - 24 \cdot v_i \leq S_g \quad \forall i \in O$$

$$(M2-9) \qquad w_i = (24 - S_{g(i)}) \cdot v_i \quad \forall i \in O$$

Zusätzlich können wir die obere Schranke des Schichtüberschreitungsindikators v_i durch die binäre Variable $y_{l(i)}$ ausdrücken mit:

$$(M2-10) \qquad v_i \leq y_{l(i)} \quad \forall i \in O$$

Dies stellt eine Vereinfachung des Modells für den Lösungsalgorithmus dar, da mit der Festlegung der zu vergebenen Jobs die zugehörigen Schichtüberschreitungsindikatorvariablen auf ihre untere Schranke fixiert werden. Die Nebenbedingung $(M2-7)$ stellt für den Fall, dass der zur Operation i gehörende Job vergeben wird, kein Problem dar, da nur ein ganzzahliges d_i innerhalb der Schranken von Restriktion $(M2-7)$ existiert. Da weiterhin die zeitliche Reihenfolge innerhalb des Jobs und auf den Maschinen für die Operationen dieses Jobs wegfallen, kann die kontinuierliche Variable t_i beliebig gewählt werden, so dass sie Nebenbedingung $(M2-8)$ auch für $v_i = 0$ erfüllt. Bereits im Modell zur Minimierung des Makespan hatten wir gesehen, dass $(M2-9)$ eigentlich keine Nebenbedingung darstellt, sondern nur zur besseren Lesbarkeit des Modell eingeführt wurde.

Zielfunktion:
Mit diesem Modell soll die Anzahl der zu vergebenen Jobs minimiert werden. Da hierbei als untere Schranke 0 herauskommen kann, ist ein Vergleich mit dieser Schranke schwierig, da jede Lösung eine relative Lücke zu dieser Schranke von 100% aufweist. Aus diesem Grund wird das Minimierungsproblem in ein Maximierungsproblem umgewandelt, in dem die maximale Anzahl an Jobs ermittelt werden soll, die planmäßig durchgeführt werden können. Es ist dabei offensichtlich, dass die optimalen Belegungspläne dieser beiden Optimierungsprobleme identisch sind. Wir erreichen dies durch folgende Zielfunktion:

$$(M2-Z) \qquad \sum_{l \in J} c_l \cdot y_l \rightarrow \text{Max !}$$

Die Konstante c_l stellt dabei eine Gewichtung der Jobs dar. Mit Hilfe dieses Gewichtes kann die Vergabe von Jobs gesteuert werden, indem Jobs, welche nur in Ausnahmefällen vergeben werden dürfen, mit einem hohen Gewicht versehen werden. Auf die Wahl des Gewichtungsfaktors werden wir im Kapitel 6 näher eingehen.

5.5 Modellerweiterungen

Im weiteren Verlauf des Projektes wurde auf unterschiedliche weitere Fragestellungen eingegangen. Diese beziehen sich zum einen auf die Abbildung von zufallsbehafteten Eingangsdaten und zum anderen auf die Einbeziehung von weiteren Optimierungskriterien, welche simultan zu berücksichtigen sind.

5.5.1 Stochastische Daten

Die Fertigungsaufträge beinhalten einige Aspekte, welche nur auf Erfahrungswerten beruhen und somit eine gewisse Unsicherheit bieten. Diese sind unter anderem:

- Materialverfügbarkeit und somit die frühestmöglichen Starttermine
- Bearbeitungs- und Umrüstzeiten

Weiterhin ist die Verfügbarkeit der Maschinen ebenfalls unsicher, da zum einen die Maschinen oder zum anderen die entsprechenden Mitarbeiter ausfallen könnten. Aus diesem Grund wurden die Möglichkeiten, diese Ungewissheiten einzubeziehen, näher untersucht. Dabei wurde zunächst untersucht, welche Rolle diese Probleme im praktischen Betrieb darstellen und ob genügend Informationen für einen mathematischen Lösungsansatz gegeben sind.

Materialverfügbarkeit:
Die mechanische Fertigung erhält das notwendige Material zum einen von externen Zulieferern und zum anderen von internen Betrieben. Da mit externen Zulieferern in der Regel Verträge geschlossen werden, welche Strafen bei zu später Lieferung vorsehen, werden dort die meisten Liefertermine eingehalten. Bei den internen Betrieben kommt es gelegentlich zu Terminverzögerungen, aber diese sind oftmals rechtzeitig bekannt, so dass eine Neuplanung der Termine möglich wäre.

Ausfall von Maschinen:
Maschinenausfälle stellen zwar prinzipiell ein großes Problem dar. Dies liegt zum einen daran, dass dadurch die Fertigungskapazität sinkt und zum anderen daran, dass Maschinenausfälle oftmals mehrere Tage andauern. Dadurch ist in der Regel eine Neuplanung des Auftragsvolumen unumgänglich und eine Abbildung dieses Aspektes in einem stochastischen Modell nicht notwendig.

Bearbeitungs- und Umrüstzeiten:
In diesem Bereich kommt es am ehesten zu kleinen Veränderungen, welche oftmals nicht eine vollständige Neuplanung der Maschinenbelegung nach sich ziehen. Daher wäre eine Modellierung von stochastischen Bearbeitungs- und/ oder Rüstzeiten aus Sicht des Anwenders hilfreich. Die Möglichkeiten, dann einen optimalen Belegungsplan auszuwählen, können vielseitig sein, da bei stochastischen Optimierungproblemen eine Reihe von Zielfunktionen vorstellbar wären. So wäre zum Beispiel ein Mo-

dell denkbar, welches die erwarteten Vergaben minimiert oder ein Modell, welches die Wahrscheinlichkeit minimiert, dass die Anzahl der Vergaben über einem Schwellenwert liegt. Da der entsprechende Betrieb bei HDW jedoch am ehesten mit einer Unikatfertigung vergleichbar ist, fehlen notwendige Informationen, die für einen mathematischen Lösungansatz notwendig wären. So lässt sich zwar mit hinreichender Gewissheit abschätzen, dass ein gewisser Prozentsatz an Fertigungsaufträgen mit einer falschen Bearbeitungszeit geplant ist, die Wahrscheinlichkeiten mit der diese Aufträge abweichen und selbst die Abweichung selbst kann jedoch nicht quantifiziert werden.

Da zum einen die Bereiche, in denen ein stochastischer Ansatz vorstellbar gewesen wäre, für die Anwendung einen zu geringen Nutzen bedeuten würden und zum anderen die Bereiche, welche einen praktischen Nutzen bringen würden, keinen mathematischen Ansatz bieten, wurde von einer näheren Untersuchung dieser Bereiche abgesehen. Statt dessen wurde verstärkt der folgende Aspekt analysiert.

5.5.2 Mehrkriterielles Scheduling

Die Anzahl der Vergaben zu minimieren, stellt das oberste Ziel der Maschinenbelegungsplanung dar. Darüber hinaus sind jedoch noch andere Zielvorgaben für die Planung wichtig. In Kapitel 2 wurde erläutert, dass der Bereich der mechanischen Fertigung von der Arbeitsvorbereitung einen zusätzlichen Bearbeitungszeitraum von zehn Tagen bekommt. Diese Zeitspanne ist zwar primär dazu gedacht, eine Entscheidung bezüglich der Vergabe des Fertigungsauftrags zu treffen. Allerdings kann diese Zeit auch für die eigentliche Fertigung genutzt werden. Weiterhin sind für die Lieferung des Werkstückes an den nachfolgenden Betrieb weitere fünf Tage eingeplant. Da eine Lieferung innerhalb der Werft normalerweise nicht fünf Tage dauert, kann diese Zeitspanne ebenfalls als Bearbeitungszeit genutzt werden. Aus diesem Grund wäre es sinnlos, einen Fertigungsauftrag zu vergeben, wenn dieser mit einer geringen Verspätung abgearbeitet werden könnte. Daher wurde nach Absprache mit HDW das Modell dahingehend erweitert, dass eine Verspätung von einem Tag bis zu drei Tagen akzeptabel wäre. Dadurch kommt ein zweites Optimierungskriterium in das Modell: Neben der Anzahl der vergebenen Fertigungsaufträge soll zusätzlich die Summe der noch zulässigen Verspätungen minimiert werden. Es ist leicht einsehbar, dass diese beiden Kriterien sich gegenseitig widersprechen. Sind nur kleine Verspätungen erlaubt, so müssen mehr Jobs vergeben werden, als wenn auch größere Verspätungen

noch erlaubt sind.

Durch diese Modelländerung begeben wir uns mit unserem Modell in den Bereich der mehrkriteriellen Optimierung. Bevor wir uns darüber Gedanken machen können, wie unser Modell angepasst werden muss, sollten wir einige Begriffe aus der mehrkriteriellen Optimierung klarstellen. So ist es zunächst nicht klar, was eine optimale Lösung unter dem Gesichtspunkt mehrerer Zielfunktionen ist. Dazu die folgende Definition (siehe auch [8]):

Definition (Schwaches Pareto Optimum)
Gegeben sei eine Menge S, sowie eine Anzahl von Zielfunktionen $Z_i : S \to \mathbb{R}$, $i \in \{1, \ldots, K\}$. Wir nennen $x \in S$ ein schwaches Pareto Optimum genau dann, wenn gilt

$$\nexists y \in S : Z_i(y) < Z_i(x) \forall i \in \{1, \ldots, K\}.$$

Wir gehen bei den jetzigen Begriffsdefinitionen von einem Minimierungsproblem aus. Weiterhin definieren wir:

Definition (Striktes Pareto Optimum)
Es gelten die Bezeichnungen wie in der vorherigen Definition. Dann nennen wir $x \in S$ ein striktes Pareto Optimum genau dann, wenn gilt

$$\nexists y \in S : Z_i(y) \leq Z_i(x) \forall i \in \{1, \ldots, K\}$$

und in mindestens einem Fall trifft $Z_j(y) < Z_j(x)$ zu.

Ein Beispiel für diese Begriffe ist in Abbildung 5.9 dargestellt. Die verschiedenen Zielfunktionen auf der dort dargestellten, zweidimensionalen Menge S sind das komponentenweise Minimum. In Worten ausgedrückt bedeuten die beiden Begriffe Folgendes:

1. Schwaches Pareto Optimum:
   Eine Lösung ist ein schwaches Pareto Optimum dann und nur dann, wenn es keine andere Lösung gibt, die in allen Zielfunktionen einen besseren Wert liefert.

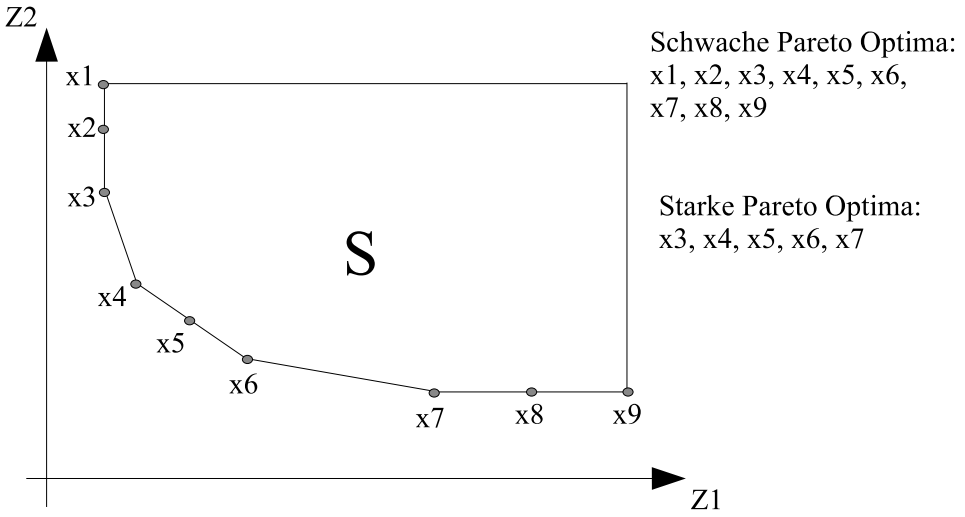

Abbildung 5.9: Veranschaulichung der Begriffe 'schwaches' bzw. 'striktes Pareto Optimum' (vgl. [8]).

2. Striktes Pareto Optimum:
   Eine Lösung ist ein starkes Pareto Optimum dann und nur dann, wenn es keine andere Lösung gibt, die in keiner Zielfunktion einen schlechteren Wert und in mindestens einer Zielfunktion einen besseren Wert liefert.

Damit sollte auch klar sein, dass jedes strikte Pareto Optimum auch ein schwaches Pareto Optimum ist. Spricht man bei einem mehrkriteriellen Optimierungsproblem von einer optimalen Lösung, so ist dabei stets ein Pareto Optimum gemeint. Weiterhin müssen wir noch den Begriff des ordentlichen Pareto Optimums definieren:

Definition (Ordentliches Pareto Optimum)
Seien die Bezeichnungen wie oben gegeben. Weiterhin seien $x, y \in S, x \neq y$ und $I_y := \{i \in \{1, \ldots, K\} : Z_i(y) < Z_i(x)\}$. $x \in S$ wird als ordentliches Pareto Optimum bezeichnet, genau dann, wenn x ein striktes Pareto Optimum ist und ein $M > 0$ existiert, so dass gilt: Für alle $y \in S$ mit $y \neq x$ und $I_y \neq \emptyset$ gilt, dass zu jedem $i \in I_y$ ein $j \in \{1, \ldots, K\}$ existiert mit $Z_j(x) < Z_j(y)$ so dass gilt:

$$\frac{Z_i(x) - Z_i(y)}{Z_j(y) - Z_j(x)} \leq M.$$

Somit können wir zu den verschiedenen Lösungsansätzen für mehrkriterielle Optimierungsprobleme kommen:

Konvexkombination von Kriterien:
Dieser Ansatz wurde von Geoffrion 1968 (vgl. [8]) vorgestellt. Dabei wird eine Konvexkombination der auftretenden Kriterien minimiert. Die zentrale Aussage ist im sogenannten Theorem von Geoffrion zusammengefasst:

Theorem (Geoffrion, 1968)
Sei S eine konvexe Lösungsmenge und Z_i $(1 \leq i \leq K)$ Kriterien die konvex auf S sind. x^0 ist ein Pareto Optimum genau dann, wenn ein $\alpha \in \mathbb{R}^K$ existiert mit $\alpha_i \in]0,1[$ und $\sum_{i=1}^{K} \alpha_i = 1$, so dass x^0 ein Optimum ist für dass Problem P_α:

$$\min g(Z(x)) \text{ mit } g(Z(x)) = \sum_{i=1}^{K} \alpha_i Z_i(x) : x \in S$$

Daraus lässt sich folgendes Lemma herleiten:

Lemma:
Seien S eine konvexe Lösungsmenge und Z_i $(1 \leq i \leq K)$ Kriterien, die konvex auf S sind. x^0 ist ein schwaches Pareto Optimum genau dann, wenn ein $\alpha \in \mathbb{R}^K$ existiert mit $\alpha_i \in [0,1]$ und $\sum_{i=1}^{K} \alpha_i = 1$, so dass x^0 eine optimale Lösung für das Problem P_α ist.

Um also schwache Pareto Optima zu finden, genügt es eine Konvexkombination der Zielfunktionen zu minimieren.

Parametrische Analyse:
Die parametrische Analyse erlaubt es, alle strikten Pareto Optima zu ermitteln. Die Kernaussage basiert dabei auf dem folgenden Theorem (vgl. [8]):

Theorem (Soland, 1979)
Sei G die Menge aller streng monoton wachsenden Funktionen von $\mathbb{R}^K$ nach $\mathbb{R}$, welche nach unten beschränkt sind auf der Menge Z (Z ist das kartesische Produkt der Bildräume von Z_i unter S) und $g \in G$. $x^0 \in S$ ist ein striktes Pareto Optimum genau dann, wenn ein $b \in \mathbb{R}^K$ existiert, so dass x^0 optimale Lösung des folgenden Problems $(P_{(g,b)})$

$$\min g(Z(x)) : x \in S \text{ und } Z(x) \leq b$$

ist.

Diese Aussage kann algorithmisch wie folgt genutzt werden:
Zunächst wird eine Funktion g festgelegt. Anschließend wird iterativ der Vektor b angepasst. Dazu wird z. B. in einem ersten Schritt der Vektor b mit sehr großen Werten initialisiert. Im weiteren Verlauf werden dann die Werte von b entweder durch den Anwender oder durch eine entsprechende Prozedur reduziert.

Der Ansatz über ε-Nebenbedingungen:
In diesem Fall wird nur bzgl. eines Kriteriums optimiert unter der Voraussetzung, dass die übrigen $K-1$ Kriterien nach oben beschränkt sind. Mit Hilfe dieses Ansatzes lassen sich strikte Pareto Optima ermitteln, wie das Theorem von Yu zeigt:

Theorem (Yu, 1974)
$x^0 \in S$ ist ein striktes Pareto Optimum genau dann wenn zu jedem $k \in \{1, \ldots, K\}$ ein $\varepsilon^k = (\varepsilon_1^k, \ldots, \varepsilon_{k-1}^k, \varepsilon_{k+1}^k, \ldots, \varepsilon_K^k) \in \mathbb{R}^{K-1}$ existiert, so dass x^0 die einzige optimale Lösung des folgenden Problems P_{ε^k}

$$\min Z_k(x) : x \in S \text{ und } Z_i(x) \leq \varepsilon_i^k \qquad \forall i \in \{1, \ldots, K\} \setminus \{k\}$$

ist.

Da dieses Theorem schwer zu überprüfen ist (auf Grund der Einzigartigkeit der Lösung), wurde es z. B. von Miettinen 1994 abgewandelt um schwache Pareto Optima zu bestimmen:

Theorem (Miettinen, 1994)
Sei $x^0 \in S$. Wenn ein $k \in \{1, \ldots, K\}$ und ein $\varepsilon^k \in \mathbb{R}^{K-1}$ wie im obigen Theorem existieren, so dass x^0 eine optimale Lösung des Problems (P_{ε^k})

$$\min Z_k(x) : x \in S \text{ und } Z_i(x) \leq \varepsilon_i^k \qquad \forall i \in \{1, \ldots, K\} \setminus \{k\}$$

ist, dann ist x^0 ein schwaches Pareto Optimum.

Die Umkehrung dieses Theorems gilt nur, wenn weiterhin gefordert wird, dass die Menge S und die Funktionen Z_i konvex sind. Dieses Theorem liefert damit zwar nur noch schwache Pareto Optima, allerdings kann es auch einfacher überprüft werden.

Der Einsatz der gewichteten Tschebycheff Metrik:
Setzen wir voraus, dass jedes Kriterium mit einem Gewicht bewertet werden kann, so kann eine Lösung des mehrkriteriellen Optimierungsproblems über die Tschebycheff Metrik erfolgen. Zunächst die Definition der Tschebycheff Metrik:

Definition (Tschebycheff Metrik)
Seien $z^1, z^2 \in \mathbb{R}^K$. Die gewichtete Tschebycheff Metrik ist ein gewichtetes Maß der Entfernung im $\mathbb{R}^K$ zwischen z^1 und z^2 gegeben durch:

$$||z^1 - z^2|| := \max_{i \in \{1,\ldots,K\}} (\lambda_i \cdot |z_i^1 - z_i^2|) \text{ mit } \lambda \in \mathbb{R}_+^k.$$

Die Kernaussage zur Bestimmung von Pareto Optima lässt sich dann wie folgt zusammenfassen:

Theorem (Bowman, 1976)
Wenn $x^0 \in S$ ein striktes Pareto Optimum ist, dann existiert $\lambda \in \mathbb{R}_+^K$, so dass x^0 eine optimale Lösung für das folgende Problem P_λ ist:

$$\min ||Z(x) - z^{\text{ut}}|| : x \in S$$

$z^{\text{ut}} \in \mathbb{R}^k$ stellt einen fiktiven Punkt dar.

Der "Ziel erreicht"-Ansatz
Dieser Ansatz ist ähnlich zur Lösung mittels der Tschebycheff Metrik und geht auf Gembicki und Wierzbicki zurück. Für jedes Kriterium wird ein sogenanntes Ziel definiert. Anschließend wird nach einer Lösung gesucht, welche dieses Ziel am besten approximiert:

Theorem (Gembicki, 1973 / Wierzbicki, 1990)
$x^0 \in S$ ist ein schwaches Pareto Optimum genau dann wenn ein Referenzpunkt $z^{ref} \in \mathbb{R}^K$ und ein Gewichtungsfaktor $w \in \mathbb{R}_+^K$ existieren, so dass x^0 eine optimale Lösung des Problems $P_{(z^{\text{ref}},w)}$

$$\max g(Z(x) : x \in S$$

mit $g(Z(x)) := \min_{1 \leq i \leq K} \left(\frac{1}{w_i} \cdot (z_i^{ref} - Z_i(x) \right)$ ist.

Ansatz über eine gewichtete Summe:

Die obigen Ansätze zur Lösung mehrkriterieller Optimierungsprobleme setzen stets voraus, dass ein sogenannter "Trade Off" zwischen den einzelnen Optimierungskriterien gestattet ist. Für unseren Anwendungsfall hingegen haben wir eine klare Prioritätsreihenfolge zwischen unseren beiden Zielfunktionen. So soll mit allen möglichen Mitteln die Anzahl der Vergaben minimiert werden. Die Minimierung der Verspätungen ist ein zweitrangiges Ziel und sollte nicht auf Kosten des primären Kriteriums durchgeführt werden. In diesen Fällen wird häufig eine gewichtete Summe der beiden Zielfunktionen als neue Zielfunktion angesetzt. Dabei müssen die Gewichte so gewählt werden, dass die primäre Zielfunktion die sekundäre dominiert. Prinzipiell könnte das gleiche Ergebnis auch durch eine iterative Lösung des Optimierungsproblems erfolgen, indem im ersten Schritt bzgl. der primären Zielfunktion optimiert wird. Danach werden die entsprechenden Variablen (in unserem Fall y_l, also die Variablen, welche beschreiben, ob ein Job vergeben wird oder nicht) fixiert und anschließend das entsprechende Modell bzgl. der zweiten Zielfunktion optimiert. Dies ist besonders ratsam, wenn in beiden Iterationen leicht zu lösende Modelle auftreten. Auf Grund der Komplexität des hier vorgestellten Scheduling Problems wenden wir jedoch die erste Vorgehensweise an.

Um diesen mehrkriteriellen Ansatz in das Modell zur Minimierung der Vergaben einzubinden, müssen wir unser Modell wie folgt ändern:

Zunächst führen wir für jeden Job eine Variable $z_l \geq 0$ ein. Sie wird die Verspätung von Job l in Tagen angeben. Mit $D_l \geq 0$ sei die maximal zulässige Verspätung eines Jobs angegeben. Es muss also gelten $z_l \leq D_l$ für alle $l \in J$. Um nun die Verspätungen zu erfassen, müssen wir die Nebenbedingung zur Einhaltung des Fertigstellungstermins wie folgt anpassen:

$$(M2-6) \quad T_l^E + 24 \cdot z_l - \left(t_{n_l} + b_{n_l} + \sum_{j \in M_{g(n_l)}} r_{jn_l} \cdot x_{jn_l}^{g(n_l)} + w_{n_l} \right) + (1 - y_l) \cdot M \geq 0$$
$$\forall l \in \{1, \ldots, |J|\}$$

In dieser Nebenbedingung kann somit der spätestmögliche Fertigstellungstermin T_l^E um bis zu z_l Tage nach hinten verschoben werden. Nehmen wir keine Veränderungen an der Zielfunktion vor, so werden in der optimalen Lösung weiterhin möglichst viele Jobs selbst durchgeführt. Die Anzahl der - innerhalb der maximal zulässigen Verspätung - verspäteten Jobs kann dabei allerdings sehr groß sein. Daher wurde

die Zielfunktion wie folgt geändert:

$$(M2-Z) \qquad \sum_{l \in J} c_l \cdot (3,5 \cdot y_l - z_l) \rightarrow \text{Max !}$$

Vernachlässigen wir den Gewichtungsfaktor c_l, so bedeutet diese Zielfunktion folgendes:
Jeder Job, der selber durchgeführt wird, geht mit dem Gewicht $3,5$ in die Zielfunktion ein. Weiterhin muss die tatsächliche Verspätung von diesem Wert abgezogen werden. Dadurch werden Vergaben stärker bestraft als Verspätungen. In unserem Modell haben wir nicht ausdrücklich verboten, dass ein Job, welcher vergeben wird, keine Verspätung haben darf. Dies ist aber auch nicht notwendig, da jede zulässige Lösung, welche diese Eigenschaft nicht erfüllt, sofort verbessert werden kann, indem die entsprechenden Verspätungen auf 0 gesetzt werden. Das Optimum kann daher nur Verspätungen von Jobs enthalten, welche auch selber durchgeführt werden.

Kapitel 6

Umsetzung

Zu dem beschriebenen Problem wurden verschiedene Algorithmen entwickelt und implementiert. Diese umfassen sowohl exakte Optimierungsverfahren als auch so genannte Heuristiken. Der Vorteil von exakten Verfahren besteht darin, dass mit der Terminierung des Algorithmus das globale Optimum erreicht wurde. Allerdings wird für diese Verfahren oftmals eine lange Rechenzeit bzw. ein hoher Speicherbedarf benötigt. Heuristiken hingegen ermitteln zulässige Punkte und versuchen diese, nach vordefinierten Regeln zu verbessern. Sollte keine Verbesserung mehr möglich sein, so bricht der Algorithmus ab. In der Regel lassen sich so nur "lokale" Optima erreichen, die beliebig weit vom globalen Optimum entfernt sein können. Der Begriff "lokales" Optimum ist hierbei im Zusammenhang mit den vordefinierten Regeln zu verstehen. Diese Regeln definieren eine Umgebung innerhalb der Menge aller zulässigen Punkte und aus diesen wird der beste ermittelt. Wir wollen uns zunächst mit den umgesetzten Heuristiken beschäftigen, um anschließend die Implementierung in Cplex zu beschreiben. Wir haben diese Reihenfolge gewählt, da die beste zulässige Lösung aus den Heuristiken als Startpunkt für das exakte Verfahren (Branch and Bound) verwendet werden kann, um so einen verbesserten Ablauf des Algorithmus zu erzielen.

6.1 Heuristiken

Als heuristische Lösungsvarianten wurden zwei verschiedene Ansätze benutzt. Die erste ist eine Heuristik, die nach dem Prinzip "First-In-First-Out" abläuft. Diese wurde als Grundlage in einem genetischen Algorithmus genutzt, welcher von Frau Vodenina in ihrer Diplomarbeit [12] entwickelt und getestet wurde.

6.1.1 First-In-First-Out-Heuristik

In einem ersten Ansatz einer heuristische Lösungsmethode wurde ein Belegungsplan nach dem First-In-First-Out-Prinzip entwickelt. Diese Methode arbeitet nach dem folgenden Prinzip:

Ablaufschema First-In-First-Out-Heuristik:

1. Initialisiere drei Vektoren *startzeiten_min*, *eingeplant* und *vergeben*. Dabei wird der erste mit den frühestmöglichen Startzeiten der einzelnen Operationen initialisiert, während der zweite und dritte als Boole'sche Vektoren mit false initialisiert werden.

2. Bestimme eine Operation i, welche folgende Eigenschaften erfüllt:

   (a) Die Operation i hat minimalen Wert im Vektor *startzeiten_min*.

   (b) Die Operation i ist nicht mit true im Vektor *eingeplant* markiert.

   (c) Der zu Operation i gehörende Job ist nicht mit true im Vektor *vergeben* markiert.

   Existiert keine solche Operation so STOP.

3. Bestimme nun eine passende Maschine m, so dass Operation i so früh wie möglich auf dieser Maschine bearbeitet werden kann.

4. Plane Operation i auf der Maschine m unter Einhaltung der gegebenen Nebenbedingungen ein und markiere die Operation i im Vektor *eingeplant* mit true.

5. Datiere die Startzeiten im Vektor *startzeiten_min* auf, d.h. aktualisiere die frühestmöglichen Startzeiten derjenigen Operationen, die zum gleichen Job wie Operation i gehören und später durchgeführt werden müssen.

6. Ist Operation i nicht die letzte Operation eines Jobs oder wird der zu Operation i gehörende Job rechtzeitig fertiggestellt, so gehe zu Schritt 2..

7. Markiere im Vektor *vergeben* den zur Operation i gehörenden Job mit true. Lösche den bereits erstellten Belegungsplan und setze die Werte des Vektors *startzeiten_min* auf die ursprünglichen Werte. Außerdem setze alle Einträge im Vektor *eingeplant*, die zu nicht vergebenen Operationen gehören, wieder auf false. Gehe anschließend zu Schritt 2..

Wir werden zunächst nachweisen, dass der obige Algorithmus terminiert und anschließend auf die Vor- und Nachteile eingehen.

Zu Beginn des Algorithmus ist eine endliche Anzahl an Operationen gegeben. Wir können nun zwei Fälle unterscheiden:

1. Fall: Schritt 7 wird nicht durchlaufen.
Dann wird in jedem Durchlauf mindestens eine Operation im Vektor *eingeplant* markiert. Da nur eine endliche Anzahl an Operationen vorhanden ist, muss der Algorithmus terminieren, da Eigenschaft (b) im zweiten Schritt nur endlich oft erfüllt sein kann.
2. Fall: Schritt 7 wird durchlaufen.
In diesem Fall werden alle Operationen, die zu einem bestimmten Job gehören im Vektor *vergeben* markiert. Da nur eine endliche Anzahl an Operationen vorhanden ist, kann es auch nur eine endliche Anzahl an Jobs geben. Da die Anzahl der vergeben Jobs mit jedem Durchlauf durch Schritt 7 erhöht wird, kann Eigenschaft (c) in Schritt 2 nur endlich oft erfüllt werden. Somit muss der Algorithmus terminieren.

Wir haben also gezeigt, dass der Algorithmus nach endlicher Zeit abbrechen muss. Aus dem Ablaufschema wird sofort klar, dass nach Terminierung der Heuristik ein zulässiger Belegungsplan erstellt wurde, in dem genau die Jobs fehlen, welche vergeben werden sollen.

Diese Heuristik bietet einige Vor- und Nachteile, auf die wir im Folgenden eingehen werden. Ein wesentlicher Nachteil besteht darin, dass jede Heuristik nur in der Lage ist, zulässige Lösungen zu generieren. Um Optimalität nachzuweisen, müssen aber für ein Maximierungs-Problem auch obere Schranken generiert werden. Mit diesen oberen Schranken kann dann die Güte einer zulässigen Lösung bewertet werden. Diese Eigenschaft fehlt naturgemäß auch dieser Heuristik. Des Weiteren wird mit dieser Heuristik nur eine einzige zulässige Lösung ermittelt. Diese kann anschließend vom Anwender genutzt oder verworfen werden. Ein automatischer Versuch, diese Lösung mit geeigneten Mitteln zu verbessern, erfolgt nicht. Neben diesen Nachteilen bietet diese Heuristik allerdings auch einige Vorteile. Der primäre Vorteil liegt in dem notwendigen Aufwand. Dieser ist hinsichtlich Speicherkapazität und Rechenzeit selbst für große Probleme eher gering, so dass der Algorithmus selbst auf älteren Rechnern eingesetzt werden kann. Eine Anschaffung von teurer Hardware ist somit

nicht notwendig. Ein weiterer Vorteil liegt in der Art und Weise, wie eine zulässige Lösung generiert wird. Durch die grundlegende Vorgehensweise, jede Operation so früh wie möglich einzuplanen (Schritt 3. und 4.) und die Operationen gemäß ihrer frühestenmöglichen Starttermine (Schritt 2.) einzuplanen, kann ein sehr dichter Maschinenbelegungsplan ermittelt werden, d.h. die Maschinen haben nur geringe Leerlaufzeiten. Außerdem scheint auf den ersten Blick die Bearbeitungszeit der einzelnen Operationen in der gleichen Größenordnung zu liegen, sofern nur die Operationen betrachtet werden, die in einem der beiden Fertigungsbereiche durchgeführt werden sollen. Sie ist zwar nicht identisch, aber die Unterschiede sind eher gering. Dies kommt ebenfalls dieser Art von Heuristik zu Gute. So wird in [2] gezeigt, dass im Falle gleicher Bearbeitungszeiten eine Heuristik, welche jede Operation so früh wie möglich einplant, für die Zielfunktion "Makespan" das Optimum liefern würde, wenn alle Jobs aus der gleichen Anzahl an Operationen bestehen würden und keine Fertigstellungstermine zu berücksichtigen wären.

6.1.2 Genetische Algorithmen

Die obige Heuristik wurde weiterhin in einem genetischen Algorithmus eingesetzt. Genetische Algorithmen sind in Anlehnung an die natürliche Evolution entstanden und fallen in die Klasse der iterativen Algorithmen. Ausgehend von einer Menge, der sogenannten Population, von Lösungen, den sogenannten Individuen, werden Paare von Individuen gebildet. Aus diesen Paaren werden in geeigneter Weise neue Individuen erzeugt. Dadurch wird die Population vergrößert, um im nächsten Schritt nach dem Prinzip "Der Stärkere überlebt" eine gewisse Anzahl von "schlechten" Individuen aus der Population zu entfernen. Mit der neuen Population wird der nächste Iterationsschritt begonnen, sofern nicht gewisse Abbruchkriterien erfüllt sind. Durch dieses Vorgehen wird die Population in jedem Schritt niemals verschlechtert. Es besteht vielmehr die Chance, immer bessere Lösungen zu finden.

In unserem Anwendungsproblem wird der genetische Algorithmus wie folgt angewendet:
Jedes Individuum stellt einen Boole'schen Vektor dar. Dieser gibt an, welche Jobs in dem Belegungsplan liegen sollen und welche nicht. Wir müssen nun sicherstellen, dass ein Belegungsplan zu einem gegebenen Individuum existiert. Dazu können verschiedene Methoden genutzt werden. Auf der einen Seite könnte mit Cplex und einer Modifikation des in Kapitel 4 vorgestellten Modells festgestellt werden, ob ein solcher Belegungsplan existiert oder nicht. Der Datensatz wird in der Regel durch

das Individuum verkleinert (es sei denn, alle Einträge sind 'wahr'), jedoch bestehen interessante Individuen aus sehr vielen Einträgen 'wahr'. Somit kann Cplex auf Grund des hohen Rechenaufwandes zwar theoretisch eingesetzt werden, für praktische Anwendungen ist es in diesem Zusammenhang jedoch unbrauchbar. Auf der anderen Seite kann mit Hilfe der "First-In-First-out"-Heuristik ein Belegungsplan schnell generiert werden. Allerdings kann es vorkommen, dass weitere Jobs vergeben werden müssen, d.h. gewisse Einträge des Individuums von 'wahr' auf 'falsch' geändert werden müssen. Trotzdem werden wir die Heuristik einsetzen, da sie mit vertretbarem Aufwand arbeitet und für unsere Zwecke brauchbare Ergebnisse liefert. Der genetische Algorithmus arbeitet dann nach folgendem Schema:

Ablaufschema genetischer Algorithmus:

1. Initialisierung:
   Initialisiere eine Anfangspopulation P von n Individuen. Dabei ist $n \in \mathbb{N} \setminus \{0\}$ eine gerade Zahl. Setze das erste Individuum auf das Individuum, welches der "First-In-First-Out"-Heuristik entspricht. Für die übrigen Individuen verschlechtere das erste Individuum zufällig um bis zu 10 Prozent.

2. Heiratsschema:
   Bilde zufällige Paare von Individuen der Population.

3. Rekombination:
   Setze $K = \emptyset$. Für jedes Paar (i, j), $i, j \in P$, führe die folgenden Schritte durch:

   - Erzeuge ein Individuum k, welches keine Einträge besitzt.
   - Sind die Einträge i_l und j_l beide 'wahr' so setze k_l ebenfalls auf 'wahr'.
   - Setze die übrigen Einträge von k zufällig auf die Werte 'wahr' oder 'falsch'. Benutze dazu eine gleichverteilte Zufallsvariable.
   - Prüfe mit der "First-In-First-Out"-Heuristik, ob zu dem Individuum k ein Belegungsplan existiert bzw. verändere Individuum k gemäß Ergebnis dieser Heuristik.
   - Füge k zur Menge K hinzu, falls gilt $k \notin P$.

4. Ersetzungsschema:
   Setze nun $P := P \cup K$. Anschließend lösche die schlechtesten Individuen, bis die Population wieder aus n Individuen besteht.

5. Abbruchbedingung:
   Wenn die Abbruchbedingungen nicht erfüllt sind, dann gehe zu Schritt 2, sonst STOP.

Wir werden nun einige Punkte des Algorithmus näher erläutern.

Initialisierung:
In der Initialisierungsphase nutzen wir aus, dass die "First-In-First-Out"-Heuristik bereits einen zulässigen Belegungsplan liefert. Somit ist ein erstes Individuum gegeben. Die übrigen Individuen könnten nun zufällig erzeugt werden. Testbeispiele haben jedoch gezeigt, dass diese zufällige Initialisierung oftmals sehr schlechte Individuen hervorbringt. Daher muss der Algorithmus in den ersten Iterationen diese Lösungen verbessern. Dieser Aufwand konnte vermieden werden, wenn die Individuen nur etwas schlechter als das erste Individuum sind. Daher wurde eine maximal 10 prozentige Verschlechterung des Individuums aus der "First-In-First-Out"-Heuristik gewählt.

Heiratsschema:
Im klassischen genetischen Algorithmus werden die Paare auf Grund der Fitness der einzelnen Individuen gebildet, d.h. es wird geprüft, welche Paare gute Individuen erzeugen können. Da dies in unserem Fall schwierig ist, wurde darauf verzichtet. Die Bildung der Paare erfolgt ausnahmslos zufällig.

Rekombination:
Das *Crossover* stellt im klassischen genetischen Algorithmus die übliche Rekombinationsmethode dar. Dabei werden die beiden Individuen, welche ein neues Individuum erzeugen sollen, an einer zufälligen Stelle durchgeschnitten und über Kreuz miteinander verknüpft. Wir haben diese Methode nicht übernommen, da durch sie die Anzahl der 'wahr'-Einträge nicht verändert wird. Statt dessen gehen wir von Folgendem aus: Ist in zwei Individuen an gleicher Stelle der Eintrag "wahr", d.h. haben beide Individuen den gleichen Job im Belegungsplan, so ist es naheliegend anzunehmen, dass dieser Job "gut" in einen Belegungsplan passt, da er in (mindestens) zwei unterschiedlichen Lösungen enthalten ist. Die übrigen Einträge werden nach dem Zufallsprinzip mit einer gleichverteilten Zufallsvariablen gewählt.

Abbruchbedingung:
Als Abbruchbedingungen bestehen viele Möglichkeiten. Natürlich möchte man aufhören, wenn das Optimum erreicht ist. Allerdings ist es in der Regel sehr schwer nachzuweisen, dass eine Lösung optimal ist. Daher werden andere Abbruchkriterien verwendet. In unserem Fall stoppt der Algorithmus, wenn ...

- ... die Anzahl der Iterationen einen Schwellwert überschreitet.
- ... die Lösung der Heuristik um einen bestimmten Wert verbessert wurde.
- ... die Population eine gewisse Anzahl von Iterationen lang nicht verbessert wurde.
- ... das beste Individuum eine gewisse Anzahl von Iterationen lang nicht verbessert wurde.

Die entsprechenden Werte können natürlich vom Anwender individuell eingestellt werden.

Für weitere Informationen bezüglich des genetischen Algorithmus und den damit durchgeführten Tests wird der Leser auf [12]verwiesen.

6.2 Cplex

Das in Kapitel 4 hergeleitete Optimierungsproblem sollte zunächst mit einem allgemeinen Optimierungstool gelöst werden. Da zum einen Cplex zur Zeit für lineare, gemischt ganzzahlige Optimierungprobleme eine marktführende Position besitzt und entsprechende Lizenzen am Institut für Mathematik vorhanden sind, wurde dieses Programmpaket ausgewählt. Zur Implementierung bietet Cplex dazu eine Reihe von Möglichkeiten an:

1. LP-File:
   Dies stellt die rudimentärste Art und Weise dar, ein Optimierungsproblem zu lösen. Dazu muss das entsprechende Problem in einem sogenannten LP-File abgelegt werden. In dieser Datei müssen die Variablen und Nebenbedingungen explizit angegeben werden. Der Aufruf von Cplex kann dann aus einer Konsole heraus durchgeführt werden.

2. Opl-Studio:
   Dies stellt eine fortgeschrittene Implementierung dar. Innerhalb einer graphischen Oberfläche werden auf der einen Seite das Modell allgemein definiert und auf der anderen Seite entsprechende Datensätze in Dateien hinterlegt. Der Vorteil besteht darin, dass das Modell für neue Datensätze nicht vom Anwender neu formuliert werden muss. Der Aufruf von Cplex erfolgt ebenfalls aus der graphischen Oberfläche heraus.

3. Concert-Technology:
   Wird ein Problem mit Hilfe der Concert-Technology implemtiert, so bietet diese dem Nutzer die größten Freiheiten. Die Concert-Technology stellt C++-Bibliotheken zur Verfügung, über die das Modell implementiert wird. Außerdem kann der Nutzer über diese Bibliotheken auch Einfluss auf den Algorithmus nehmen, sowie andere Algorithmen zuvor ausführen, um die so gewonnenen Lösungen als Startlösung an Cplex zu übergeben.

Eine erste einfache Variante wurde mit Opl-Studio erzeugt. Diese verfügte jedoch noch nicht über alle notwendigen Eigenschaften. Im weiteren Verlauf der Arbeit wurde dann ein Softwareprototyp entwickelt, welcher die Concert-Technology nutzt. Dadurch konnte vor allem eine Heuristik benutzt werden, die eine erste zulässige Lösung für die eigentliche Optimierung liefert. Dies hat den Vorteil, dass Cplex immer eine zulässige Lösung liefert, wenn es auch die Lösung des Heuristik ist. Weiterhin wird durch diese Startlösung in vielen Fällen die Rechenzeit von Cplex reduziert. Nur in eher seltenen Fällen kommt es zu einer längeren Rechenzeit. In Kapitel 8 (Ergebnisse) werden wir sehen, dass die Heuristik bereits sehr gute Lösungen liefert, so dass Cplex oftmals nur noch nachweisen musste, dass die Lösung der Heuristik gut genug ist. Des weiteren konnte durch die vollständige Implementierung in einem C++ Softwarecode eine vereinfachte Dateneingabe und -ausgabe ermöglicht werden, welche in den beiden anderen Varianten für den nichtmathematischen Nutzer in der Regel zu unübersichtlich bzw. zu abstrakt ist. Insgesamt wurden zwei verschiedene Versionen des Prototypen entwickelt. Eine Version liefert eine graphische Benutzeroberfläche mit und ist für den Einsatz beim Projektpartner gedacht. Diese Version ist ausschließlich auf PCs mit Windows-Betriebssystem lauffähig. Da am Institut für Mathematik zum einen die schnellen Rechner ein Linux-Betriebssystem haben und zum anderen die graphische Benutzeroberfläche für intensive Testrechnungen nicht sehr gut geeignet ist, wurde eine Konsolenanwendung entwickelt. Diese ist (nach entsprechender Übersetzung) sowohl auf Windows- als auch auf Linux PCs lauffähig. Da sie mit Ausnahme der graphischen Oberfläche die gleichen Routinen

und Funktionen enthält, sind die Ergebnisse aus Kapitel 7, welche vollständig mit der Konsolenanwendung ermittelt wurden, direkt auf die Anwendung, die auf der Werft eingesetzt werden soll, übertragbar.

Kapitel 7

Anwendungsvorteile

Das in Kapitel 4 entwickelte Modell bietet in Zusammenhang mit den darauf aufbauenden Algorithmen eine Reihe von Vorteilen gegenüber der bisherigen Vorgehensweise. Diese Vorteile haben sich allerdings erst im Laufe des Projektes ergeben. Das primäre Ziel des Projektes war es, eine Methode zu entwickeln, welche auf der einen Seite die Lösungsqualität gegebenüber den bisherigen Lösungen verbessert, auf der anderen Seite vor allem die Zeit für die manuelle Nachbearbeitung reduziert. Dies kann insbesondere durch sehr gute Ergebnisse erreicht werden.

Im Laufe des Projektes war absehbar, dass die entwickelten Algorithmen, insbesondere die Heuristiken, schnell Lösungen liefern, welche für den Anwender auf der Werft hinreichend gute Belegungspläne darstellen. Daher wurde über weitere Anwendungsmöglichkeiten diskutiert. Es stellte sich heraus, dass eine Simulation von verschiedenen Maschinenpark- und Schichtplanszenarien auf großes Interesse beim Anwender stieß. Die Vorteile dieser Simulation liegen auf der Hand:

1. Simulation von Maschinenparkszenarien:
   Durch die Erstellung von Lösungen auf der Grundlage verschiedener Maschinenparks bietet sich die Möglichkeit, Engpässe im Maschinenpark zu identifizieren. Diese Informationen können dann als Grundlage für zukünftige Investitionen bei Neuanschaffungen bzw. Reparaturen von Maschinen dienen. Dadurch lassen sich die finanziellen Mittel besser einsetzen. Die Simulation unterschiedlicher Maschinenparkmodelle bietet weiterhin die Möglichkeit, Kosteneffekte bei nicht identischen Maschinen zu untersuchen. Besteht eine Maschinengruppe z. B. aus zwei Untergruppen von Maschinen, die sich nur darin unterscheiden, dass die Produktion auf den Maschinen zu unterschiedlichen Kosten erfolgt, so kann bei der Erstellung des Belegungsplanes zunächst nur die günstigere Untergruppe benutzt werden und anschließend beide Gruppen.

Der Anwender kann dann in Abhängigkeit von der "Güte" der Belegungspläne entscheiden, ob sich eine Produktion auf den teuren Maschinen lohnt oder nicht. Dabei muss der Anwender sich natürlich Gedanken darüber machen, wie er die Güte misst.

2. Simluation von Schichtplanszenarien:
   Bei HDW verfügen die Mitarbeiter über ein Arbeitsstundenkonto. Daher ist es durchaus üblich, dass in Phasen mit einem hohen Arbeitsvolumen Überstunden angesammelt werden, die in Phasen eines geringeren Arbeitsvolumens abgebaut werden. Da zusätzlich die Schichtpläne mit einer gewissen Vorlaufzeit geändert werden können, ist die Planungsseite daran interessiert, gute Schichtpläne zu erstellen. Diese sollen es dann ermöglichen, die Arbeiten während eines hohen Arbeitsvolumens termingerecht zu erledigen. Da für jede Maschinengruppe ein eigener Schichtplan existiert, kann sich das hohe Arbeitsvolumen durch die Anpassung der Schichtpläne für gewisse Maschinengruppen ausdrücken. Durch die Simulation kann gewährleistet werden, dass nur in den Bereichen Überstunden anfallen, in denen es notwendig ist.

Zwar bietet die bisherige automatische Generierung von Belegungsplänen auch die Möglichkeit, die Schichtpläne und den Maschinenpark zu variieren. Da dieses Programm aber eine enge Verbindung zur Datenbank hat, müssten diese Veränderungen mit hohem Aufwand von der realen Datenbank abgeschirmt werden, da es sonst zu Inkonsistenzen kommen könnte. Außerdem hat die bisherige automatische Generierung den Nachteil, dass die Ermittlung von Vergaben manuell erfolgen muss, was für ausgiebige Simluationen zu aufwändig ist.

Weiterhin bieten die neuen Algorithmen die Möglichkeit, Wartungsaufträge zu planen. Ein Wartungsauftrag wird dazu wie ein "normaler" Auftrag im Datenbestand definiert. Die Algorithmen planen diesen dann ein. Bisher wurden Wartungsarbeiten geplant, indem die entsprechende Maschine für einen bestimmten Zeitraum (in der Regel ein Tag) aus dem Maschinenpark entfernt wurde. Dabei konnte die Wahl des Wartungszeitraums mehr oder weniger gut sein, je nach Erfahrung des Planers. Außerdem bietet sich so die Möglichkeit, dem Wartungsunternehmen exakte Zeitangaben mitzuteilen, wann die Wartung erfolgen kann, sofern sich das Wartungsunternehmen auf diese Vorgehensweise einlässt. Es besteht also nun die Möglichkeit, Wartungsarbeiten besser in den Belegungsplan zu integrieren und weiterhin eine bessere zeitliche Abstimmung mit dem Wartungsunternehmen zu erreichen.

In der letzten Phase des Projektes sind von der Seite des Projektpartners noch weitere denkbare Einsatzmöglichkeiten in Erwägung gezogen worden. So hat der Bereich der mechanischen Fertigung zur Zeit noch ein sehr großes Arbeitsvolumen, welches in den kommenden Jahren auch weiterhin bestehen wird. Allerdings ist man sich im Klaren, dass es auch in diesem Bereich je nach Auftragsentwicklung zu Überkapazitäten kommen kann. In diesem Fall könnten Aufträge von Fremdfirmen bearbeitet werden. Dazu ist es jedoch notwendig sicherzustellen, dass diese Aufträge fristgerecht abgewickelt werden. Ansonsten kann es, wie bei Fertigungsaufträgen an Fremdfirmen üblich, zu Konventionalstrafen kommen.

Kapitel 8

Ergebnisse

In diesem Kapitel werden die Ergebnisse vorgestellt, welche mit dem entwickelten Prototypen erzielt wurden. Im ersten Abschnitt dieses Kapitels werden wir auf die Datengrundlage und die eingesetzte Hardware eingehen. Im zweiten Abschnitt werden dann die bisherigen Rechenergebnisse dargestellt. Dabei wird zunächst auf die Problemgröße eingegangen. Anschließend werden dann Ergebnisse zu verschiedenen Schichtplan- und Maschinenparkmodellen gezeigt, um den Effekt einer Simulation in diesem Bereich zu veranschaulichen. Im letzen Abschnitt werden die gleichen Berechnungen mit einer leicht modifizierten Zielfunktion präsentiert.

8.1 Hardware- und Datengrundlage

Hardwaregrundlagen:
Der Projektpartner war daran interessiert, die Berechnungen auf einem "normalen" Computer durchzuführen. Zwar steht auf der Werft auch ein entsprechender Server zur Verfügung, jedoch stellt dies nur eine Notlösung dar. Daher wurden die Berechnungen auf einem Computer mit den folgenden wichtigen Kennzeichen durchgeführt:

1. CPU: 3Ghz
2. RAM: 2GB

Diese Konfiguration, insbesondere die Speichergröße, war vor allem für die Berechnungen mit Cplex notwendig, da die entstehenden Modelle sehr groß werden (vgl. 7.2.1). Für den Einsatz der Heuristiken ist die Speichergröße nicht wichtig, da diese Algorithmen einen sehr geringen Speicherbedarf haben. Von der CPU-Geschwindigkeit profitieren allerdings alle Algorithmen. Die Softwareanforderungen

Maschinengruppe	Maschinenanzahl	Schichtlänge	Erläuterung
BOHR	1	24	Bohrmaschine
CNC1	5	16	CNC-Bohrmaschine
DREH	12	8	Drehmaschine
FRAES	7	8	Fräsmaschine
SAEGE	1	8	Säge
SCHLEIF	12	8	Schleifmaschine
VERZ	3	16	Verzahnmaschine
STOSS	1	8	Zahnradstoßmaschine

Tabelle 8.1: Maschinenparkmodell der Kleinmechanik in der Ausgangssituation

Maschinengruppe	Maschinenanzahl	Schichtlänge	Erläuterung
BOHRG	3	16	Bohrmaschine
BOWP	2	8	Plattenbohrwerk
CBFK	5	8	Bohrwerk
DREHG	5	8	Drehmaschine
DREHK	4	8	Karusseldrehmaschine
DREHHOB	1	8	Drehhobler
LANGHOB	1	8	Längshobler
FRAESG	1	16	Fräsmaschine
SPREIZ	2	16	Spreizdorn

Tabelle 8.2: Maschinenparkmodell der Großmechanik in der Ausgangssituation

wurden bereits in Kapitel 6, insbesondere im Abschnitt 6.2 dargelegt.

Auf abweichende PC-Konfigurationen wird bei den entsprechenden Testrechnungen dann im Einzelnen eingegangen.

Datengrundlage:

Da das Problem in zwei separate Teilprobleme zerfällt(vgl. 3.1.2), wurden Testrechnungen zu jedem der beiden Teilprobleme durchgeführt. Im Folgenden wollen wir uns mit den betrachteten Maschinenparkmodellen für die beiden Teilprobleme beschäftigen. Die Maschinenparkmodelle sind in ihrer Ausgangslage in den Tabellen 8.1 bzw. 8.2 dargestellt. Die Ausgangslage ist hierbei so zu verstehen, dass, ausgehend von den Ergebnissen mit diesen Maschinenparkmodellen, andere Maschinenpark- und Schichtplanmodelle berechnet wurden. Die Veränderungen dieser Varianten werden an entsprechender Stelle dargestellt.

Sämtliche Auftragsdaten, welche für die Testrechnungen benutzt wurden, stammen aus realen Datenbeständen des Projektpartners. Diese Datensätze mussten jedoch

angepasst werden, da sie einige Inkonsistenzen enthielten. So waren in bisher allen Datensätzen zum Teil Bearbeitungszeiten von 0 Stunden geplant. Dieses Phänomen konnte bisher nicht erklärt werden. Daher wurden Aufträge mit fehlerhaften Daten der Einfachheit halber aus den Datensätzen entfernt. Anschließend wurden Datensätze mit 100 bis 500 Operationen zusammengestellt. Dieses Auftragsvolumen musste innerhalb einer Woche eingeplant werden. Größere Datensätze konnten nur auf einem Server, bzw. nur noch mit der Heuristik bearbeitet werden, da die entstehenden Modelle sonst zu groß wurden.

8.2 Rahmenbedingungen der Testrechnungen

Wie in Kapitel 6 bereits angesprochen, wurden alle Testrechnungen auf Linux-Systemen durchgeführt. Es wurde eine maximale Rechenzeit von 10 Stunden angenommen, um ein Problem zu lösen. Dieser Wert erscheint zwar sehr groß, jedoch ist es vom Projektpartner durchaus gewünscht, z.B. die Optimierung am Montag zu beginnen. Die Ergebnisse müssen dann am Dienstag vorliegen. Daher stellt die Rechenzeit keinen kritischen Anforderungspunkt an die Softwarelösung. Wie in Kapitel 4 bereits erläutert, werden zur Zeit bis zu 30% aller Jobs an Fremdfirmen vergeben. Da bei den vorliegenden Datensätzen die First-In-First-Out Heuristik bereits Ergebnisse lieferte, die deutlich unter diesen 30% liegen, wurde eine Lösung als zufriedenstellend betrachtet, wenn der Zielfunktionswert um maximal 1% noch verbessert werden könnte. Im realen Anwendungsfall wird dieser Wert eher vergrößert werden.

Sollten Testrechnungen mit anderen Rahmenbedingungen durchgeführt werden, so wird dies an der entsprechenden Stelle vermerkt.

8.3 Ergebnisdarstellungen

8.3.1 Problemgrößen

In diesem Abschnitt wollen wir uns mit der Größe der entstehenden Modelle beschäftigen. Dazu betrachten wir die Aufragsdaten aus fünf Szenarien der Kleinmechanik mit 100 bis 500 Operationen. Man könnte zunächst annehmen, dass eine Verdoppelung des Datenbestandes auch zu einer Verdoppelung des Problems führt. Dies

Datensatz	(M2-7)...(M2-10)	(M2-5)/(M2-6)	(M2-4)	(M2-1)	(M2-2)	(M2-3)
W23KM-100	351	89	13	102	141	1962
W23KM-200	691	166	36	202	243	9698
W23KM-300	1041	243	60	303	344	24686
W23KM-400	1381	322	81	403	444	46432
W23KM-500	1749	396	111	507	549	76866

Tabelle 8.3: Übersicht über die Anzahl der aufzustellenden Nebenbedingungen

ist jedoch ein Irrtum. Betrachten wir dazu die Anzahl an Nebenbedingungen und Variablen, die notwendig sind, um das Optimierungsproblem gemäß Abschnitt 4.4.2 zu erstellen, so erkennen wir in Abbildung 8.1, dass sich die Probleme überlinear entwickeln. Diese Entwicklung bei den Variablen ist bei Hinzuziehung des Modells allerdings nicht überraschend. Dazu ein kleines Beispiel bei dem wir nur die Anzahl der Variablen x_{ij}^g für eine Maschinengruppe betrachten:

Anzahl Maschinen	Anzahl Operationen	x_{ij}^g-Variablen
3	7	97
3	8	118
3	9	141

Um die Anzahl der x_{ij}^g-Variablen zu berechnen, wird die Summe aus der Anzahl an Maschinen und Operationen der Maschinengruppe g quadriert. Damit haben wir die Anzahl an Variablen, wenn jede Maschine und Operation genau einen Vorgänger und genau einen Nachfolger haben muss. Da jedoch die Maschinen keinen Vorgänger benötigen, subtrahieren wir von diesem Ergebnis die Anzahl der Maschinen. Um das überlineare Wachstum des Modells bzgl. der Restriktionen zu analysieren, untersuchen wir die Anzahl der auftretenden Nebenbedingungen. Diese sind in Tabelle 8.3 aufgeführt. Dabei fällt insbesondere ein sehr schnelles Anwachsen der Restriktionsart (M2-3) auf. Diese Nebenbedingung stellt sicher (vgl. 4.4.2), dass eine Operation erst dann auf einer Maschine beginnen kann, wenn die vorherige Operation auf dieser Maschine beendet wurde, mit anderen Worten, sie gewährleistet die zeitliche Reihenfolge auf den Maschinen. Da dies für jede Kombination von zwei Operationen auf einer Maschinengruppe abgefragt werden muss, ist das überlineare Wachstum nicht mehr verwunderlich.

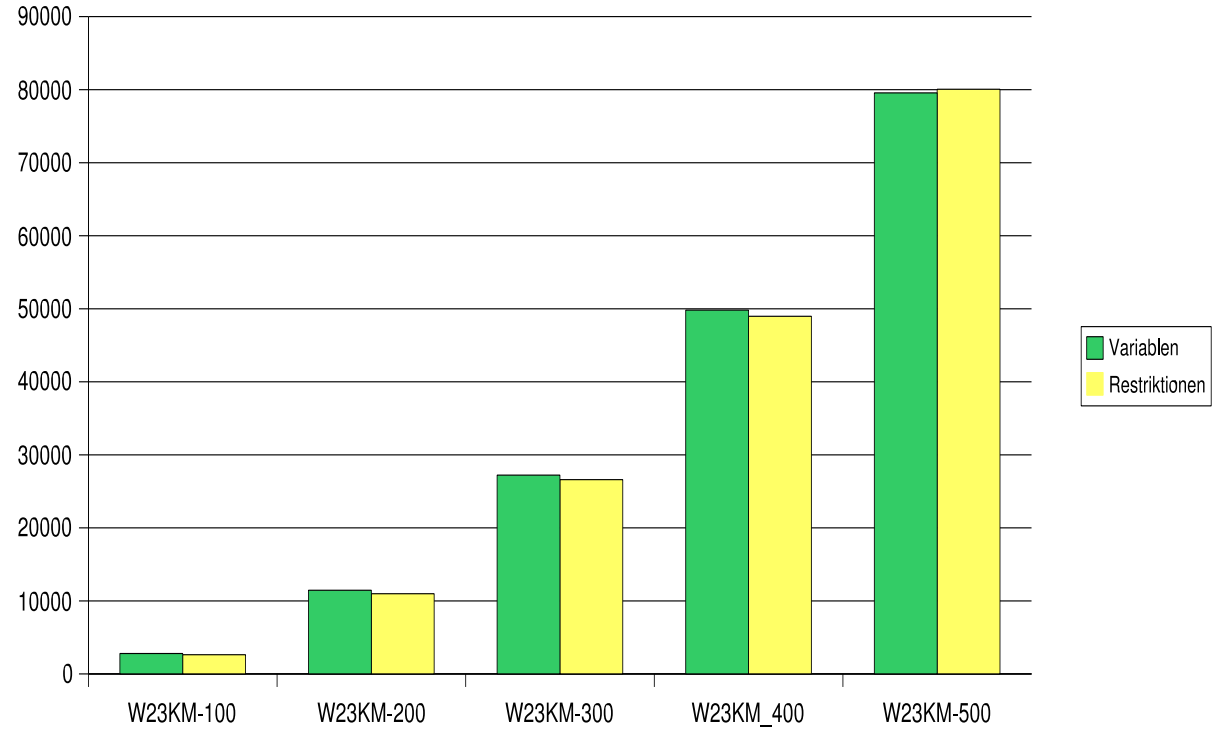

Abbildung 8.1: Entwicklung der Anzahl der Variablen und Restriktionen zur Erstellung des Modells gemäß 4.4.2

8.3.2 Maschinenpark- und Schichtplanmodelle

Im ersten Abschnitt werden Ergebnisse präsentiert, welche im realen Anwendungseinsatz von größtem Interesse sind. Es werden verschiedene Datensätze mit unterschiedlichen Schichtplan- und Maschinenparkszenarien untersucht. Die allgemeine Maschinenparkdatei (vgl. 8.1) wurde dabei zunächst angenommen. Anschließend wurde diese wie folgt geändert:

Bezeichnung	Maschinengruppe	Anzahl Maschinen	Schichtlänge
Kleinmechanik01	BOHR	1	24
Kleinmechanik02	BOHR	1	12
Kleinmechanik03	BOHR	2	24

Tabelle 8.4: Übersicht über die Veränderungen der Maschinenparkdatei "Kleinmechanik"

Mit "Kleinmechanik01" ist dabei die ursprüngliche Maschinenparkdatei bezeichnet. Die Ergebnisse dieser Tests sind in den Tabellen 8.5, 8.6 und 8.7 zusammengefasst. Wir wollen nun einige beispielhafte Erkenntnisse aus diesen Tests näher diskutieren:

Daten	Max.-Verspätung	Lücke (%)	Zeit	Heursitik	Cplex
W23KM-100	0 (Tage)	1,14	>10h	308,00	308,00
W23KM-200	0 (Tage)	3,11	>10h	563,50	563,50
W23KM-300	0 (Tage)	10,96	>10h	766,50	766,50
W23KM-400	0 (Tage)	10,65	>10h	1018,50	1018,50
W23KM-500	0 (Tage)	15,79	>10h	1197,00	1197,00
W23KM-100	1 (Tage)	0,04	1s	311,39	311,39
W23KM-200	1 (Tage)	0,20	19s	579,83	579,83
W23KM-300	1 (Tage)	0,95	233s	842,53	842,53
W23KM-400	1 (Tage)	1,97	>10h	1105,23	1105,23
W23KM-500	1 (Tage)	4,17	>10h	1330,47	1330,47
W23KM-100	2 (Tage)	0,04	1s	311,39	311,39
W23KM-200	2 (Tage)	0,20	23s	579,83	579,83
W23KM-300	2 (Tage)	0,95	262s	842,53	842,53
W23KM-400	2 (Tage)	1,34	>10h	1112,10	1112,10
W23KM-500	2 (Tage)	2,25	>10h	1355,55	1355,55
W23KM-100	3 (Tage)	0,04	1s	311,39	311,39
W23KM-200	3 (Tage)	0,20	23s	579,83	579,83
W23KM-300	3 (Tage)	0,95	262s	842,53	842,53
W23KM-400	3 (Tage)	1,34	>10h	1112,10	1112,10
W23KM-500	3 (Tage)	2,22	>10h	1355,86	1355,86

Tabelle 8.5: Tabelle der Ergebnisse zur Variation der Maschinenparkdatei bzgl. Kleinmechanik01, dem Ausgangsszenario. Die letzten beiden Spalten geben den jeweiligen Zielfunktionswert gemäß des Modells aus 5.5.2 an.

Vergleich der First-In-First-Out Heuristik mit dem Cplexmodul:

Bei den Testrechnungen konnte das exakte mathematische Verfahren die durch die Heuristik ermittelte Lösung nur in einem Fall (siehe Tabelle 8.6) verbessern. Der Grund für dieses Verhalten liegt zum einen in der Güte der ermittelten Lösungen und zum anderen in der Komplexität des Problems. So konnte das exakte Verfahren in einigen Fällen feststellen, dass die gefundene Lösung nur noch um weniger als ein Prozent verbessert werden kann. In diesen Fällen konnte das exakte Verfahren den Optimierungsprozess beenden. In den übrigen Fällen wurde von Cplex eine Optimierung eingeleitet, jedoch konnte innerhalb des Zeithorizontes weder eine bessere Lösung noch eine bessere Schranke für das Problem ermittelt werden. Dies lässt darauf schließen, dass die von der Heuristik gefundene Lösung bereits recht gut ist und nicht auf triviale Art und Weise verbessert werden kann.

Weiterhin fällt auf, dass das von Cplex ermittelte Optimierungspotential (in der Spalte Lücke als maximal erreichbare relative Verbesserung der Zielfunktion angegeben), einer starken Schwankung unterliegt. Dies liegt vor allem am jeweiligen Datensatz. Die Größe des Datensatzes liegt zwischen 100 und 500 Operationen, der

Zeitraum, der für die Einplanung dieser Operationen zur Verfügung steht, bleibt jedoch konstant bei einer Woche. Wir erinnern uns daran (vgl. Abschnitt 4.1.3), dass ein Optimierungszeitraum von 2 bis 4 Wochen mit 1200 bis 2100 Operationen angestrebt war, d.h. pro Woche fallen ca. 350 Operationen im Bereich der Kleinmechanik an. Somit ist klar, dass die Szenarien mit 500 Operationen Ergebnisse mit einer großen Optimalitätslücke produzieren. Betrachten wir die Szenarien mit 300 bzw. 400 Operationen bzgl. des relativen Anteils an Vergaben genauer, so sieht dies schon wesentlich besser aus (vgl Abbildung 8.2). Wir sehen dort, dass selbst bei dem schlechten Szenario Kleinmechanik02 der relative Anteil der Vergaben unter 15% bleibt. Dies stellt im Vergleich zur bisherigen Vorgehensweise, mit bis 30% an vergebenen Jobs, eine Halbierung dar. Weiterhin sieht man, dass bei einer zulässigen Verspätung von nur einem Tag der Anteil der vergebenen Jobs auf unter 5% reduziert werden kann.

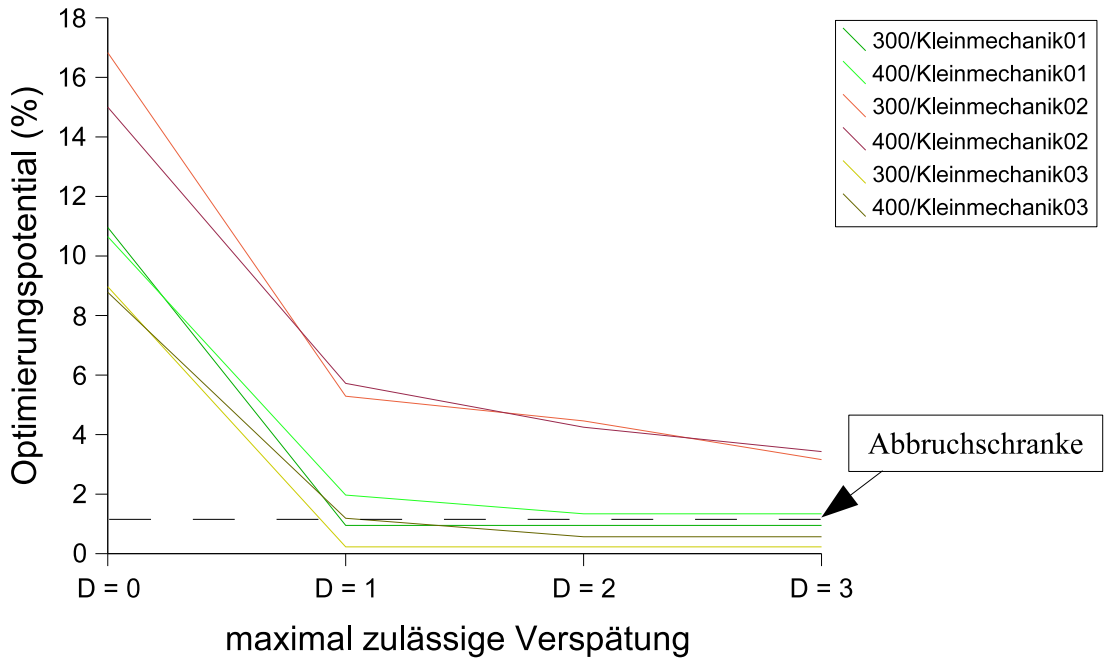

Abbildung 8.2: Übersicht über das relative Optimierungspotential der Datensätze mit 300 bzw. 400 Operationen bzgl. der verschiedenen Maschinenparkszenarien und maximal zulässigen Verspätungen (D).

Wir werden nun auf die Entwicklung der Vergaben im Bezug auf die Anzahl der verspäteten Jobs eingehen.

Daten	Max.-Verspätung	Lücke (%)	Zeit	Heuristik	Cplex
W23KM-100	0 (Tage)	14,10%	>10h	273,00	273,00
W23KM-200	0 (Tage)	10,67%	>10h	525,00	525,00
W23KM-300	0 (Tage)	16,83%	>10h	728,00	728,00
W23KM-400	0 (Tage)	15,00%	>10h	980,00	980,00
W23KM-500	0 (Tage)	20,00%	>10h	1155,00	1155,00
W23KM-100	1 (Tage)	1,94%	>10h	305,57	305,57
W23KM-200	1 (Tage)	2,42%	>10h	567,29	567,29
W23KM-300	1 (Tage)	5,29%	>10h	807,76	807,76
W23KM-400	1 (Tage)	5,72%	>10h	1066,07	1066,07
W23KM-500	1 (Tage)	7,67%	>10h	1287,25	1287,25
W23KM-100	2 (Tage)	0,91%	41s	307,96	308,70
W23KM-200	2 (Tage)	1,26%	>10h	573,76	573,76
W23KM-300	2 (Tage)	4,46%	>10h	814,16	814,16
W23KM-400	2 (Tage)	4,25%	>10h	1081,07	1081,07
W23KM-500	2 (Tage)	5,41%	>10h	1314,91	1314,91
W23KM-100	3 (Tage)	1,15%	>10h	307,96	307,96
W23KM-200	3 (Tage)	1,26%	>10h	573,76	573,76
W23KM-300	3 (Tage)	3,16%	>10h	824,46	824,46
W23KM-400	3 (Tage)	3,43%	>10h	1089,66	1089,66
W23KM-500	3 (Tage)	5,02%	>10h	1319,74	1319,74

Tabelle 8.6: Tabelle der Ergebnisse zur Variation der Maschinenparkdatei bzgl. Kleinmechanik02, Variation der Maschinengruppe 'BOHR' (halbierte Schichtlänge im Vergleich zum Ausgangsszenario)

Entwicklung der Vergaben und Verspätungen:

Interessanter als die Entwicklung der Zielfunktion dürfte für den Anwender die Entwicklung der Vergaben und verspäteten Jobs in Abhängigkeit von den jeweils angenommenen Maschinenparkszenarien sein. Darauf wollen wir nun detailierter eingehen. Insbesondere wollen wir dabei aufzeigen, wie groß die maximal zulässige Verspätung sein muss und wie viele verspätete Jobs auftreten, um die vergebenen Jobs doch selbst zu fertigen. Die Entwicklung der Vergaben und der verspäteten Jobs ist in den Abbildungen 8.3 bis 8.7 dargestellt. Es ist dabei auffällig, dass in den Datensätzen mit 100 bis 300 Operationen in der Regel eine maximal zulässige Verspätung von einem Tag ausreicht, um einen Maschinenbelegungsplan zu generieren, welcher ohne Vergaben auskommt. Nur im Fall des Maschinenparkszenarios Kleinmechanik02 trifft dies für den Datensatz mit 300 Operationen nicht zu. Dies ist auf die extreme Kapazitätsreduzierung im Szenario zurückzuführen. Selbst im Datensatz mit 500 Operationen, welcher eigentlich mehr Operationen enthält als pro Woche durchzuführen sind, bewirkt eine maximal zulässige Verspätung von einem Tag eine Reduzierung der Vergaben um ca. 78, 61 bzw. 83% je nach Maschinen-

Daten	Max.-Verspätung	Lücke (%)	Zeit	Heuristik	Cplex
W23KM-100	0 (Tage)	0,00%	>10h	311,50	311,50
W23KM-200	0 (Tage)	1,22%	>10h	574,00	574,00
W23KM-300	0 (Tage)	8,97%	>10h	780,50	780,50
W23KM-400	0 (Tage)	8,78%	>10h	1036,00	1036,00
W23KM-500	0 (Tage)	13,47%	>10h	1221,50	1221,50
W23KM-100	1 (Tage)	0,00%	1s	311,50	311,50
W23KM-200	1 (Tage)	0,03%	19s	580,80	580,80
W23KM-300	1 (Tage)	0,23%	233s	848,54	848,54
W23KM-400	1 (Tage)	1,19%	>10h	1113,75	1113,75
W23KM-500	1 (Tage)	2,57%	>10h	1351,24	1351,24
W23KM-100	2 (Tage)	0,00%	1s	311,50	311,50
W23KM-200	2 (Tage)	0,03%	23s	580,80	580,80
W23KM-300	2 (Tage)	0,23%	262s	848,54	848,54
W23KM-400	2 (Tage)	0,57%	>10h	1120,62	1120,62
W23KM-500	2 (Tage)	1,52%	>10h	1366,55	1366,55
W23KM-100	3 (Tage)	0,00%	1s	311,50	311,50
W23KM-200	3 (Tage)	0,03%	23s	580,80	580,80
W23KM-300	3 (Tage)	0,23%	262s	848,54	848,54
W23KM-400	3 (Tage)	0,57%	>10h	1120,62	1120,62
W23KM-500	3 (Tage)	1,40%	>10h	1366,87	1366,87

Tabelle 8.7: Tabelle der Ergebnisse zur Variation der Maschinenparkdatei bzgl. Kleinmechanik03, Variation der Maschinengruppe 'BOHR' (verdoppelte Maschinenanzahl im Vergleich zum Ausgangsszenario)

parkszenario. Die Anzahl der verspäteten Jobs liegt dabei in der Regel nur leicht über der Anzahl der Vergaben, im Vergleich zum Ergebnis mit um einen Tag reduzierter maximal zulässiger Verspätung. Somit kann man fast sagen, dass nur die Jobs verspätet sind, welche sonst vergeben worden wären. Prinzipiell müssten die Ergebnisse noch dahingehend untersucht werden, ob die verspäteten Jobs auch tatsächlich mit dieser Verspätung fertiggestellt werden sollen. Unter Umständen könnten Jobs darunter sein, welche auf jeden Fall pünklich fertiggestellt werden müssen. Da für diese Untersuchung jedoch weitere Informationen bzgl. der Jobs vorliegen müssen, wird darauf an dieser Stelle verzichtet.

8.3.3 Cplexmodul ohne Startlösung

Um zu erkennen, welchen Anteil die Startheuristik an den obigen Rechenergebnissen hat, wurden auch Testrechnungen durchgeführt, bei denen Cplex ohne Kenntnisse aus der Startheuristik die gestellten Probleme lösen musste. Dazu wurden ebenfalls

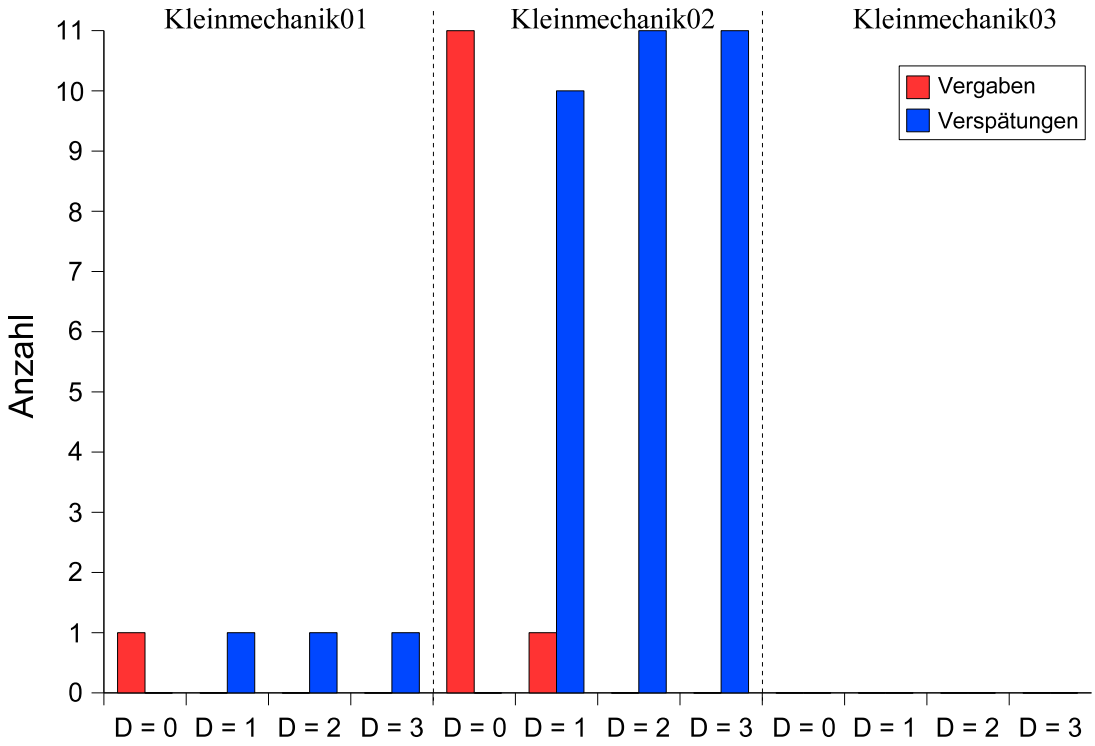

Abbildung 8.3: Übersicht über die Entwicklung der Vergaben und Verspätungen von W23KM-100 in Abhängigkeit von der maximal zulässigen Verspätung (D) in Tagen, sowie den verschiedenen Maschinenparkszenarien.

die verschiedenen Maschinenparkszenarien angenommen. Zur Vereinfachung wurden nur maximal zulässige Verspätungen von 0 bzw. 3 Tagen getestet. Die Ergebnisse sind in Tabelle 8.8 dargestellt.

Es ist sofort auffällig, dass Cplex mit Datensätzen, welche mehr als 200 Operationen umfassen, Probleme hat. Bei diesen Datensätzen verbleibt am Ende des Optimierungsvorganges ein sehr großes Optimierungspotential. Nur bei den kleineren Datensätzen konnte (zum Teil abhängig vom Maschinenparkszenario und der maximal zulässigen Verspätung) ein akzeptables Ergebniss erzielt werden. In wenigen Fällen wurde sogar die optimale Lösung ermittelt. Vergleichen wir das verbleibende Optimierungspotential ohne Kenntnisse aus der Startheuristik mit dem verbleibenden Optimierungspotenzial mit Kenntnissen aus der Startheuristik, so erkennen wir, welchen Einfluss diese zusätzlichen Informationen auf den Ablauf des Optimierungsalgorithmus haben. In fast allen Fällen haben sich die zusätzlichen Informationen ausgezahlt und zu besseren Ergebnissen geführt. Nur in einem Fall (Datensatz: W23KM-100, Maschinenpark: Kleinmechanik02, max. Verspätung: 3 Tage) konnte ohne die Informationen aus der Startheuristik ein besseres Ergebnis erzielt werden.

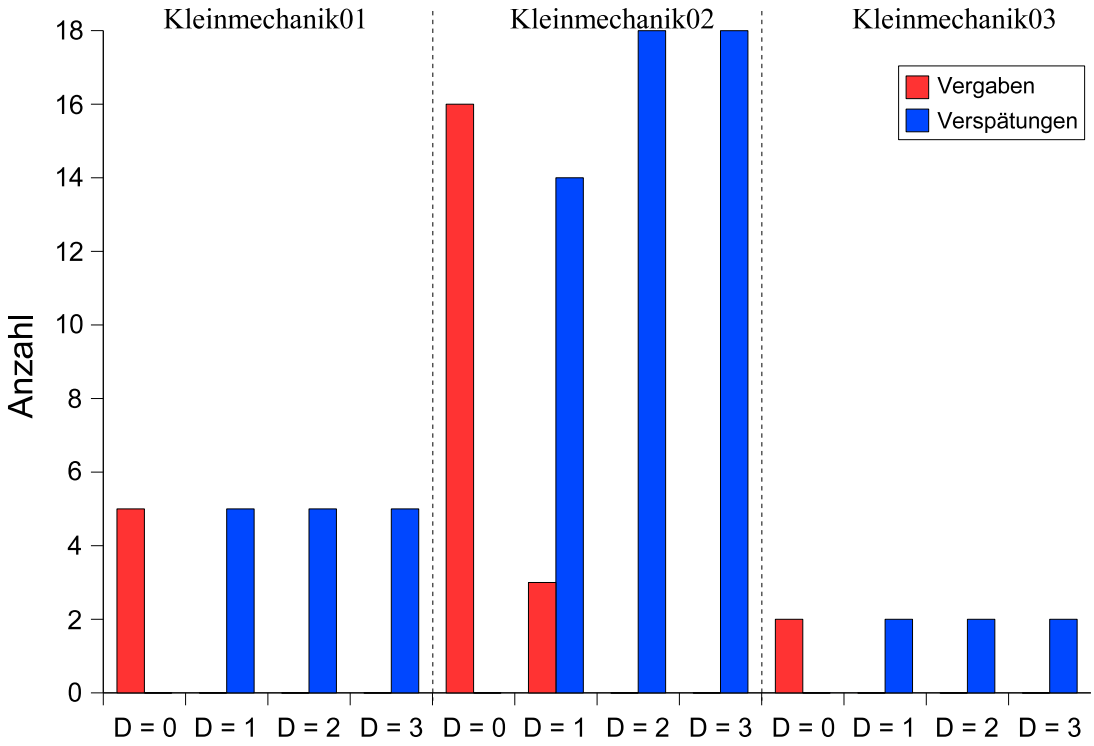

Abbildung 8.4: Übersicht über die Entwicklung der Vergaben und Verspätungen von W23KM-200 in Abhängigkeit von der maximal zulässigen Verspätung (D) in Tagen, sowie den verschiedenen Maschinenparkszenarien.

Allerdings sehen wir auch, dass es sich dabei nur um eine sehr geringe Verbesserung handelt (1,15%→ 1,08%).

8.3.4 Variierte Zielfunktion

In den bisherigen Testrechnungen sollte die Zielfunktion $\sum c_l \cdot (3,5 \cdot y_l - z_l)$ maximiert werden. Dabei war $c_l = 1$, d.h. zwischen den einzelnen Jobs wurde nicht weiter differenziert. Dies ist gerade in unserem Anwendungskontext nicht der Fall, da es Bauteile gibt, die wichtiger sind als andere. Dies kann über eine entsprechende Wahl des Gewichtungsfaktors c_l abgebildet werden. Es stehen dazu mehrere Möglichkeiten zur Verfügung:

- Bewertung mit Kosten:
  Der Gewichtungsfaktor c_l kann sich aus den auftretenden Kosten zusammensetzen. Dabei können sowohl die Kosten für eine Vergabe bzw. die Eigenfertigung des entsprechenden Jobs angesetzt werden.

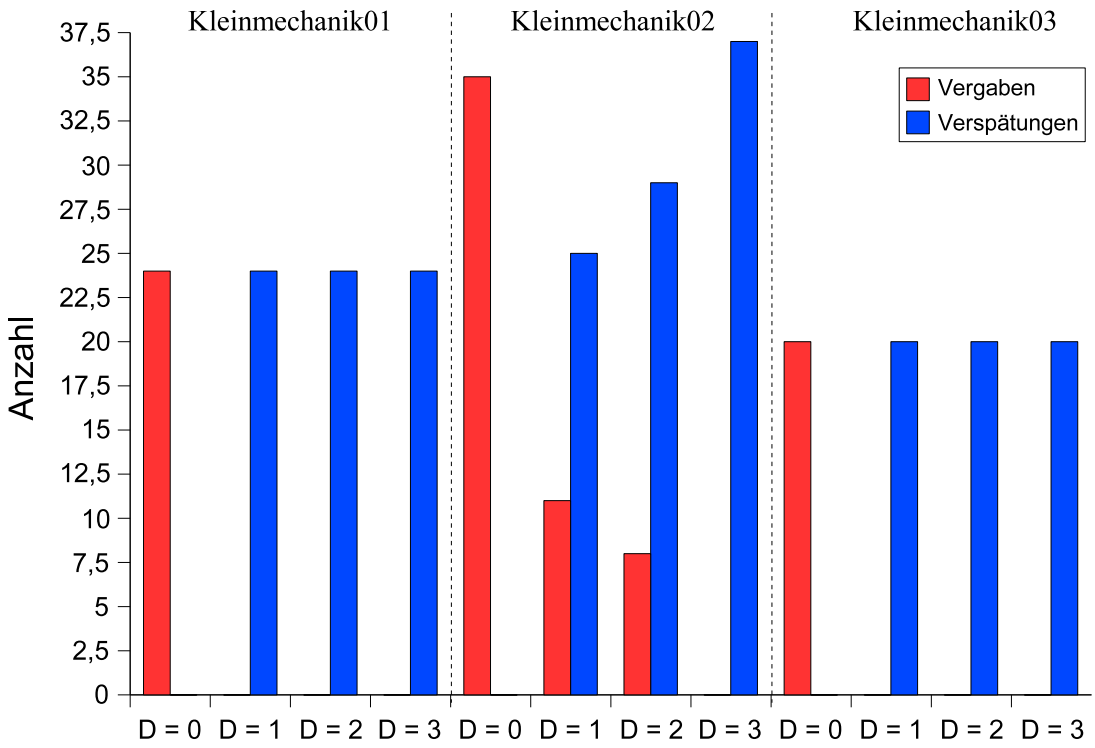

Abbildung 8.5: Übersicht über die Entwicklung der Vergaben und Verspätungen von W23KM-300 in Abhängigkeit von der maximal zulässigen Verspätung (D) in Tagen, sowie den verschiedenen Maschinenparkszenarien.

- Bewertung mit Zeit:
  Der Gewichtungsfaktor kann abhängig von der notwendigen Bearbeitungszeit der einzelnen Operationen festgelegt werden. Hierbei kann auch nach der Bearbeitungszeit auf verschiedenen Maschinengruppen weiter differenziert werden.

Es sind theoretisch weitere Ansatzmöglichkeiten denkbar. Auf Grund der vorhandenen Daten wurde eine zeitliche Bewertung des Gewichtungsfaktors angesetzt. Eine weitere Differenzierung bzgl. einzelner Maschinengruppen erfolgte allerdings nicht, d.h. es gilt: $c_l := \sum_{i \in J_l} b_i$.

Es wurden Testrechnungen mit dieser Zielfunktion bzgl. der verschiedenen Maschinenparkszenarien durchgeführt. Die maximal zulässige Verspätung wurde mit 0 bzw. 3 Tagen angesetzt. Die Ergebnisse sind in Tabelle 8.9 zusammengefasst.

Zunächst bleibt festzuhalten, dass bei diesen Testrechnungen die Lösung der Heuristik nur selten verbessert werden konnte. Zwar konte sie von Cplex häufiger verbessert werden als bei den Testrechnungen mit der ursprünglichen Zielfunktion, trotz-

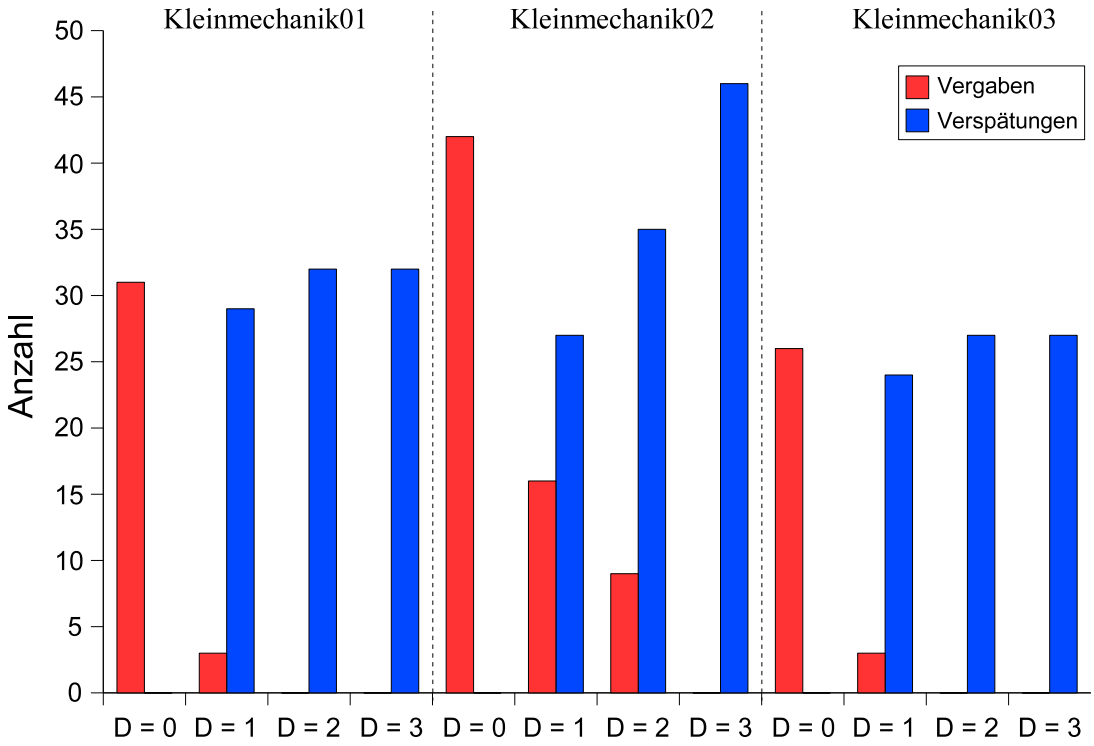

Abbildung 8.6: Übersicht über die Entwicklung der Vergaben und Verspätungen von W23KM-400 in Abhängigkeit von der maximal zulässigen Verspätung (D) in Tagen, sowie den verschiedenen Maschinenparkszenarien.

dem überwiegen noch die Testrechnungen, in denen Cplex keine Fortschritte erzielen konnte. Vergleichen wir die Lücke in diesen Berechnungen mit der Lücke bzgl. der ursprünglichen Zielfunktion, so stellen wir fest, dass in vielen Fällen eine Verbesserung der Lücke werden konnte. Dies ist darauf zurückzuführen, dass mit der veränderten Zielfunktion eine unterschiedliche Bewertung der einzelnen Jobs erreicht wird. Dies ist insbesondere für Cplex sehr hilfreich. Außerdem bleibt festzuhalten, dass die Optimalitätslücke bzgl. der veränderten Zielfunktion mit wenigen Ausnahmen, die im Maschinenparkszenario "Kleinmechanik02" auftreten, sehr klein ist (oftmals unter 5%) und somit das Ergebnis für Anwendungszwecke eine hervorragende Grundlage darstellt.

Weiterhin wurden Testrechnungen mit der variierten Zielfunktion und ohne Kenntnisse aus der Startheuristik durchgeführt. Diese sind in Tabelle 8.10 aufgeführt. Wir sehen hier ein ähnliches Bild wie in Tabelle 8.8. Offensichtlich konnte Cplex durch die veränderte Zielfunktion zwar die Optimalitätslücke zum Teil deutlich reduzieren, sie liegt aber in fast allen Fällen noch zu weit von einem akzeptablen Wert entfernt.

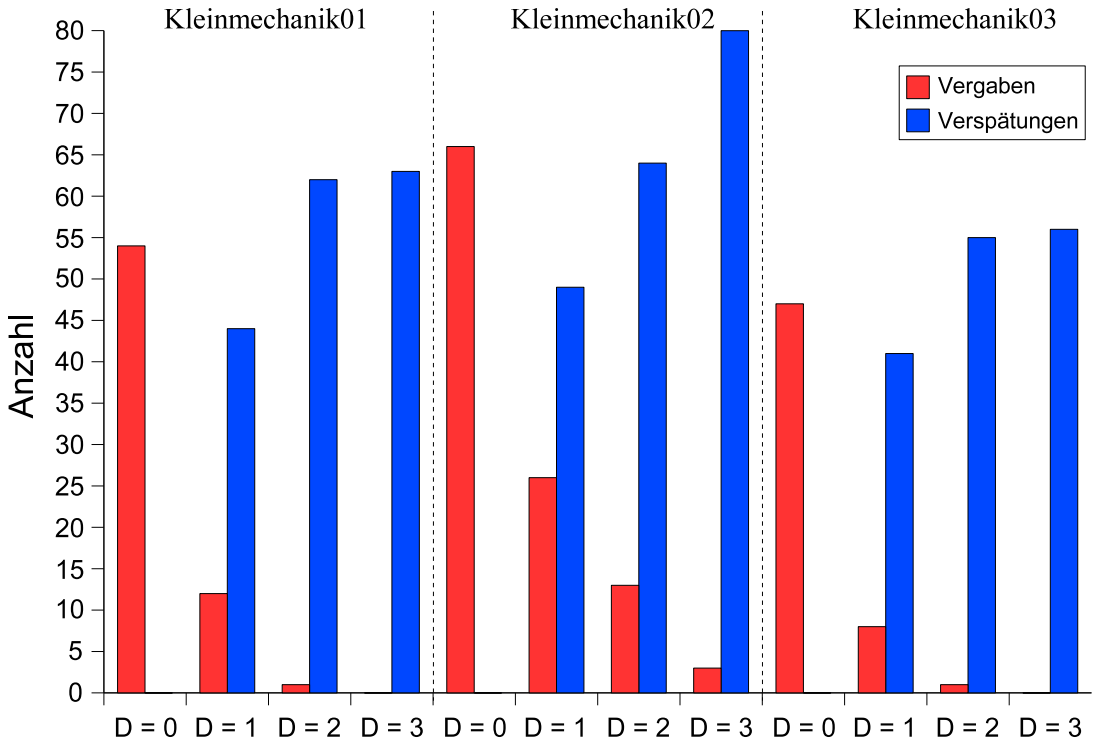

Abbildung 8.7: Übersicht über die Entwicklung der Vergaben und Verspätungen von W23KM-500 in Abhängigkeit von der maximal zulässigen Verspätung (D) in Tagen, sowie den verschiedenen Maschinenparkszenarien.

8.3.5 Ergebnisse mit anderen Datensätzen

Da für die folgenden Testrechnungen das Cplexmodul auf Grund der Problem größe nicht verwendet wurde, wurden diese Testrechnungen auf einem Athlon XP2000+ (getaktet mit 1,66 Ghz) und 512 MB RAM durchgeführt. Die Rechenzeit lag bei allen Versuchen unter 20 Minuten, so dass eine exakte Zeitmessung nicht durchgeführt wurde.

Ergebnisse mit großen Datensätzen der Kleinmechanik:

Neben den Testrechnungen mit den relativ kleinen Datensätzen der Kleinmechanik wurden auch Testrechnungen mit großen Datensätzen durchgeführt. Hierbei wurden Datensätze zusammengestellt welche innerhalb von vier Wochen zu bearbeiten waren. Die relvanten Informationen zu den Datensätzen sind in Tabelle 8.11 zusammengefasst.

Die Ergebnisse sind in Tabelle 8.12 aufgelistet. Wir erkennen, dass sich die guten Ergebnisse der Heuristik bei den kleinen Datensätzen (vgl. 8.3.2) auch auf die großen

Datensätze überträgt. In fast allen Testrechnungen lag die Anzahl der vergebenen Jobs deutlich unter 10%, oftmals sogar unter 5%. Im Vergleich mit der derzeitigen Maschinenbelegungsplanung bei der bis zu 30% aller Jobs vergeben werden, ist dies ein deutlicher Fortschritt. Die Anzahl der Vergaben ist offensichtlich aber auch vom Datensatz und den Rahmenbedingungen abhängig. So sehen wir, dass bzgl Maschinenparkszenario "Kleinmechanik02" und einer maximal zulässigen Verspätung von 0 Tagen, die Anzahl der Vergaben auf knapp 18% ansteigt. Wir sehen aber auch, dass schon eine maximal zulässige Verspätung von nur einem Tag eine deutliche Reduktion der Vergaben bewirkt.

Ergebnisse der Großmechanik:
Zur Vervollständigung der Ergebnisse wurden auch Testrechnungen mit Datensätzen der Großmechanik durchgeführt. Diese umfassen Datensätzen mit 76 bis 285 Jobs, welche innerhalb einer Woche durchzuführen waren. Auf Grund der hervorragenden Ergebnisse der Heuristik bzgl dieser Datensätze wurde auf eine weitere Optimierung mit Cplex verzichtet. Die Ergebnisse sind in Tabelle 8.13 aufgelistet. Die Datensätze bauen aufeinander auf, d.h. die Jobs in "GM-100" sind in "GM-200" enthalten. Auffällig ist vor allem bei den Datensätzen "GM-200" und "GM-300", dass eine Erhöhung der maximal zulässigen Verspätung von 0 Tagen auf einen Tag zu einer Reduktion der Vergaben führt, eine weitere Erhöhung der maximal zulässigen Verspätung jedoch keine Auswirkungen zeigt. Dadurch könnte man vermuten, dass es Jobs im Datensatz gibt, welche "schlecht" geplant sind, d.h. deren frühester Start- und spätester Endtermin zu dicht beieinander liegen. Ansonsten bleibt noch zu bemerken, dass - im Gegensatz zur Kleinmechanik - bei diesen Datensätzen die Summe aus Vergaben und Verspätungen stets gleich ist. Dies ist ebenfalls auf den Datensatz zurückzuführen.

8.4 Schlussfolgerungen

Aus den vorgestellten Ergebnissen lassen sich eine Reihe von Schlussfolgerungen ziehen. Diese beziehen sich sowohl auf die algorithmische Vorgehensweise zur Optimierung der Maschinenbelegung in der mechanischen Fertigung des Projektpartners, als auch auf die veränderbaren Rahmenbedingungen.

Algorithmische Vorgehensweise:
Die durchgeführten Testrechnungen haben gezeigt, dass die Heuristik bereits recht gute Ergebnisse erzielt. Eine anschließende Optimierung mit einem exakten Algorithmus liefert nur in wenigen Fälle bessere Ergebnisse, welche eher marginal sind. Frau Vodenina hatte in ihrer Diplomarbeit gezeigt (vgl. [12]), dass mit Hilfe eines genetischen Algorithmus, der auf der Heuristik basiert, die Ergebnisse dieser Heuristik unter Umständen ebenfalls leicht verbessert werden können. Bezieht man die Kostenseite für den exakten Algorithmus mit in die Überlegungen ein, so empfiehlt sich der Einsatz der Heuristik, welcher gegebenenfalls durch den genetischen Algorithmus abgerundet wird.

Einfluss der veränderbaren Rahmenbedingungen:
Die vorgestellten Ergebnisse zeigen sehr deutlich den Einfluss gewisser Rahmenbedingungen auf die Lösungsgüte des gestellten Problems. So lässt sich folgendes festhalten:

- Die maximal zulässige Verspätung:
  Betrachten wir die Ergebnisse nochmals bzgl. der maximal zulässigen Verspätung, so stellen wir fest, dass bereits eine zulässige Verspätung von nur einem Tag deutlich bessere Maschinenbelegungspläne liefert. Daher sollte auf der Werft dieser Gedanke weiter verfolgt werden, um Kosten für die Vergabe von Jobs zu verringern.

- Maschinenparkveränderungen:
  Weiterhin haben die Ergebnisse gezeigt, dass die Maschinengruppe 'BOHR' einen Engpass im Maschinenpark der Kleinmechanik darstellt. Wurde in dieser Maschinengruppe die Schichtlänge halbiert, so hatte dies sehr negative Auswirkungen auf die erzielbaren Ergebnisse. Wurde dagegen eine zweite Maschine zu dieser Gruppe hinzugefügt, so konnten die negativen Auswirkungen der halbierten Schichtlänge abgefangen werden und sogar bessere Ergebnisse ermittelt werden.

Datensatz	Maschinenpark	max. Verspätung	Lücke (%)	Zeit	Lücke*
W23KM-100	Kleinmechanik01	0 Tage	206,90	> 10h	1,14
W23KM-200	Kleinmechanik01	0 Tage	472,41	> 10h	3,11
W23KM-300	Kleinmechanik01	0 Tage	737,93	> 10h	10,96
W23KM-400	Kleinmechanik01	0 Tage	1010,34	> 10h	10,65
W23KM-500	Kleinmechanik01	0 Tage	1265,52	> 10h	15,79
W23KM-100	Kleinmechanik01	3 Tage	0,98	3s	0,04
W23KM-200	Kleinmechanik01	3 Tage	5,14	> 10h	0,20
W23KM-300	Kleinmechanik01	3 Tage	737,93	> 10h	0,95
W23KM-400	Kleinmechanik01	3 Tage	1010,34	> 10h	1,34
W23KM-500	Kleinmechanik01	3 Tage	1265,52	> 10h	2,22
W23KM-100	Kleinmechanik02	0 Tage	206,90	> 10h	14,10
W23KM-200	Kleinmechanik02	0 Tage	453,33	> 10h	10,67
W23KM-300	Kleinmechanik02	0 Tage	737,93	> 10h	16,83
W23KM-400	Kleinmechanik02	0 Tage	1010,34	> 10h	15,00
W23KM-500	Kleinmechanik02	0 Tage	1265,52	> 10h	20,00
W23KM-100	Kleinmechanik02	3 Tage	1,08	> 10h	1,15
W23KM-200	Kleinmechanik02	3 Tage	6,27	> 10h	1,26
W23KM-300	Kleinmechanik02	3 Tage	737,93	> 10h	3,16
W23KM-400	Kleinmechanik02	3 Tage	1010,34	> 10h	3,43
W23KM-500	Kleinmechanik02	3 Tage	1220,00	> 10h	5,02
W23KM-100	Kleinmechanik03	0 Tage	0,00	3s	0,00
W23KM-200	Kleinmechanik03	0 Tage	453,33	> 10h	1,22
W23KM-300	Kleinmechanik03	0 Tage	710,00	> 10h	8,97
W23KM-400	Kleinmechanik03	0 Tage	1010,34	> 10h	8,78
W23KM-500	Kleinmechanik03	0 Tage	1265,52	> 10h	13,47
W23KM-100	Kleinmechanik03	3 Tage	0,98	10s	0,00
W23KM-200	Kleinmechanik03	3 Tage	2,14	> 10h	0,03
W23KM-300	Kleinmechanik03	3 Tage	5,85	> 10h	0,23
W23KM-400	Kleinmechanik03	3 Tage	973,33	> 10h	0,57
W23KM-500	Kleinmechanik03	3 Tage	1220,00	> 10h	1,40

Tabelle 8.8: Übersicht über das verbleibende Optimierungspotential bei Lösung ohne Kenntnisse aus der Startheuristik. In der Spalte 'Lücke*' ist das verbleibende Optimierungspotential bei Lösung mit Kenntnissen aus der Startheuristik eingetragen.

Datensatz	Maschinenpark	max. Verspätung	Lücke (%)	Zeit
W23KM-100	Kleinmechanik01	0 Tage	0,00	1s
W23KM-200	Kleinmechanik01	0 Tage	7,12	> 10h
W23KM-300	Kleinmechanik01	0 Tage	5,49	> 10h
W23KM-400	Kleinmechanik01	0 Tage	4,45	> 10h
W23KM-500	Kleinmechanik01	0 Tage	7,91	> 10h
W23KM-100	Kleinmechanik01	3 Tage	0,14	1s
W23KM-200	Kleinmechanik01	3 Tage	0,66	3s
W23KM-300	Kleinmechanik01	3 Tage	0,61	13s
W23KM-400	Kleinmechanik01	3 Tage	0,58	34s
W23KM-500	Kleinmechanik01	3 Tage	1,63	> 10h
W23KM-100	Kleinmechanik02	0 Tage	12,36	> 10h
W23KM-200	Kleinmechanik02	0 Tage	14,21	> 10h
W23KM-300	Kleinmechanik02	0 Tage	12,08	> 10h
W23KM-400	Kleinmechanik02	0 Tage	9,06	> 10h
W23KM-500	Kleinmechanik02	0 Tage	11,29	> 10h
W23KM-100	Kleinmechanik02	3 Tage	1,70	> 10h
W23KM-200	Kleinmechanik02	3 Tage	2,86	> 10h
W23KM-300	Kleinmechanik02	3 Tage	2,57	> 10h
W23KM-400	Kleinmechanik02	3 Tage	2,01	> 10h
W23KM-500	Kleinmechanik02	3 Tage	5,18	> 10h
W23KM-100	Kleinmechanik03	0 Tage	0,00	0s
W23KM-200	Kleinmechanik03	0 Tage	0,00	3s
W23KM-300	Kleinmechanik03	0 Tage	0,60	13s
W23KM-400	Kleinmechanik03	0 Tage	0,93	34s
W23KM-500	Kleinmechanik03	0 Tage	4,63	> 10h
W23KM-100	Kleinmechanik03	3 Tage	0,00	1s
W23KM-200	Kleinmechanik03	3 Tage	0,00	7s
W23KM-300	Kleinmechanik03	3 Tage	0,12	30s
W23KM-400	Kleinmechanik03	3 Tage	0,55	87s
W23KM-500	Kleinmechanik03	3 Tage	0,98	223s

Tabelle 8.9: Übersicht über das verbleibende Optimierungspotential bei Lösung des Problems mit variierter Zielfunktion und Kenntnissen aus der Startheuristik.

Datensatz	Maschinenpark	max. Verspätung	Lücke (%)	Zeit
W23KM-100	Kleinmechanik01	0 Tage	0,00	1s
W23KM-200	Kleinmechanik01	0 Tage	238,04	> 10h
W23KM-300	Kleinmechanik01	0 Tage	387,94	> 10h
W23KM-400	Kleinmechanik01	0 Tage	566,31	> 10h
W23KM-500	Kleinmechanik01	0 Tage	901,35	> 10h
W23KM-100	Kleinmechanik01	3 Tage	0,00	1s
W23KM-200	Kleinmechanik01	3 Tage	4,77	> 10h
W23KM-300	Kleinmechanik01	3 Tage	387.70	> 10h
W23KM-400	Kleinmechanik01	3 Tage	573,06	> 10h
W23KM-500	Kleinmechanik01	3 Tage	901,35	> 10h
W23KM-100	Kleinmechanik02	0 Tage	0,00	9s
W23KM-200	Kleinmechanik02	0 Tage	238,04	> 10h
W23KM-300	Kleinmechanik02	0 Tage	387,94	> 10h
W23KM-400	Kleinmechanik02	0 Tage	573,06	> 10h
W23KM-500	Kleinmechanik02	0 Tage	901,35	> 10h
W23KM-100	Kleinmechanik02	3 Tage	0,28	2s
W23KM-200	Kleinmechanik02	3 Tage	238,04	> 10h
W23KM-300	Kleinmechanik02	3 Tage	387,94	> 10h
W23KM-400	Kleinmechanik02	3 Tage	573,06	> 10h
W23KM-500	Kleinmechanik02	3 Tage	901,35	> 10h
W23KM-100	Kleinmechanik03	0 Tage	0,00	2s
W23KM-200	Kleinmechanik03	0 Tage	212,07	> 10h
W23KM-300	Kleinmechanik03	0 Tage	359,25	> 10h
W23KM-400	Kleinmechanik03	0 Tage	533,48	> 10h
W23KM-500	Kleinmechanik03	0 Tage	840,48	> 10h
W23KM-100	Kleinmechanik03	3 Tage	0,04	2s
W23KM-200	Kleinmechanik03	3 Tage	1,06	> 10h
W23KM-300	Kleinmechanik03	3 Tage	359,25	> 10h
W23KM-400	Kleinmechanik03	3 Tage	533,48	> 10h
W23KM-500	Kleinmechanik03	3 Tage	840,48	> 10h

Tabelle 8.10: Übersicht über das verbleibende Optimierungspotential bei Lösung des Problems mit variierter Zielfunktion und ohne Kenntnisse aus der Startheuristik.

Datensatz	Anzahl Jobs	Anzahl Operationen
W22-W25	1222	1931
W23-W26	1424	2214

Tabelle 8.11: Informationen über die Anzahl der Jobs und Operationen in den großen Datensätzen der Kleinmechanik.

Datensatz	Maschinenpark	max. Versp.	# Vergaben	#Versp.	Vergaben (%)
W22-W25	Kleinmechanik01	0 (Tage)	100	0	8,18
W22-W25	Kleinmechanik01	1 (Tage)	29	110	2,37
W22-W25	Kleinmechanik01	2 (Tage)	21	124	1,72
W22-W25	Kleinmechanik01	3 (Tage)	15	133	1,23
W22-W25	Kleinmechanik02	0 (Tage)	134	0	10,97
W22-W25	Kleinmechanik02	1 (Tage)	44	152	3,60
W22-W25	Kleinmechanik02	2 (Tage)	24	181	1,91
W22-W25	Kleinmechanik02	3 (Tage)	14	209	1,15
W22-W25	Kleinmechanik03	0 (Tage)	90	0	7,36
W22-W25	Kleinmechanik03	1 (Tage)	29	87	2,37
W22-W25	Kleinmechanik03	2 (Tage)	16	109	1,31
W22-W25	Kleinmechanik03	3 (Tage)	14	115	1,15
W23-W26	Kleinmechanik01	0 (Tage)	201	0	14,12
W23-W26	Kleinmechanik01	1 (Tage)	92	172	6,46
W23-W26	Kleinmechanik01	2 (Tage)	36	254	2,53
W23-W26	Kleinmechanik01	3 (Tage)	18	293	1,26
W23-W26	Kleinmechanik02	0 (Tage)	253	0	17,77
W23-W26	Kleinmechanik02	1 (Tage)	150	161	10,53
W23-W26	Kleinmechanik02	2 (Tage)	75	271	5,27
W23-W26	Kleinmechanik02	3 (Tage)	45	325	3,16
W23-W26	Kleinmechanik03	0 (Tage)	185	0	12,99
W23-W26	Kleinmechanik03	1 (Tage)	91	149	6,39
W23-W26	Kleinmechanik03	2 (Tage)	31	239	2,18
W23-W26	Kleinmechanik03	3 (Tage)	17	264	1,19

Tabelle 8.12: Übersicht über die Anzahl der Vergaben und Verspätungen bei den Ergebnissen zu den großen Datensätzen der Kleinmechanik.

Datensatz	# Jobs	max. Versp.	# Vergaben	#Versp.	Vergaben (%)
GM-100	76	0 (Tage)	1	0	1,32
GM-100	76	1 (Tage)	0	1	0,00
GM-100	76	2 (Tage)	0	1	0,00
GM-100	76	3 (Tage)	0	1	0,00
GM-200	151	0 (Tage)	4	0	2,65
GM-200	151	1 (Tage)	2	2	1,32
GM-200	151	2 (Tage)	2	2	1,32
GM-200	151	3 (Tage)	2	2	1,32
GM-300	216	0 (Tage)	7	0	3,24
GM-300	216	1 (Tage)	5	2	2,31
GM-300	216	2 (Tage)	5	2	2,31
GM-300	216	3 (Tage)	5	2	2,31
GM-400	285	0 (Tage)	11	0	3,86
GM-400	285	1 (Tage)	8	3	2,81
GM-400	285	2 (Tage)	7	4	2,46
GM-400	285	3 (Tage)	6	5	2,11

Tabelle 8.13: Übersicht über die Anzahl der Vergaben und Verspätungen bei den Ergebnissen zu den Datensätzen der Großmechanik.

Anhang A

Mathematische Bezeichnungen und Modelle

A.1 Mengen

J	Menge aller Jobs
O	Menge aller Operationen
M	Menge aller Maschinen
M_g^O	Menge aller Operationen, die auf Maschinengruppe g durchgeführt werden müssen
M_g	Menge aller Maschinen von Maschinengruppe g zzgl. M_g^O

A.2 Daten

b_i	Bearbeitungszeit von Operation i
r_{ij}	Rüstzeit zwischen Operation i und j
T_l^S	frühester Starttermin von Job l
T_l^E	spätester Fertigstellungstermin von Job l
S_g	Schichtlänge von Maschinengruppe g
c_l	Gewichtungsfaktor eines Jobs

A.3 Variablen

x_{ij}	$\in \{0,1\}$	Reihenfolgevariable
y_l	$\in \{0,1\}$	Fremdvergabeindikator
v_i	$\in \{0,1\}$	Schichtüberschreitungsindikator
d_i	$\in \mathbb{N}_0$	Umrechnungsvariable
t_i	≥ 0	Startzeit von Operation i
w_i	≥ 0	zusätzliche Bearbeitungszeit durch Schichtüberschreitung
z_l	≥ 0	Verspätung von Job l

A.4 Optimierungsmodell - minimaler Makespan

$$(M1-Z) \quad C_{\max} \rightarrow \text{Min !}$$

$$(M1-0) \quad C_{\max} \geq t_{n_l} + b_{n_l} + \sum_{j \in M_{g(n_l)}} r_{jn_l} \cdot x_{jn_l}^{g(n_l)} + w_{n_l} \quad \forall l \in \{1, \ldots, |J|\}$$

$$(M1-1) \quad \sum_{j \in M_g \setminus \{i\}} x_{ji}^g = 1 \quad \forall i \in M_g^O, g \in \{1, \ldots, G\}$$

$$(M1-2) \quad \sum_{j \in M_g^O \setminus \{i\}} x_{ij}^g \leq 1 \quad \forall i \in M_g, g \in \{1, \ldots, G\}$$

$$(M1-3) \quad t_i \geq t_j + b_j + \sum_{h \in M_g} r_{hj} \cdot x_{hj}^g + w_j - (1 - x_{ji}^g) \cdot M \quad \forall i, j \in M_g^O, g \in \{1, \ldots, G\}$$

$$(M1-4) \quad t_{i+1} \geq t_i + b_i + \sum_{j \in M_{g(i)} \setminus \{i\}} r_{ji} \cdot x_{ji}^{g(i)} + w_i$$
$$\forall i \in \{n_{l-1}+1, \ldots, n_l - 1\}, \forall l \in \{1, \ldots, |J|\}$$

$$(M1-5) \quad t_{n_{l-1}+1} \geq T_l^S \quad \forall l \in \{1, \ldots, |J|\}$$

$$(M1-7) \quad \frac{t_i}{24} \geq d_i \geq \frac{t_i}{24} - \frac{2399}{2400} \quad \forall i \in O$$

$$(M1-8) \quad t_i - 24 \cdot d_i + b_i + \sum_{j \in M_{g(i)}} r_{ji} \cdot x_{ji}^{g(i)} - 24 \cdot v_i \leq S_g \quad \forall i \in O$$

$$(M1-9) \quad w_i = (24 - S_{g(i)}) \cdot v_i \quad \forall i \in O$$

A.5 Optimierungsmodell - minimale Anzahl an Vergaben

$$(M2-Z) \quad \sum_{l \in J} c_l \cdot y_l \rightarrow \text{Max !}$$

$$(M2-1) \quad \sum_{j \in M_g \setminus \{i\}} x_{ji}^g = y_{l(i)} \quad \forall i \in M_g^O, g \in \{1, \ldots, G\}$$

$$(M2-2) \quad \sum_{j \in M_g^O \setminus \{i\}} x_{ij}^g \leq y_{l(i)} \quad \forall i \in M_g, g \in \{1, \ldots, G\}$$

$$(M2-3) \quad t_i \geq t_j + b_j + \sum_{h \in M_g} r_{hj} \cdot x_{hj}^g + w_j - (1 - x_{ji}^g) \cdot M \quad \forall i, j \in M_g^O, g \in \{1, \ldots, G\}$$

$$(M2-4) \quad t_{i+1} \geq t_i + b_i + \sum_{j \in M_{g(i)} \setminus \{i\}} r_{ji} \cdot x_{ji}^{g(i)} + w_i - (1 - y_{l(i)}) \cdot M$$
$$\forall i \in \{n_{l-1}+1, \ldots, n_l - 1\}, \forall l \in \{1, \ldots, |J|\}$$

$$(M2-5) \quad t_{n_{l-1}+1} \geq T_l^S \cdot y_l \quad \forall l \in \{1, \ldots, |J|\}$$

$$(M2-6) \quad T_l^E - \left(t_{n_l} + b_{n_l} + \sum_{j \in M_{g(n_l)}} r_{jn_l} \cdot x_{jn_l}^{g(n_l)} + w_{n_l} \right) + (1 - y_l) \cdot M \geq 0$$
$$\forall l \in \{1, \ldots, |J|\}$$

$$(M2-7) \quad \frac{t_i}{24} \geq d_i \geq \frac{t_i}{24} - \frac{2399}{2400} \quad \forall i \in O$$

$$(M2-8) \quad t_i - 24 \cdot d_i + b_i + \sum_{j \in M_{g(i)}} r_{ji} \cdot x_{ji}^{g(i)} - 24 \cdot v_i \leq S_g \quad \forall i \in O$$

$$(M2-9) \quad w_i = (24 - S_{g(i)}) \cdot v_i \quad \forall i \in O$$

$$(M2-10) \quad v_i \leq y_{l(i)} \quad \forall i \in O$$

A.6 Optimierungsmodell - minimale Anzahl an Vergaben und Verspätungen

$$(M2-Z) \quad \sum_{l\in J} c_l \cdot (3,5 \cdot y_l - z_l) \rightarrow \text{Max !}$$

$$(M2-1) \quad \sum_{j\in M_g \setminus \{i\}} x_{ji}^g = y_{l(i)} \quad \forall i \in M_g^O, g \in \{1,\ldots,G\}$$

$$(M2-2) \quad \sum_{j\in M_g^O \setminus \{i\}} x_{ij}^g \leq y_{l(i)} \quad \forall i \in M_g, g \in \{1,\ldots,G\}$$

$$(M2-3) \quad t_i \geq t_j + b_j + \sum_{h\in M_g} r_{hj} \cdot x_{hj}^g + w_j - (1 - x_{ji}^g) \cdot M \quad \forall i,j \in M_g^O, g \in \{1,\ldots,G\}$$

$$(M2-4) \quad t_{i+1} \geq t_i + b_i + \sum_{j\in M_{g(i)} \setminus \{i\}} r_{ji} \cdot x_{ji}^{g(i)} + w_i - (1 - y_{l(i)}) \cdot M$$
$$\forall i \in \{n_{l-1}+1,\ldots,n_l - 1\}, \forall l \in \{1,\ldots,|J|\}$$

$$(M2-5) \quad t_{n_{l-1}+1} \geq T_l^S \cdot y_l \quad \forall l \in \{1,\ldots,|J|\}$$

$$(M2-6) \quad T_l^E + 24 \cdot z_l - \left(t_{n_l} + b_{n_l} + \sum_{j\in M_{g(n_l)}} r_{jn_l} \cdot x_{jn_l}^{g(n_l)} + w_{n_l} \right) + (1 - y_l) \cdot M \geq 0$$
$$\forall l \in \{1,\ldots,|J|\}$$

$$(M2-7) \quad \tfrac{t_i}{24} \geq d_i \geq \tfrac{t_i}{24} - \tfrac{2399}{2400} \quad \forall i \in O$$

$$(M2-8) \quad t_i - 24 \cdot d_i + b_i + \sum_{j\in M_{g(i)}} r_{ji} \cdot x_{ji}^{g(i)} - 24 \cdot v_i \leq S_g \quad \forall i \in O$$

$$(M2-9) \quad w_i = (24 - S_{g(i)}) \cdot v_i \quad \forall i \in O$$

$$(M2-10) \quad v_i \leq y_{l(i)} \quad \forall i \in O$$

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] S. C. Graves, A. H. G. Rinooy Kan, P. H. Zipkin: Handbooks in operations research and management science (Vol 4), North Holland 1993

[2] P. Brucker: Scheduling algorithms, Springer Verlag, 2001

[3] P. Chretienne, E. G. Coffman, J. K. Lenstra, Z. Liu: Scheduling theory and its application, Wiley 1997

[4] E. Aarts, J. K. Lenstra: Local search in combinatorial optimization, Wiley, 1997

[5] T. E. Morton, D. W. Pentico: Heuristic scheduling systems, Wiley, 1993

[6] D. E. Brown, W. T. Scherer: Intelligent scheduling systems, Kluwer 1997

[7] M. Pinedo: Scheduling: theory, algorithms and systems, Prentice Hall, 2002

[8] V. T'kindt, J.-C. Bilaut: Multicriteria scheduling, Springer Verlag, 2002

[9] M. dell'Amico, F. Maffioli, S. Martello: Annotated bibliographies in combinatorial optimization, Wiley 1997

[10] J. B. Chambers, J. W. Barnes: Flexible Job Shop Scheduling by Tabu Search, http://net.cs.utexas.edu/users/jbc/home/papers/papers.html, 1996

[11] J. B. Chambers, J. W. Barnes: Reactive Search for Flexible Job Shop Scheduling, http://net.cs.utexas.edu/users/jbc/home/papers/papers.html, 1996

[12] A. Vodenina: Ein genetischer Algorithmus zur Lösung eines Scheduling-Problems, Universität Duisburg-Essen Campus Duisburg, Diplomarbeit 2004

[13] A. Guinet, M. Legrand: Reduction of job-shop problems to flow-shop problems with precedence constraints, European Journal of Operational Research 109 (1998) 96 - 110

[14] A. Guinet: Efficiency of reductions of job-shop to flow shop problems, European Journal of Operational Research 125 (2000) 469 - 485

[15] D. Bertsimas, J. Sethuraman: From fluid relaxations to practical algorithms for job shop scheduling: the makespan objective, Math. Program., Ser. A 92: 61 - 102 (2002)

[16] R. Borndörfer, M. Grötschel, A. Löbel: Scheduling Duties by Adaptive Column Generation, Zuse-Institut Berlin (ZIB), ZIB-Report 01-02 (January 2001)

[17] S. O. Krumke, J. Rambau, L. M. Torres: Real-Time Dispatching of Guided and Unguided Automobile Service Units with Soft Time Windows,Zuse-Institut Berlin (ZIB), ZIB-Report 01-22 (September 2001)

[18] P. Martin, D. B. Shmoys: A New Approach to Computing Optimal Schedules for the Job-Shop Scheduling Problem, http://www.cornell.edu/ shmoys/ipcopaper.ps, 1996

[19] Z.-L. Chen, W. B. Powell: A Column Generation Based Decomposition Algorithm for a Parallel Machine Just-In-Time Scheduling Problem, http://www.castlelab.princeton.edu/Resources/pjit.pdf, 1999

[20] A. S. Jain, S. Meeran: Deterministic Job-Shop Scheduling: Past, Present and Future, http://www.dundee.ac.uk/ãsjain/papers/ejor1.ps, 1998

[21] C. Schwindt: Generation of Resource-Constraint Project Scheduling Problems with Minimal and Maximal Time Lags, Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe, WIOR-489, 1996

[22] R. H. Möhring: Combinatorial Optimization and Graph Algorithms, Biennial Report 1999/2000, Institut für Mathematik, Technische Universität Berlin

[23] G. Törner: Einführung in die Scheduling-Theorie, Vorlesungsskript Wintersemester 2001/2002, Universität Duisburg-Essen, Campus Duisburg

[24] Dudenredaktion 1988: Duden Informatik - Ein Sachlexikon für Studium und Praxis, Mannheim: Dudenverlag