Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Entwicklung eines Optimierungsmodells mit Berücksichtigung der Arbeitsgangreihenfolge für engpassorientierte Produktionsplanung

Bachelorthesis im Bereich

Produktionstechnik und -management

Autor:	Simon Schütt Maike-Harder-Weg 57 22399 Hamburg
Matrikel-Nr.:	1859897
Abgabetermin:	26. August 2010
Gutachter:	Prof. Dr.-Ing. Jochen Kreutzfeldt
Zweitgutachter:	M. Eng. Dipl. Wirt.-Ing. (FH) Johannes Hinckeldeyn
Abgabedatum:	26.August 2010
Adresse:	Hochschule für Angewandte Wissenschaften Fakultät Technik und Informatik Department Maschinenbau und Produktion Berliner Tor 21 20099 Hamburg

I. Inhaltsverzeichnis

I. Inhaltsverzeichnis I

II. Formelzeichen und Abkürzungen II

III. Abbildungsverzeichnis IV

IV. Tabellenverzeichnis V

1 Einleitung 1

1.1 Forschungsprojekt DePlaVis an der HAW Hamburg 1

1.2 Zielsetzung und Aufbau der Arbeit 2

2 Stand der Forschung zur Engpassplanung und -bewertung anhand von Durchsatzkennlinien 4

2.1 Vorstellung der Durchsatzkennlinientheorie 4

2.1.1 Beschreibung des DePlaVis-Algorithmus nach Kreutzfeldt 4

2.1.2 Beschreibung des DePlaVis-Algorithmus 2.0 12

2.2 Probleme durch nicht Berücksichtigung der Arbeitsgangreihenfolge 17

3 Einführung in das Operations Research 20

3.1 Grundlagen des Operations Research 20

3.2 Optimierungsmethoden des Operations Research 22

3.2.1 Lineare Optimierung 23

3.2.2 Ganzzahlige und kombinatorische Optimierung 33

3.3 Einführung in die Maschinenbelegungsplanung 35

3.3.1 Grundlagen der Maschinenbelegungsplanung 35

3.3.2 Notation und Resultate der Maschinenbelegungsplanung 38

4 Entwicklung eines Lösungsansatzes zur Berücksichtigung der Arbeitsgangreihenfolge 42

4.1 Allgemeine Überlegungen zur Arbeitsgangreihenfolgebildung 42

4.2 Formulierung des neuen Algorithmus unter Berücksichtigung der Arbeitsgangreihenfolge 44

4.3 Anwendungsdemonstration des Algorithmus anhand eines Fallbeispiels 51

4.4 Implementierungsvorschlag des neuen Algorithmus 58

4.5 Erstellung eines Demonstrators in Matlab 60

4.6 Bewertung der Ergebnisse 65

5 Zusammenfassung und Ausblick 68

6 Literaturverzeichnis 70

Anhang A 74

Anhang B 76

II. Formelzeichen und Abkürzungen

AF_i	Auftrag i
AS_i	Arbeitsstation i
a_j	Bedarf j
BQ_i	Belastungsquotient i
b	Kapazitätsvektor
C_j	Fertigstellungszeitpunkt
c	Vektor der Zielkoeffizienten
F	Flow-Shop-Problem
$FiFo$	First-in-First-out
h	Stunde
IP	Integer-Programming
J	Job-Shop-Problem
j	Job / Auftrag
K_{AS_k}	Kapazität auf der Arbeitsstation k
L_j	Verspätung
LP	Linear Programming
l_j	Untergrenze
M_i	Maschine i
MIP	Mixed-Integer-Programming
O	Open-Shop-Problem
P	Produkt
p_{ij}	Bearbeitungsdauer
R	Widerstand
ROI	Return on invest
r_j	Bereitstellungstermin
T_j	Terminüberschreitung

u_j	Obergrenze
w_j	Gewicht j
ZE	Zeiteinheiten
α	Maschinenkonfiguration
β	Jobeigenschaften
γ	Zielfunktion

III. Abbildungsverzeichnis

Abbildung 2.1: Auswirkung eines Engpasses auf das Zielsystem der Produktionslogistik 5
Abbildung 2.2: Darstellung der „Theory of Constraints“ als Kreislauf 6
Abbildung 2.3: Darstellung des Trichtermodells 7
Abbildung 2.4: eines Produktionssystems 8
Abbildung 2.5: Analogie elektrisches Netzwerk zu einem Fertigungsnetzwerk 11
Abbildung 2.6: Darstellung einer Durchsatzkennlinie 11
Abbildung 2.7: Übersicht des Fertigungsnetzwerks mit Arbeitsstationen inklusive der Kapazität und der Zuordnung der Aufträge 14
Abbildung 2.8: Fertigungsnetzwerk nach dem Eliminieren der berücksichtigten Aufträge 17
Abbildung 2.9: Engpässe aufgrund disproportionaler Teilkapazitäten 18
Abbildung 3.1: Modelle strukturiert nach ihrem Zweck 22
Abbildung 3.2: Aufstellen eines linearen Optimierungsproblems 24
Abbildung 3.3: Grafische Lösung eines 2-dimensionalen LP-Modells 32
Abbildung 3.4: Konvexe und nichtkonvexe Bereiche 33
Abbildung 3.5: Eine Maschinenbelegungsplanung mit vier Aufträgen und drei Maschinen 37
Abbildung 4.1: Branch-and-Bound-Verfahren 46
Abbildung 4.2: Bereits zugewiesene und noch nicht zugewiesene Aufträge 48
Abbildung 4.3: Vorgehensweise des Algorithmus zur Minimierung der gesamt Terminüberschreitung 50
Abbildung 4.4: Übersicht des Fertigungsnetzwerks mit Arbeitsstationen inklusive der Kapazität und der Zuordnung der Aufträge 52
Abbildung 4.5: Darstellung der ersten Ebene des Suchbaumes und der sich ergebenden Knoten 55
Abbildung 4.6: Branch-and-Bound für die Ermittlung der optimalen Reihenfolge 56
Abbildung 4.7: Maschinenbelegungsplan für das gesamte Fertigungssystem 58
Abbildung 4.8: Implementierungsvorschlag des neuen Algorithmus 59

IV. Tabellenverzeichnis

Tabelle 2.1: Belastungsquotienten der einzelnen Arbeitsstationen 15
Tabelle 4.1: Belastungsquotienten der einzelnen Arbeitsstationen 53
Tabelle 4.2: Eingangsdaten zur Ermittlung der optimalen Reihenfolge auf Arbeitsstation 3 ..53
Tabelle 4.3: Resultierende Auftragsliste für den Engpass.....57
Tabelle 4.4: Datenermittlung der Rechenzeit des Demonstrators 63
Tabelle 4.5: Rechenzeit für die Reihenfolgebildung auf einer Maschine 64

1 Einleitung

In vielen Produktionsumgebungen beeinflussen Engpässe aufgrund ihrer vielfältigen Auswirkungen die Leistung der Fertigung und die logistischen Zielgrößen eines Unternehmens. Sie führen zu Warteschlangen vor den Bearbeitungsstationen und somit zu hohen Beständen, Überlastungen der Arbeitssysteme, streuenden Durchlaufzeiten, Ineffizienzen und Durchsatzverlusten. Auf der anderen Seite bietet jedoch die Kenntnis über die eigene Engpasssituation immer auch Möglichkeiten der Optimierung. Durch gezielte Verbesserungen an den Engpässen kann unter Umständen die gesamte Systemleistung signifikant beeinflusst werden. Aus diesem Grund sollte der Identifizierung und der Bewertung der Optimierungspotenziale von Engpässen eine besondere Rolle zugetragen werden [vgl. GOL08].

1.1 Forschungsprojekt DePlaVis an der HAW Hamburg

Das Forschungsprojekt DePlaVis wird an der Hochschule für Angewandte Wissenschaften Hamburg, kurz HAW Hamburg, im Department für Maschinenbau und Produktion durchgeführt. Die Abkürzung DePlaVis steht für „Durchsatzsteigerung im Anlagenbau und -betrieb durch engpassorientierte Planung und Steuerung“. Das Forschungsprojekt ist unterstütz durch das Bundesministerium für Bildung und Forschung (BMBF).

Ziel des Forschungsprojektes ist die Unterstützung des deutschen Maschinen- und Anlagenbaus [vgl. KRE07 S.2 f.]. Mit Hilfe des DePlaVis-Algorithmus ist es möglich innerhalb eines Produktionssystems Kapazitätsengpässe festzustellen und diese zu bewerten. Dieses geschieht mit Hilfe des DePlaVis-Algorithmus, welcher auf der Berechnung der sogenannten Durchsatzkennlinie basiert [vgl. KRE95 S.70 f.]. Dabei wird der Durchsatz des Auftragsstromes als Funktion über die Einlastung aufgetragen. Anhand dieser Darstellung können der nicht durchgesetzte Auftragsbestand und die Wirkung auf den Engpass sowohl auf die vor- als auch auf die nachgelagerten Kapazitäten quantifiziert werden. Aus der Summe aller Durchsatzkennlinien wird eine Kennlinie für die gesamte Produktion generiert. Somit kann die Bedeutung und Wirkung der Engpassressource für das gesamte Produktionssystem berechnet und visualisiert werden.

Die Bewertung des Engpasses kann zusätzlich über zwei Kennzahlen erfolgen [vgl. KRE95 S.74 ff.]. Der Bestand an nicht durchsetzbaren Aufträgen kann ermittelt werden und bildet das sogenannte Durchsatzpotential. Hierbei handelt es sich um das Arbeitsvolumen, welches in einer Periode nicht mehr durchgesetzt werden kann und somit als Bestand zurück bleibt. Desweiteren kann aus dem direkten und indirekten Durchsatz der Grenzdurchsatz ermittelt werden [vgl. SCH08 S.48 ff.]. Somit kann die Auswirkung des Engpasses auf vor- und nachgelagerte Prozessschritte quantifiziert werden. Zudem können Rückschlüsse darüber gewonnen werden, welche Durchsatzsteigerung eine Erhöhung der Engpasskapazität um eine Einheit mit sich bringt [vgl. SCH09 S.1 f.].

Im Rahmen früherer Arbeiten wurden der Algorithmus und seine Funktionen in einer Software implementiert [vgl. ALT08 S.48 ff.]. Es sind jedoch noch nicht alle reellen Gegebenheiten der Praxis implementiert. So fehlt beispielweise die Berücksichtigung der Arbeitsgangreihenfolgeplanung. Die Durchsetzbarkeit eines Auftrages wird bisher von der Überlastung der Durchsatzschranke abhängig gemacht. So entstehen Berechnungsfehler, die zu einem zu hohen Durchsatz führen.

1.2 Zielsetzung und Aufbau der Arbeit

Ziele dieser Arbeit sind die Integration der Arbeitsreihenfolge in ein durchsatzsteigerndes Optimierungsmodel. Dabei gilt es einen Algorithmus zu formulieren, der die freizugebenden Aufträge basierend auf den Erkenntnissen der Durchsatzkennlinientheorie generiert. Weiterhin soll eine ein Demonstrator programmiert werden, der die Reihenfolgebildung anhand von diskreten Arbeitsinhalten übernimmt. Der neu entwickelte Algorithmus soll in den bestehenden DePlaVis-Algorithmus implementiert werden.

In Abschnitt Zwei der Arbeit erfolgt eine Vorstellung der Durchsatzkennlinientheorie und des DePlaVis-Algorithmus. Zunächst wird die Druchsatzkennlinientheorie nach Kreutzfeldt vorgestellt. Danach wird der DePlaVis-Algorithmus 2.0 beschrieben und anhand eines Fallbeispiels veranschaulicht. Die Probleme der nicht Berücksichtigung der Arbeitsgang-reihenfolge werden im letzten Teil des zweiten Kapitels erläutert.

Eine Einführung in das Operation Research erfolgt in Kapitel drei. Zunächst werden verschiedene Optimierungsverfahren vorgestellt. Hier soll speziell auf die lineare und ganzzahlig-kombinatorische Optimierung eingegangen werden. Die lineare Optimierung bildet die Grundlage der Optimierungsprobleme. Die ganzzahlig-kombinatorische ist ein Sonderfall der linearen Optimierung. Im Anschluss daran gilt es eine Einführung in die Maschinenbelegungsplanung zu geben.

Der Lösungsvorschlag wird in Kapitel vier ausgearbeitet. Dabei werden zuerst allgemeine Überlegungen zur Arbeitsreihenfolgebildung aufgestellt sowie Begriffe und Grundlagen erläutert. Auf dieser Basis wird ein Lösungsvorschlag basierend auf einem Ein-Maschinen-Problem abgeleitet. Dieser wird als Optimierungsmodel formuliert. Ziel ist es eine Auftragsreihenfolgeplanung auf dem identifizierten Engpass vorzunehmen unter der Bedingung des maximalen Durchsatzes in einer Periode. Der verbesserte Algorithmus wird anhand eines Fallbeispiels anschließend eingehend demonstriert. Danach wird ein Implementierungsvorschlag des neuen Algorithmus mit Hilfe eines Ablaufdiagramms erzeugt. Am Ende des Kapitels wird ein in Matlab programmierter Demonstrator vorgestellt und die erarbeiteten Ergebnisse bewertet.

Abschließend erfolgt in Kapitel fünf eine Zusammenfassung der Ergebnisse der Arbeit und ein Ausblick auf weitere Themenfelder wird gegeben.

2 Stand der Forschung zur Engpassplanung und -bewertung anhand von Durchsatzkennlinien

In der Literatur sind Ansätze zum Engpassmanagement bereits seit längerem bekannt. In diesem Kapitel erfolgt deshalb eine Vorstellung grundlegender Modelle zum Thema des Engpassmanagements. Hier wird besonderes Augenmerk auf die Durchsatzkennlinien-theorie gelegt, da sie die Basis des DePlaVis-Algorithmus darstellt und als Ausgangsbasis für den zu erarbeitenden Lösungsvorschlag dient. Im Anschluss daran erfolgt eine Darstellung der Probleme bei nicht Berücksichtigung der Arbeitsgangreihenfolge.

2.1 Vorstellung der Durchsatzkennlinientheorie

Mit Hilfe der Durchsatzkennlinientheorie kann eine Engpassbewertung durchgeführt werden. Der Zusammenhang zwischen der Einlastung eines Auftragsstromes und dem Durchsatz wird dazu mittels einer sogenannten Durchsatzkennlinie visualisiert. Im nachfolgenden Kapitel wird die Theorie beschrieben.

2.1.1 Beschreibung des DePlaVis-Algorithmus nach Kreutzfeldt

Die Engpässe innerhalb eines Produktionssystems sind die beschränkenden Elemente und wirken sich auf die gesamte Systemleistung aus [vgl. KRS09 S.2 ff.]. Dies kann hinsichtlich der Erfolgsfaktoren in das Zielsystem der Produktionslogistik eingeordnet werden [vgl. WIE08 S.3 f.]. In der Abbildung 2.1 sind die Auswirkungen auf ein Produktionssystem einmal dargestellt. Das Produktionssystem verfügt über fünf, als Kreise dargestellte, Arbeits-stationen, durch die zwei Materialströme fließen. Geht man von gleichen Kapazitäten an allen Arbeitsstationen aus, so muss sich an Arbeitsstation drei der Systemengpass ergeben.

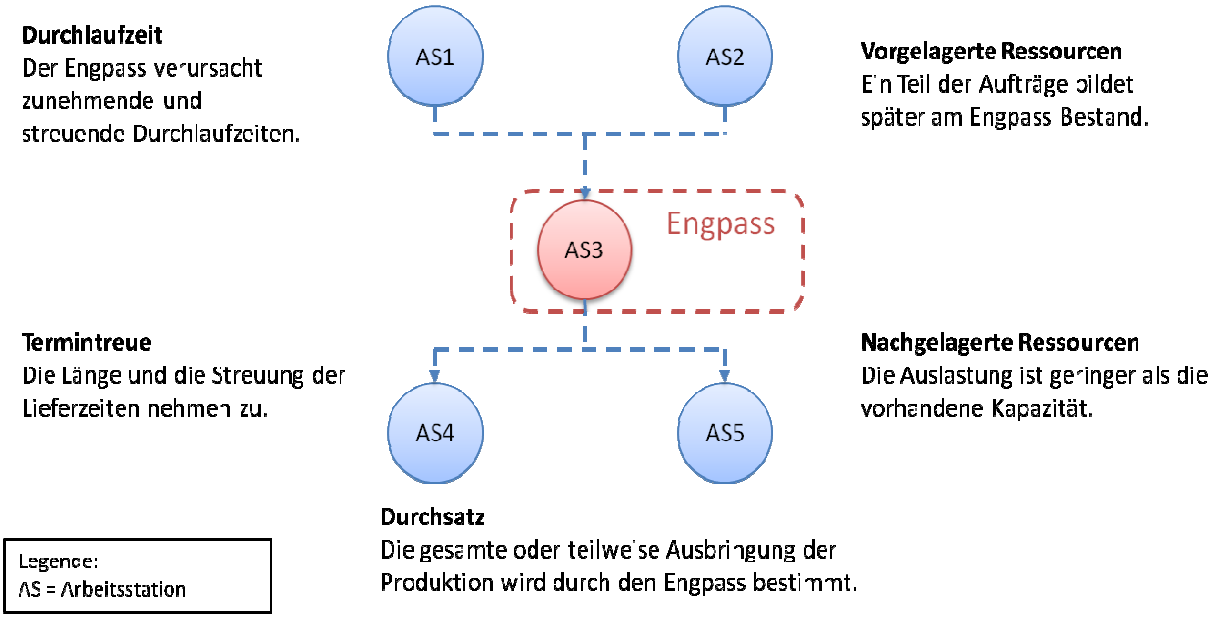

Abbildung 2.1: Auswirkung eines Engpasses auf das Zielsystem der Produktionslogistik [in Anlehnung an SCH08 S.4]

Vor Arbeitsstation drei kommt es zur Bildung von Beständen und somit zu Kapitalbindung. Jedes System, sei es organisatorisch, physikalisch, elektrisch oder ein Produktionssystem, in dem ankommende Elemente Anforderungen an Ressourcen stellen, kann man als Warteschlangen- oder Stauungssystem bezeichnen [vgl. ZIM08 S.397 ff.]. Steigende Bestände führen wiederum zu längeren und, durch das Priorisieren von Aufträgen, zu streuenden Durchlaufzeiten. Dieses kann sich negativ auf die Termintreue und im schlimmsten Fall auf die Liefertreue auswirken. Aufgrund der Engpasssituation an Arbeitsstation drei kommt es auf den dahinter liegenden Stationen, vier und fünf, zu einer Unterlast. Dies resultiert daraus, dass auf Station drei nicht alle eingeplanten Aufträge abgearbeitet werden können, die jeweils auf Station vier und fünf eingeplant sind. Zusammenfassend ist davon auszugehen, dass die Engpässe eines Produktionssystems ausschlaggebend für dessen Leistung und Durchsatz sind.

Gutenberg hat 1976 zum ersten Mal dieses Postulat verfasst [vgl. GUT76 S.163 ff.]. Ein Engpass ist als produktiver Faktor beschrieben, der nur bis zu einer bestimmten Kapazität ausgelastet werden kann. Demnach stellt dieser spezifische Faktor den schwächsten Bestandteil im Produktionssystem dar und zwingt die Unternehmen zur Planungsausrichtung auf dem Engpass. Dieses Phänomen wird als „Dominanz des Minimumfaktors“ bezeichnet [vgl. GUT76 S.164]. Als Lösung wird das „Ausgleichsgesetz zur Planung“ vorgeschlagen. Danach richtet sich die kurzfristige Planung eines Unternehmens am Engpassfaktor aus, da

dieser nicht erweitert werden kann. Langfristig erfolgt eine Anpassung des Engpassfaktors an die anderen Teilbereiche [vgl. GUT S.164].

Basierend auf den Annahmen der Dominanz von Engpässen wurde von Goldratt die „Theory of Constraints“ entwickelt [vgl. GOL08 S.5 ff.]. Sie stellt einen systematischen und kontinuierlichen Verbesserungszyklus mit fünf Schritten dar:

1. Die Systemengpässe werden bestimmt und priorisiert.
2. Es wird über die Ausnutzung der Systemengpässe entschieden.
3. Alle anderen Entscheidungen und Systembestandteile werden dem Engpass untergeordnet.
4. Der Systemengpass wird erweitert.
5. Das System wird erneut auf Engpässe überprüft. Wird ein neuer Engpass festgestellt, dann beginnt der Zyklus von Neuem.

In Abbildung 2.2 ist der Zyklus noch einmal anschaulich in einer Grafik dargestellt.

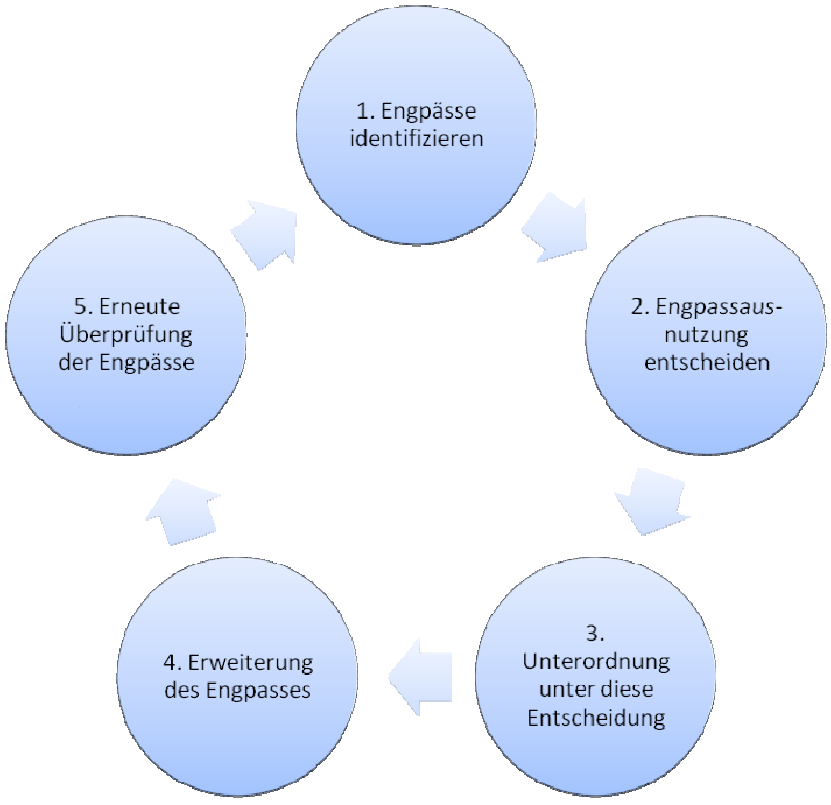

Abbildung 2.2: Darstellung der „Theory of Constraints“ als Kreislauf [Eigene Darstellung]

Trotz des erfolgreichen Einsatzes dieser Methodik wurden von Goldratt nie Details zur Engpassidentifikation veröffentlicht [vgl. HAB99 S.18 f]. Eine Methode Engpässe und ihre Auswirkungen ermitteln zu können ist eine quantitative Systembeschreibung. Neben der Simulation[1] von Produktionssystemen und der Warteschlangentheorie [nach NYH05 S.418 ff.] ist die Methode des Trichtermodells eine geeignete Vorgehensweise. Auf ihr basierend wurde die Kennlinientheorie abgeleitet [nach WIE92 S.3 f.]. Das Trichtermodell ist in Abbildung 2.3 dargestellt.

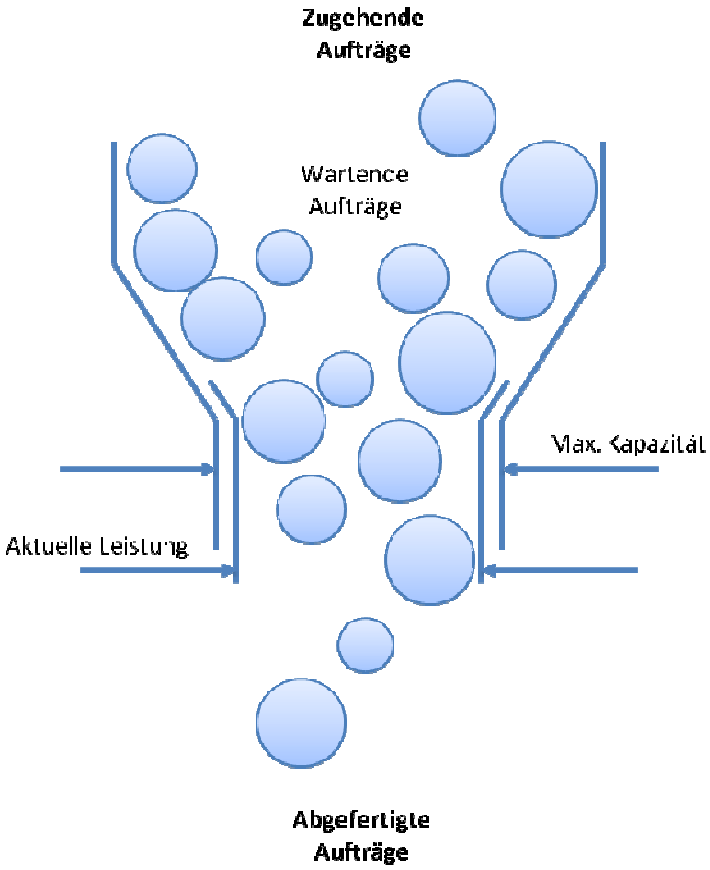

Abbildung 2.3: Darstellung des Trichtermodells [in Anlehnung an WIE92 S.3]

Das Trichtermodell veranschaulicht sehr schön, wie zugehende Aufträge von oben bildlich in den Trichter gefüllt werden und nach unten als abgefertigte Aufträge aus dem Trichter abfließen [vgl. WIE92 S.3 f.]. Die Maßeinheit für einen Auftrag ist dabei der Arbeitsinhalt in Stunden. Aufgrund der Tatsache, dass Aufträge manchmal nicht in dem gleichen Ausmaß abgearbeitet werden können, bildet sich Bestand vor den jeweiligen Arbeitsstationen. Dies

[1] Simulation ist ein Verfahren zur Nachbildung eines Systems mit seinen dynamischen Prozessen in einem experimentellen Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind. [VDI Richtlinie 3633]

entspricht der Trichterfüllhöhe. Der Durchmesser der unteren Trichteröffnung wird als maximale Kapazität angesehen. Da diese Kapazität in gewissen Grenzen variiert, kann die Trichteröffnung verkleinert oder vergrößert werden.

Ausgehend vom Trichtermodell (siehe Abbildung 2.3) werden Wechselwirkungen zwischen Bezugsgrößen im Zielsystem der Produktionslogistik abgeleitet. Der Bestand ist dabei eine unabhängige Größe wogegen die Größen Leistung, Durchlaufzeit und Termintreue resultierende Größen sind. Diese drei Kennlinien sind in Abbildung 2.4 dargestellt.

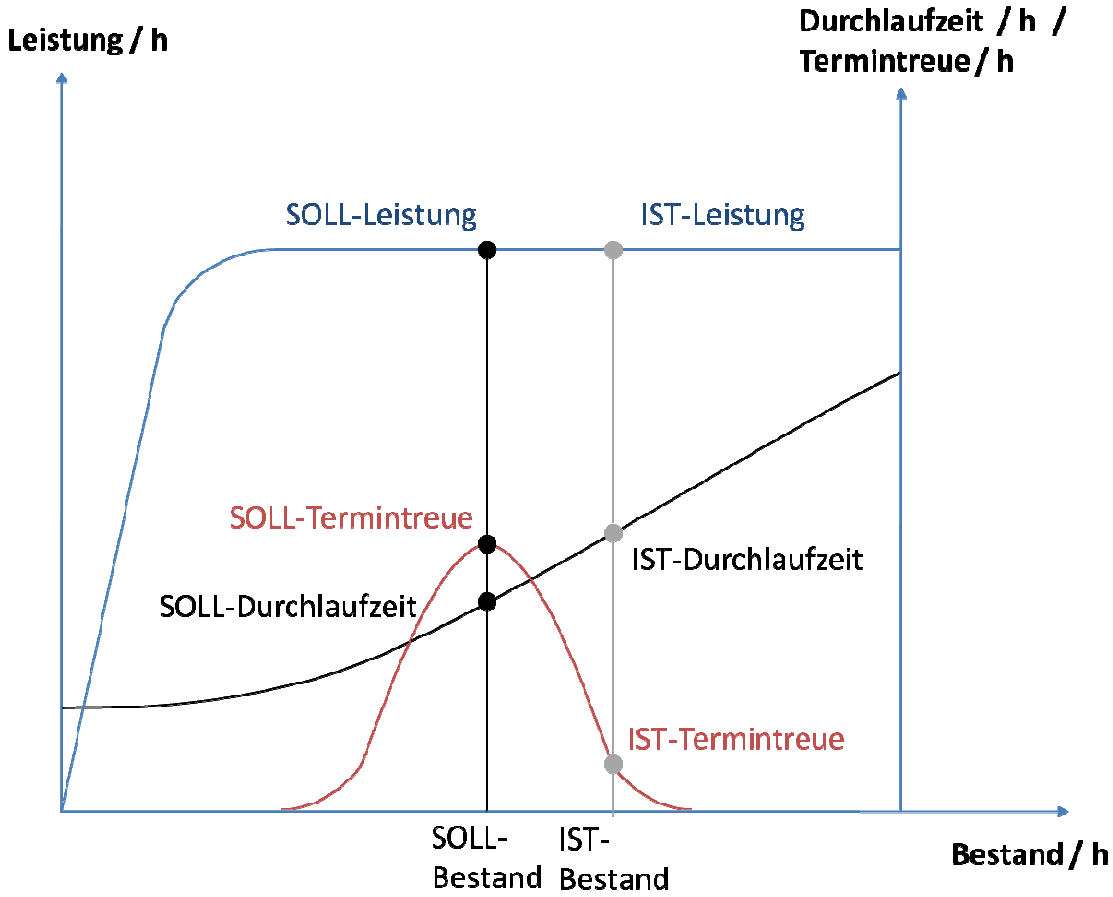

Abbildung 2.4: eines Produktionssystems [in Anlehnung an NYH04 S.39]

Es ist zu erkennen, dass die Leistungskennlinie, hier blau dargestellt, zunächst stark ansteigt, anschließend abflacht und asymptotisch gegen einen Grenzwert läuft [vgl. NYH03 S.61 ff.]. Folglich muss ein Mindestbestand im Produktionssystem vorliegen, damit die maximale Leistung erbracht werden kann. Bei Unterschreitung dieses Mindestbestandes kommt es zu Leerzeiten auf den Arbeitsstationen und somit zu einer ineffizienten Produktion. In diesem Anfangsbereich ist die Leistung proportional zum Systembestand. Im Bereich des

Mindestbestandes entstehen keine Warte- und Leerzeiten auf den Arbeitsstationen. Das System wird mit der bestmöglichen Leistung betrieben. Wird der Mindestbestand weiter überschritten, so ist kein Leistungszuwachs mehr möglich, da lediglich Warteschlangen vor den Arbeitsstationen entstehen. Diese arbeiten jedoch bereits mit voller Kapazität und somit kann kein Leistungszuwachs erreicht werden.

Die schwarze Linie kennzeichnet die Durchlaufzeit[2]. Unterhalb des Mindestbestandes liegt die Durchlaufzeit auf einem Niveau, welches nicht zu unterschreiten ist. Die Mindest-durchlaufzeit setzt sich aus der notwendigen Bearbeitungszeit[3] und der Mindest-übergangszeit zusammen [vgl. NYH03 S.61 ff.]. Bei Überschreitung des Mindestbestandes bilden sich erste Warteschlangen im System, damit steigen auch die Übergangszeit und die Durchlaufzeit an. Je größer der Bestand im Produktionssystem, desto länger wird auch die durchschnittliche Durchlaufzeit.

Die Termintreue ist als rote Linie dargestellt. Die Termintreue wird hier als minimale Abweichung vom ursprünglichen Termin betrachtet [vgl. NYH04 S.3 ff.]. Dieses bedeutet, dass sich sowohl verfrühte als auch verspätete Aufträge negativ auf die Termintreue auswirken. Der Endtermin eines Auftrages hängt direkt mit der Durchlaufzeit zusammen. Da die Durchlaufzeit jedoch im normalen Betrieb nicht fix ist, sondern um einen Mittelwert streut, kommt es zu dynamischen Engpässen.

Die Betriebskennlinien Darstellung von Nyhuis ist eine vereinfachte Darstellung der Leistung eines Systems. Nyhuis benutzt eine C-Norm-Funktion, um den Verlauf der Leistungs-kennlinie zu approximieren, stellt aber keine kausalen Zusammenhänge dar. Ist ein System sehr stark überlastet, so kommt man schnell zu dem Urteil, dass sich auch die Leistung des Systems stark verschlechtert. Dies Bedeutet die Leistungskennlinie würde in der Realität abfallen. Die Auftragsbetrachtung erfolgt unter der Annahme, alles sei in einem Fluss. Diese Betrachtungsweise vernachlässigt allerdings den Sachbestand, dass es in der Regel nur diskrete Aufträge in einem System gibt. Somit kann die Darstellung der Betriebskennlinie nur eine idealisierte sein.

[2] Als Durchlaufzeit wird die Zeit bezeichnet, die ein Auftrag von seinem Beginn bis zum Ende seiner vollständigen Abarbeitung benötigt

[3] Als Bearbeitungszeit wird die Summe aus Rüstzeit und Einzelbearbeitungszeit multipliziert mit der Losgröße berechnet.

Mit Hilfe der Modellierung der Wechselwirkungen von Bestand, Leistung, Durchlaufzeit und Termintreue kann ein Produktionssystem quantitativ beschrieben werden. Zur Erkennung und Bewertung von Engpässen wurde die sog. Durchsatzkennlinie entwickelt und 1995 zum ersten Mal von Kreutzfeldt beschrieben [vgl. KRE95 S.70 ff.]. Es gelten drei Grundannahmen:

1. Die Leistung und Kosten eines Systems werden durch seine Engpässe bestimmt. Die Engpassleistung wiederum kann durch den Durchsatz[4] an Aufträgen gemessen werden, den die betroffene Arbeitsstation erzeugt. Im Gegensatz zur Wertschöpfung werden nur fertiggestellte Aufträge betrachtet. Somit wird der Auftragsbestand vor dem Engpass nicht berücksichtigt.
2. Der Durchsatz eines jeden Auftrages wird durch eine Durchsatzschranke beschränkt [vgl. KRE95 S.51]. Eine Durchsatzschranke ist definiert als die Station mit dem größten Verhältnis von Belastung zu Kapazität, die den Durchsatz eines bestimmten Auftrages beschränkt.
3. Alle Aufträge, die durch die gleiche Durchsatzschranke laufen, werden zu einem kontinuierlichen Auftragsstrom zusammengefasst [vgl. KRE95 S.69]. Dabei wird der Auftragsstrom vereinfacht als frei skalierbar angesehen. Er enthält die durchschnittliche Einlastung[5] über der Arbeitsstation innerhalb einer Periode.

Damit wird ein Produktionssystem als ein Netzwerk aus Durchsatzschranken und Auftragsströmen modelliert [vgl. SCH09 S.10 f.]. Ein Algorithmus zur Bestimmung und zur Bewertung von Engpässen auf der Basis von Durchsatzkennlinien lässt sich mit einer Analogie zur Elektrotechnik entwickeln. In diesem Vergleich kann der Engpass annähernd als ein elektrischer Widerstand verstanden werden. Mit steigendem Widerstand wird der Stromdurchfluss, der hier dem Auftragsstrom entspricht, behindert. Ein elektrischer Widerstand würde nun bei zu hoher Belastung durchbrennen. Ein Engpass hingegen kann immer stärker belastet werden ohne sich dabei zu verändern. Die eingerasteten Aufträge üben, wie eine Potentialdifferenz, Spannung auf die einzelnen Aufträge aus. In Anlehnung an die Berechnung eines elektrischen Netzwerkes kann ein Algorithmus abgeleitet werden, der den Engpass anhand seiner Wirkung auf den Systemdurchsatz bestimmt. Die Analogie ist der Abbildung 2.5 zu entnehmen.

[4] Der Durchsatz ist ein Synonym für die Ausbringung der Produktionsleistung [vgl. DIN 8743]

[5] Als Einlastung wird der Arbeitsinhalt aller, innerhalb einer Periode, eingeplanten Aufträge auf einer Arbeitsstation oder in einem Auftragsstrom

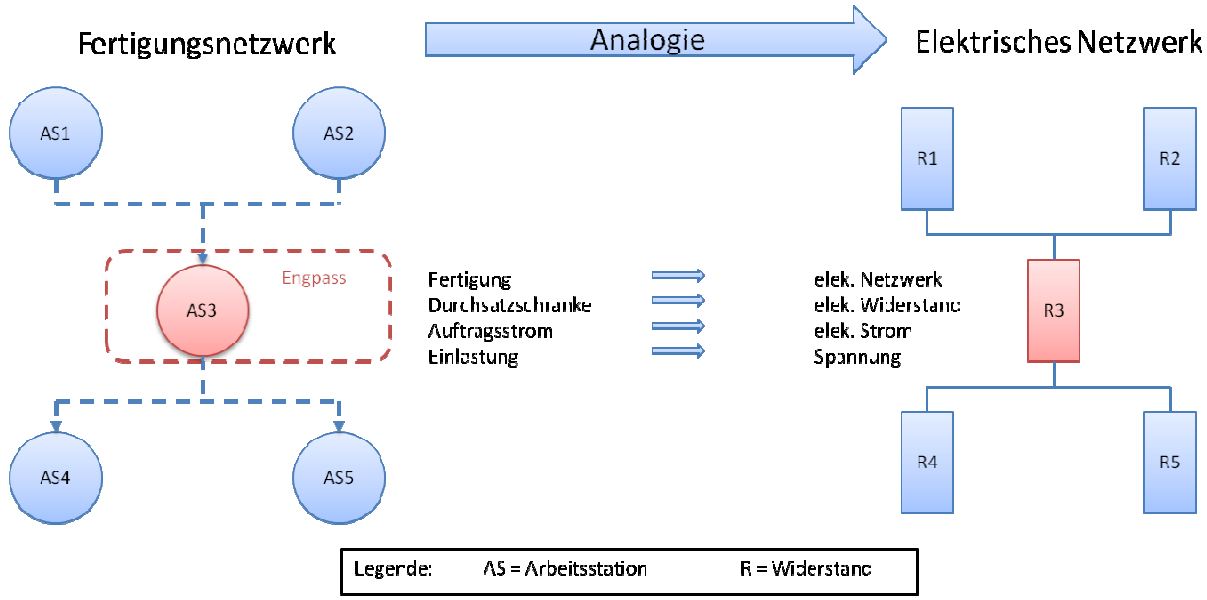

Abbildung 2.5: Analogie elektrisches Netzwerk zu einem Fertigungsnetzwerk [In Anlehnung an KRE07 S.12]

So kann für jede Durchsatzschranke eine Durchsatzkennlinie gezeichnet werden. Die Durchsatzkennlinie beschreibt als Funktion den Zusammenhang zwischen Durchsatz als abhängiger Größe und Einlastung als unabhängige Größe [vgl. KRE95 S.70 ff.]. In der folgenden Abbildung 2.6 ist eine Durchsatzkennlinie beispielhaft dargestellt.

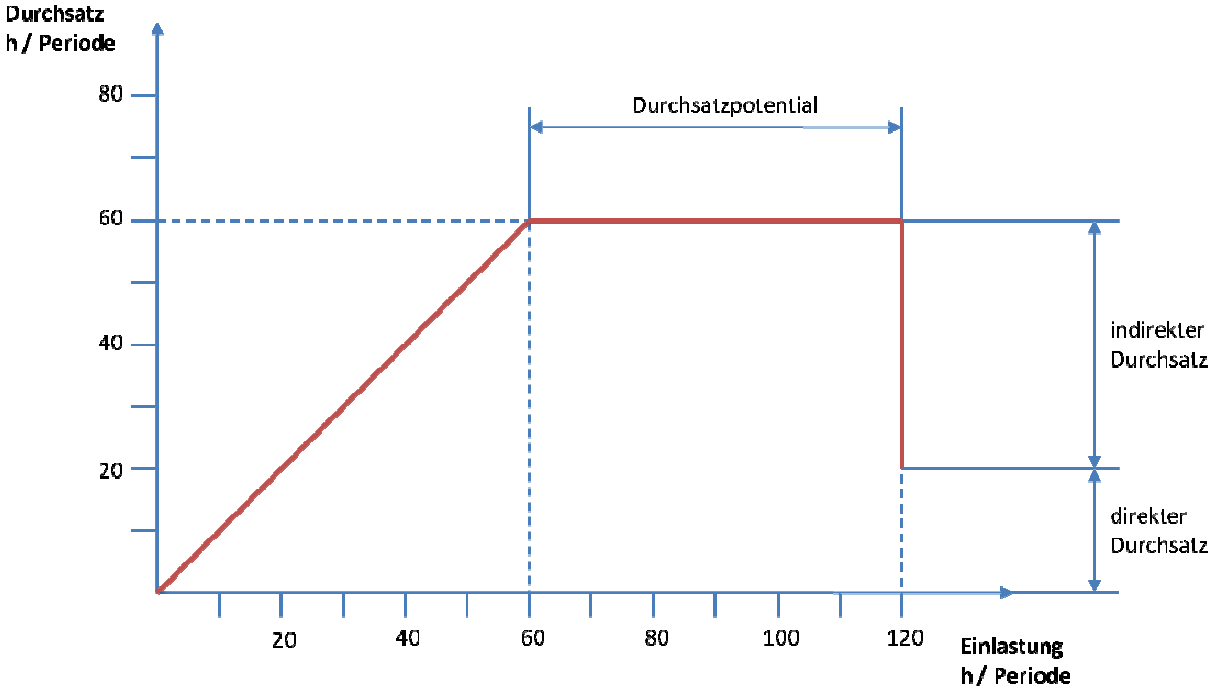

Abbildung 2.6: Darstellung einer Durchsatzkennlinie [vgl. SCH08 S.12]

Die eingelastete Arbeit ist hier auf der Abszisse und die durchgesetzte Arbeit auf der Ordinate einzutragen. Es wurden $120\ \frac{h}{Periode}$ eingelastet und $60\ \frac{h}{Periode}$ durchgesetzt. Es ist zu erkennen, dass bei einer Einlastung von $60\ \frac{h}{Periode}$ alle Aufträge durchgesetzt werden können. Dies wird durch die Winkelhalbierende am Anfang der Kennlinie dargestellt.

Werden Aufträge darüber hinaus eingeplant, so können diese die Durchsatzschranke nicht mehr passieren, d.h. sie werden nicht mehr abgearbeitet. Diesen Sachverhalt stellt die horizontale Linie dar. Sie wird als Durchsatzpotential bezeichnet. Der Begriff begründet sich dadurch, dass das Potential bei Erhöhung der Engpasskapazität maximal gehoben werden kann [vgl. SCH09 S.12]. Eine Steigerung der Kapazität über diesen Betrag hinaus ist nicht sinnvoll, da es zu weiteren Freikapazitäten und somit zu Ineffizienzen kommt. Wird also eine Optimierung des Produktionssystems im Sinne einer Bestandsreduzierung angestrebt, sollten zunächst die Engpassstationen mit dem größten Durchsatzpotential betrachtet werden.

Anhand der Durchsatzkennlinie lassen sich weiter die Auswirkungen des Engpasses auf vor- und nachgelagerte Prozessschritte ermitteln. Dazu wird die gesamte durchgesetzte Arbeit des Auftragsstromes untersucht und danach unterschieden, ob der Arbeitsanteil auf der Durchsatzschranke selbst oder auf einer der anderen Stationen im Verlauf des Auftragsstromes abgearbeitet wurde. Die durchgesetzte Arbeit an der Durchsatzschranke wird als direkter Durchsatz bezeichnet, während der Durchsatzbeitrag der anderen Stationen als indirekter Durchsatz benannt wird.

2.1.2 Beschreibung des DePlaVis-Algorithmus 2.0

Im Rahmen des DePlaVis-Projektes wurde ein neuer Algorithmus an den bestehenden Algorithmus von Kreutzfeldt angelehnt. Dieser neue Ansatz rechnet jedoch deterministisch und setzt lineare Zusammenhänge voraus. Das Vorgehen gliedert sich dabei wie folgt [vgl. ALT08]:

1. *Bestimmung des maximalen Verhältnis von Belastung zu Kapazität*
    1. Sollte eine der Arbeitsstationen eine Kapazität von null aufweisen, so wird diese betrachtet. Treten mehrere Arbeitsstationen mit einer Kapazität von null auf, so wird die Arbeitsstation selektiert, die über die größte Belastung verfügt.

2. Sollten zwei oder mehr Arbeitsstationen das gleiche Verhältnis von Belastung zu Kapazität aufweisen, so wird die Station gewählt, die die höchste Belastung aufweist.

2. Aufstellen des Optimierungsmodells für die ausgewählte Arbeitsstation

Zielfunktion

$$max \sum AF_i \qquad [2.3]$$

unter der Nebenbedingung

$$\sum a_j * AF_i \leq K_{AS_k} \qquad [2.4]$$

$$mit\ AF_i \in \{0,1\}$$

AF_i = Auftrag i

a_j = Bedarf j

K_{AS_k} = Kapazität der Arbeitsstation k

3. Nach Lösung des Optimierungsproblems kann der direkte und indirekte Durchsatz bestimmt werden. Anschließend wird über den Schlupf des gelösten Problems die freie Kapazität bestimmt.

4. Sollten noch nicht alle Aufträge berücksichtigt worden sein, dann wird wieder bei Schritt 1 begonnen.

In einem Beispiel soll die Vorgehensweise des DePlaVis-Algorithmus 2.0 einmal veranschaulicht werden. Das in Abbildung 2.7 dargestellte Fertigungsnetz gilt es zu betrachten.

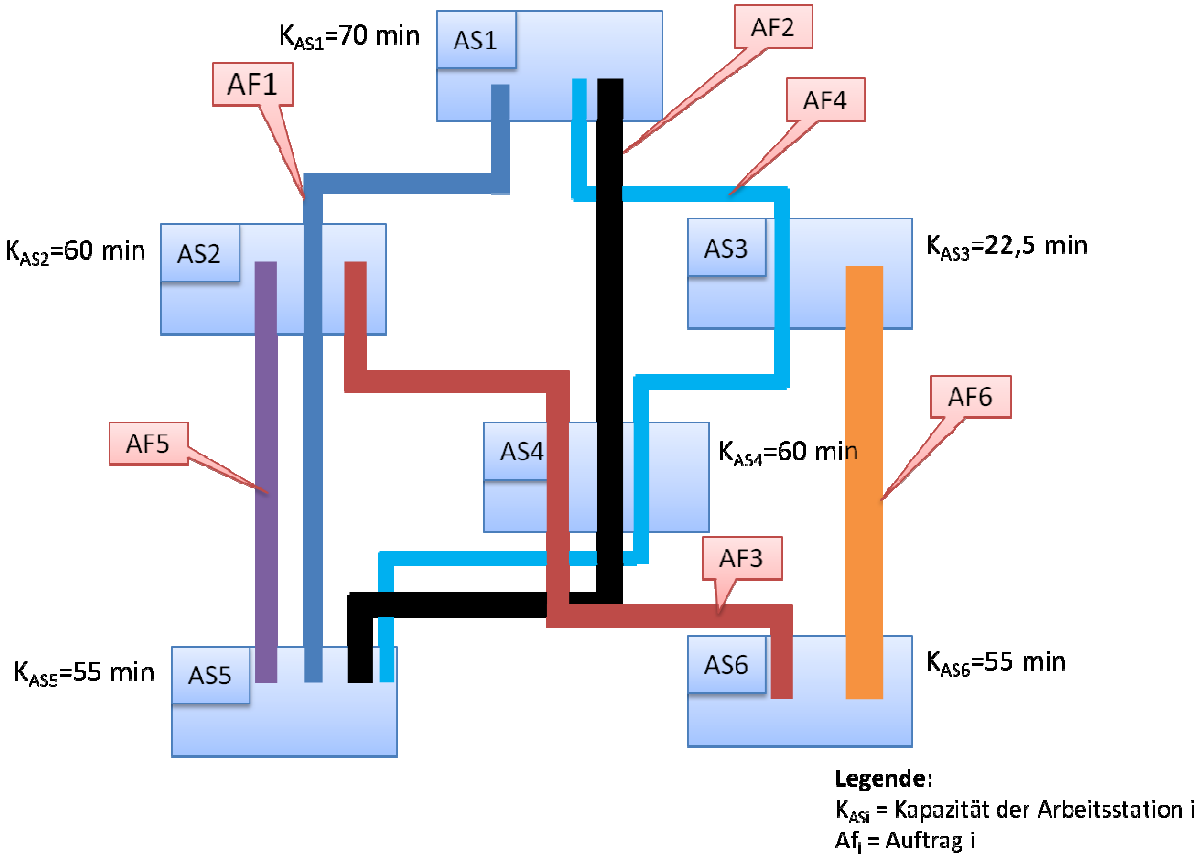

Abbildung 2.7: Übersicht des Fertigungsnetzwerks mit Arbeitsstationen inklusive der Kapazität und der Zuordnung der Aufträge [Eigene Darstellung]

Dabei sind denen als Rechtecke dargestellten Arbeitsstationen AS_i jeweils eine Kapazität K_{AS_i} zugeordnet. Die Aufträge AF_i sind als farbliche Verbindungen zwischen den jeweiligen Arbeitsstationen dargestellt. Es gilt nun zunächst das maximale Verhältnis von Belastung zu Kapazität zu bestimmen. Dabei wird der sogenannte Belastungsquotient eingeführt. Dieser errechnet sich aus:

$$BQ_i = \frac{AF_{ij}}{K_{AS_i}} \qquad [2.5]$$

Die Auftragsströme belasten die Arbeitsstationen dabei wie folgt:

- *AF 1*

  AS1 = 20 ZE

  AS2 = 20 ZE

  AS5 = 35 ZE

- *AF 2*

  AS1 = 25 ZE

  AS4 = 25 ZE

  AS5 = 25 ZE

- *AF 3*

  AS2 = 15 ZE

  AS4 = 15 ZE

  AS6 = 15 ZE

- *AF 4*

  AS1 = 15 ZE

  AS3 = 15 ZE

  AS4 = 15 ZE

  AS5 = 25 ZE

- AF 5

  AS2 = 25 ZE

  AS6 = 30

Somit ergeben sich für die Arbeitsstationen folgende Belastungsquotienten, die in Tabelle 2.1 aufgeführt sind. Es ist zu erkennen, dass die Arbeitsstation 5 den höchsten Belastungsquotienten mit 2,00 aufweist.

Tabelle 2.1: Belastungsquotienten der einzelnen Arbeitsstationen

Arbeitsstation	$\boldsymbol{BQ_i = \frac{AF_{ij}}{K_{AS_i}}}$
AS 1	0,85
AS 2	1,00
AS 3	2,00
AS 4	0,91
AS 5	2,00
AS 6	0,82

Nach Schritt drei der Vorgehensweise des DePlaVis-Algorithmus 2.0 gilt es nun das Optimierungsproblem aufzustellen und zu lösen.

Zielfunktion

$Maximiere\ (AF_1 + AF_2 + AF_4 + AF_5)$ [2.6]

Unter der Nebenbedingung dass

AS_1: $20 * AF_1 + 25 * AF_2 + 15 * AF_4 \leq 70$

AS_2: $20 * AF_1 + 25 * AF_5 \leq 60$

AS_3: $15 * AF_4 \leq 22{,}5$

AS_4: $25 * AF_2 + 15 * AF_4 \leq 60$

AS_5: $35 * AF_1 + 25 * AF_2 + 25 * AF_4 + 25 * AF_5 \leq 55$

AF_i $[0,1]$

Nach lösen des Optimierungsproblems sollen die Aufträge AF_2 und AF_4 freigegeben werden. Daraus resultiert ein Gesamtdurchsatz von 145 ZE bei einer Einlastung von 270 ZE. Davon sind 50 ZE direkter Durchsatz und 95 ZE indirekter Durchsatz. Der Rest Bestand beträgt 125 ZE.

Da jedoch noch nicht alle Aufträge berücksichtigt wurden, gilt nach Schritt vier des DePlaVis-Algorithmus 2.0 nun erneut bei Schritt eins zu beginnen. Dabei ist zu berücksichtigen, dass die Kapazitäten nun nicht mehr voll zur Verfügung stehen, sondern angepasst werden müssen. In Abbildung 2.8 sind die restlichen Aufträge, ihre Materialflüsse und die angepassten Kapazitäten dargestellt. An Arbeitsstation drei ergibt sich der neue Systemengpass mit einem Belastungsquotienten von 4,00. Da bei allen Arbeitsstationen die Kapazitäten ausreichend sind, kann der Auftrag fünf komplett durchgesetzt werden. Der gesamte Durchsatz beträgt somit 45 ZE, der direkte Durchsatz 15 ZE und der indirekte Durchsatz 30 ZE bei einer Einlastung von 45 ZE und einem Bestand von 45 ZE. Aufgrund der Kapazität von Arbeitsstation drei kann der Auftrag sechs nicht mehr durchgesetzt werden und es ergibt sich ein Bestand von 60 ZE.

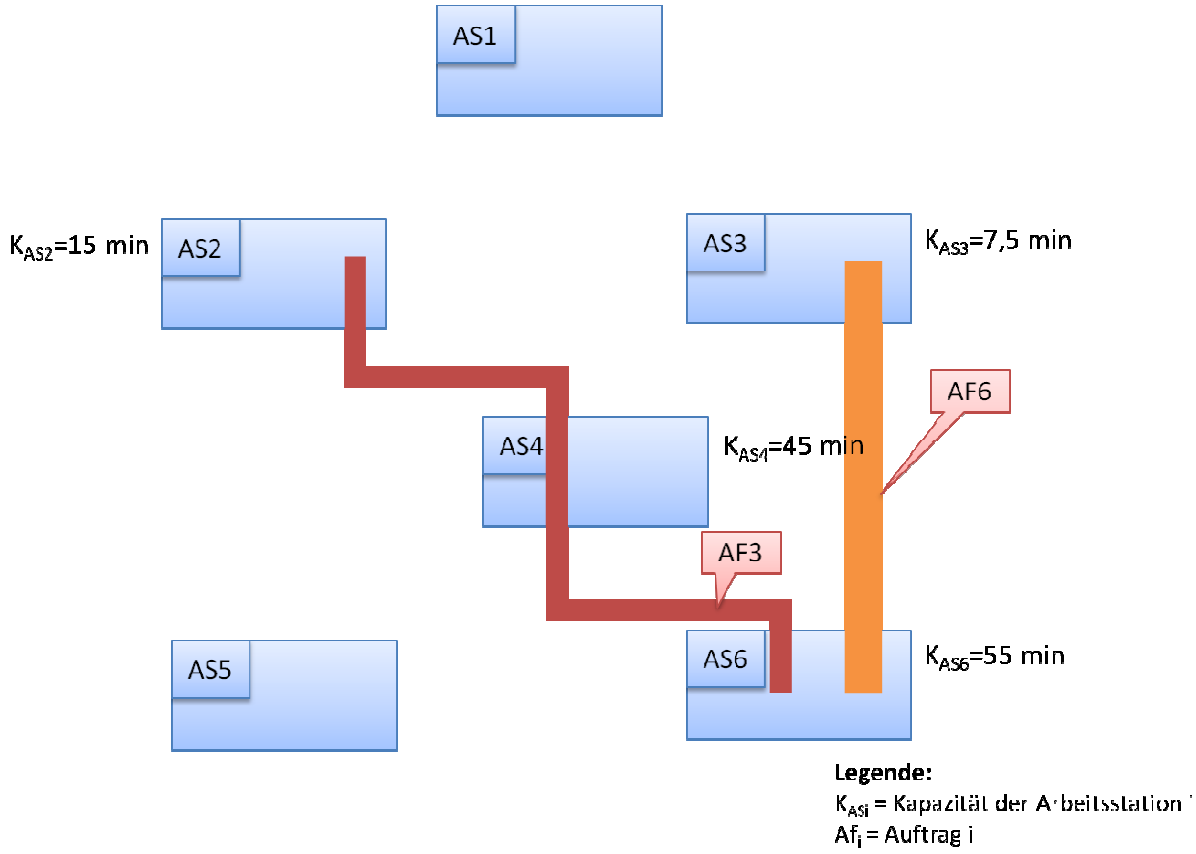

Abbildung 2.8: Fertigungsnetzwerk nach dem Eliminieren der berücksichtigten Aufträge [Eigene Darstellung]

Aufgrund der Annahme von linearen Zusammenhängen und nur einem Optimierungskriterium ist der DePlaVis-Algorithmus 2.0 stark eingeschränkt. Der DePlaVis-Algorithmus 2.0 ermittelt daher nur die Aufträge, die im Bezug auf den maximalen Durchsatz optimal sind. Es ist nicht möglich nach mehreren Kriterien zu optimieren, beispielsweise nach dem höchsten Durchsatz und der gleichzeitig kleinsten Durchlaufzeiten. Auch die in der Realität nicht linearen Zusammenhänge können hier nicht abgebildet werden. Das Ergebnis ist also stark idealisiert und im Hinblick auf die freizugebenen Aufträge in einer bestimmten Reihenfolge nicht optimal.

2.2 Probleme durch nicht Berücksichtigung der Arbeitsgangreihenfolge

Ein grundsätzliches Problem bei der Engpassermittlung ist die Nichtberücksichtigung der Arbeitsgangreihenfolge. Die Aussagen über Belastung und Kapazität werden lediglich durch die Addition von Bearbeitungs- und Übergangszeiten getroffen. Es ist dabei egal, ob eine Vorwärts- oder Rückwärtsterminierung durchgeführt wurde, da in beiden Fällen falsche Werte ermittelt werden. Bei der Addition der Bearbeitungszeiten, die innerhalb einer Periode

auf einer Bearbeitungsstation geplant sind, wird keine Aussage darüber getroffen, ob die eingelasteten Aufträge die jeweilige Arbeitsstation überhaupt erreichen.

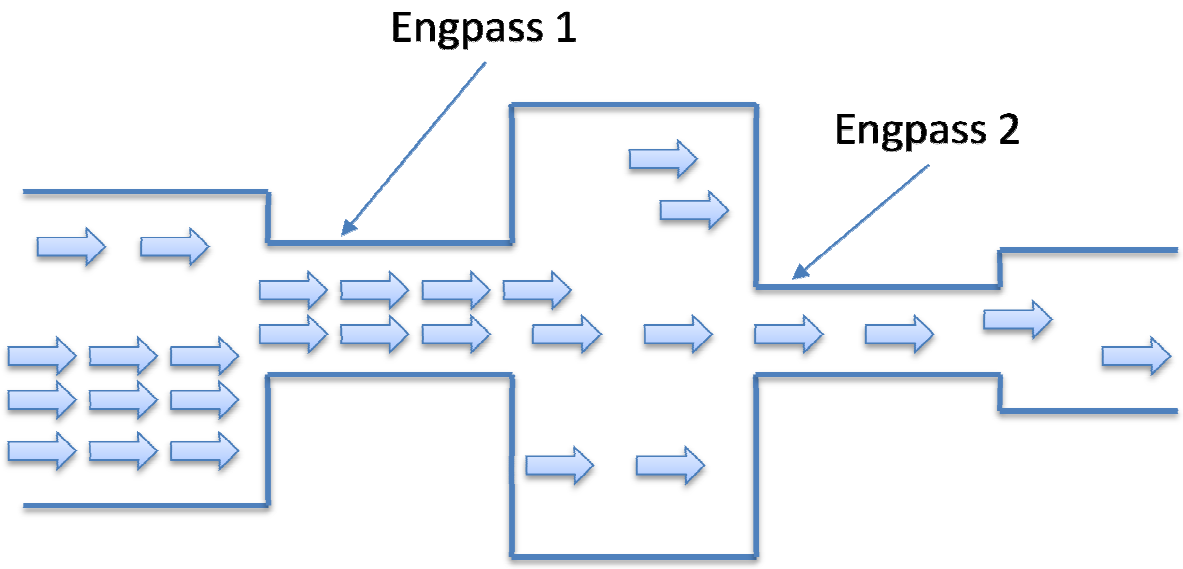

Abbildung 2.9: Engpässe aufgrund disproportionaler Teilkapazitäten [in Anlehnung an HAB99 S.18]

In Abbildung 2.9 ist ein Produktionssystem als Rohrleitung dargestellt, die von Aufträgen durchflossen wird. Die Aufträge sind als Pfeile abgebildet. Der Durchmesser des Rohres kann als Maß für die Teilkapazität angenommen werden. Je größer der Rohrdurchmesser, desto größer ist also die Kapazität der Bearbeitungsstation. Im vorliegenden Beispiel werden die ersten Aufträge bereits durch den Engpass eins aufgehalten und können somit nicht mehr innerhalb der Periode abgearbeitet werden. Sie haben also gar nicht die Möglichkeit den Engpass zwei noch zu erreichen. Daher ist die angenommene Belastung der nachfolgenden Arbeitsstationen zu hoch eingeschätzt worden und es kommt zu sogenannten Pseudoengpässen. Der DePlaVis-Algorithmus bildet nun eine Reihenfolge nach dem größten Engpass [vgl. Kapitel 2.1]. Die Einlastung an der Durchsatzschranke mit der höchsten Belastung wird durch den Abfertigungsfaktor reduziert und die Kapazität auf den vor- und nachgelagerten Stationen korrigiert. Da der Auftragsstrom als kontinuierlich angesehen werden kann, ist dieses Vorgehen zulässig [vgl. KRE95 S.68 f.]. Ein solcher Strom, vergleichbar mit dem von Flüssigkeiten oder Gasen, lässt sich beliebig oft in infinitesimal kleine Stücke zerlegen und es entstehen keine Wartezeiten bei Abarbeitung einzelner Bestandteile. Der Durchsatz eines Auftragsstromes hängt also nur von der Abarbeitungswahrscheinlichkeit an der Durchsatzschranke ab. In der Praxis hingegen handelt es sich für gewöhnlich um eine endliche Anzahl an diskreten Aufträgen, die auch über einen genau definierten und diskreten Arbeitsinhalt verfügen. Weiterhin unterliegen die

Aufträge einer gewissen Verteilung in den Dimensionen Arbeitsinhalt und Termin, welche u.a. von Produktionsprozess, -planung und Losgröße abhängig sind. Daher lässt sich ein Auftragsstrom in der Praxis nicht in beliebig viele und kleine Bestandteile aufteilen.

Ein weiteres Problem ist das dynamische Engpassverhalten. Der bisherige Algorithmus nutzt nur Durchschnittswerte. Somit sind diese Gegebenheiten nicht berücksichtigt. Die Einlastung wird also nicht zeitdifferenziert betrachtet, daher ist keine Information vorhanden, wann der Auftrag eine Arbeitsstation erreicht. Dieses kann zu temporären Engpässen führen und somit zu Warteschlangen. Diese verursachen wiederrum Verzögerungen im Ablauf, die später nicht mehr aufgeholt werden können und demzufolge zu Durchsatzverlusten führen.

Somit lassen sich drei Gründe zusammenfassen, die bei nicht Berücksichtigung der Arbeitsreihenfolge auftreten:

1. Durch die diskreten Arbeitsinhalte der einzelnen Aufträge entstehen Wartezeiten für nachfolgende Aufträge. Wird ein Auftrag auf einer Bearbeitungsstation abgefertigt, so ist diese blockiert für andere nachfolgende Aufträge. Kommen weiterhin Aufträge an der Arbeitsstation an, so bilden sich Warteschlangen, welche den Materialfluss hindern und somit den Durchsatz reduzieren. Dieses Verhalten tritt genau dann auf, wenn zwei Materialströme auf einer Arbeitsstation durchgesetzt werden müssen oder die Arbeitsinhalte der Aufträge verschieden groß sind. In diesem Fall spielt die Kenntnis, ob ein Auftrag auf der vorherigen Station abgearbeitet wurde eine große Rolle, da nur so eine Aussage darüber getroffen werden kann, ob sich eine Warteschlange, die den Durchsatz behindert, bildet.
2. Ein Auftrag gilt erst als durchgesetzt, wenn er komplett abgeschlossen ist. Ist ein Aufrag nicht am Ende einer Periode komplett durchgesetzt, so gilt dieser als nicht durchgesetzt. Daher muss der Durchsatz des Aufragstromes am Ende immer auf volle Aufträge abgerundet werden.
3. Die Länge der Bearbeitungsreihenfolge beeinflusst die Abarbeitung eines Auftrages. Je später ein Auftrag gestartet wird, desto geringer ist die Wahrscheinlichkeit, dass er noch in der laufenden Periode durchgesetzt werden kann. Dieses geht einher mit der Länge der Bearbeitungsfolge. Je länger also die Reihenfolge, desto unwahrscheinlicher ist es, dass ein Auftrag alle Stationen passieren und komplett abgearbeitet werden kann.

3 Einführung in das Operations Research

Im nachfolgenden Kapitel wird eine Einführung in die Thematik des Operations Research gegeben. Zu Beginn werden Grundlagen und Begrifflichkeiten kurz erklärt, um so einen Einstieg in die Thematik zu gewährleisten. Es sollen verschiedene Optimierungsverfahren vorgestellt werden, da auf dieser Grundlage das spätere Optimierungsproblem zu lösen ist. Im Nachfolgenden gilt es einen Einblick in die Maschinenbelegungsplanung zu geben.

3.1 Grundlagen des Operations Research

Unter dem Begriff des Operations Research wird im Allgemeinen die Entwicklung und der Einsatz quantitativer Modelle und Methoden zur Entscheidungsunterstützung in Unternehmen und Organisationen verstanden [vgl. SUM09 S.5 f.]. Dieser Begriff, wie auch Operational Research oder in Deutschland Unternehmensforschung, wurde Mitte des vorigen Jahrhunderts von der englischen Armee geprägt [vgl. ZIM08 S.6 f.]. Angewandte Vorgehensweisen und Methoden sind teilweise jedoch deutlich älter [vgl. WER08 S.1 ff.]. Der Durchbruch gelang im zweiten Weltkrieg, indem das Operations Research einige Erfolge auf militärischer Seite feiern konnte [vgl. HOS01 S.42]. In den 1960er Jahren wurden zunehmend Organisationen und Unternehmen auf diese Methodik aufmerksam, da die Erfolge des Militärs gegen Ende des zweiten Weltkrieges hauptsächlich dem Operations Research zu geschrieben wurden [vgl. ZIM08 S.8 f].

Typische Ansätze des Operations Research sind Optimierung[6] und Simulation. Dabei gilt es ein abstraktes Modell für einen Ausschnitt der Realität abzubilden und mit dessen Hilfe Analysen durchzuführen, um somit eine gute Basis für Entscheidungen abzuleiten [vgl. SUM09 S. 5 f.]. Ein Model ist dabei ein zweckorientiertes, ggf. vereinfachtes Abbild eines Ausschnittes der Realität, welches hinsichtlich der interessierenden Zusammenhänge strukturähnlich oder strukturgleich ist [vgl. WER08 S.3].

Charakteristisch für die Vorgehensweise des Operations Research ist das Bestreben für komplexe Situationen optimale Handlungsvorschläge zu ermitteln. Hierbei wird Optimalität

[6] Optimierung ist in diesem Zusammenhang ein Teilgebiet der numerischen Mathematik, das sich mit der optimalen Festlegung von Größen, Eigenschaften, zeitlichen Abläufen u.a. eines Systems unter gleichzeitiger Berücksichtigung von Nebenbedingungen befasst.

entscheidungstheoretisch fundiert, d.h. unter den zu berücksichtigen Nebenbedingungen gilt es, die beste Alternative, gemessen an der Zielerreichung, auszuwählen [vgl. WER08 S.1 f.]. Der Schwerpunkt zur Lösung solcher Fragestellungen liegt auf der Entwicklung von formal-mathematischen Methoden. Diese werden mit Hilfe der Informations- und Kommunikationstechnologie in sogenannte Entscheidungsunterstützungssysteme eingebettet [vgl. SUM09 S. 6 ff.].

Entscheidungen sollten durch Planung vorbereitet werden. Unter Planung wird ein vom Planungsträger durchgeführter, systematischer und rationaler Prozess zur Ermittlung von Maßnahmen zur zukünftigen Zielerreichung bezeichnet [vgl. DOD02 S.1 f.]. Neben der Ermittlung von Alternativen zur Zielerreichung kann auch das Aufzeigen von anzustrebenden Lösungen als Aufgabe der Planung betrachtet werden. Planung wird quantitativ genannt, wenn mathematische Modelle und Methoden genutzt werden, die zur Entscheidungsfindung dienen. Insbesondere bei der Lösung komplexer Fragestellungen sind quantitative Methoden von großer Bedeutung. Durch die damit verbundene strukturierte Herangehensweise, die die Planung voraussetzt, an ein Problem und die Entwicklung eines geeigneten Modells ergeben sich wertvolle Einsichten für den Entscheidungsträger. Dieser kann also auf der Grundlage, der sich durch die Planung ergebenden Möglichkeiten, gut Ergebnisse ableiten. Diese werden in der Regel nach Lösung des Modells noch verbessert, indem entweder bereits ein optimaler Vorschlag festgestellt wird oder Hinweise auf eine Ergebnisverbesserung abgeleitet werden können [vgl. WER08 S. 2 ff.].

Ein wesentliches Merkmal des Operations Research besteht darin, einen relevanten Ausschnitt der Realität abstrahiert in einem quantitativen Modell zu entwickeln, d.h. die Realität adäquat abzubilden. Aus diesem Modell werden unter Einsatz eines speziell auf die Modellstruktur angestimmten Algorithmus[7] und geeigneter Informationsverarbeitungstechnologien Lösungen ermittelt [vgl. DOD02 S.3 ff.]. Bei einem Operations Research Problem muss zunächst geklärt werden, was der Entscheidungsträger wirklich erreichen will. Das Ziel des Entscheidungsträgers wird in der Zielfunktion, die entweder maximiert oder minimiert werden kann, formuliert. Zu klären ist jedoch zusätzlich, welche Entscheidungsvariablen (Freiheitsgrade) und welche Nebenbedingungen (Restriktionen) zu berücksichtigen sind. Ist ein Modell aufgestellt und die Daten sind korrekt erfasst, bereinigt sowie in

[7] Unter einem Algorithmus kann ganz allgemein eine Verarbeitungsvorschrift zur Lösung eines Problems verstanden werden. Eine detailliertere Darstellung der Verarbeitungsvorschrift kann durch ein ablauffähiges Programm in einer Programmiersprache geschehen und mit einem Flussdiagramm oder Ablaufdiagramm, dargestellt werden. [DIN 6601]

ausreichendem Maße vorhanden, kann man durch Analyse des Modells Lösungen für optimale Entscheidungen generieren. Aufgrund der Isomorphie[8] des Modells mit der Realität sind die Lösungen in echte Entscheidungssituationen der Realität übertragbar. Strukturgleichheit liegt vor, wenn sowohl die beiden Mengen als auch die beiden Strukturen bis auf ihre Beziehungen übereinstimmen. In Abbildung 3.1 sind drei Modelle nach ihrem Zweck strukturiert.

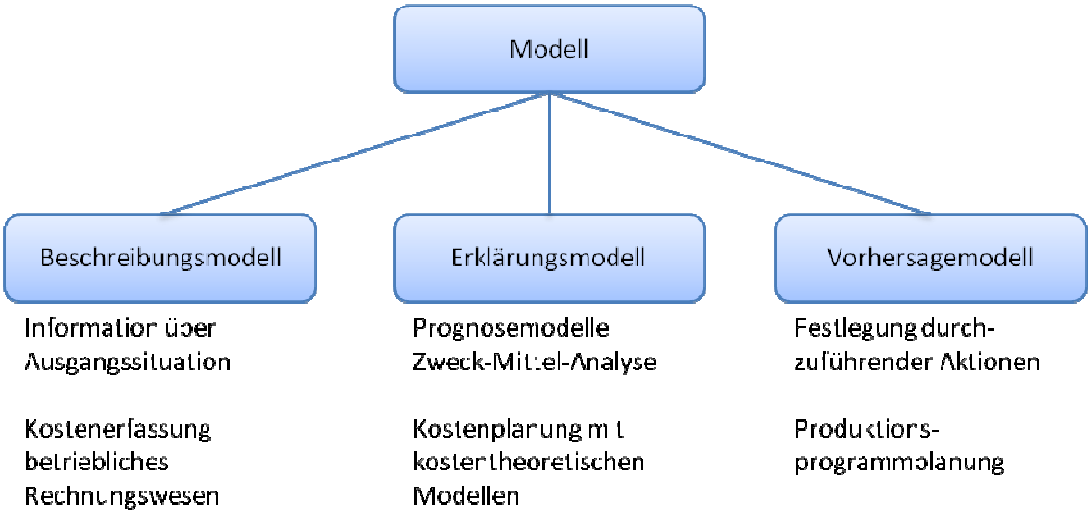

Abbildung 3.1: Modelle strukturiert nach ihrem Zweck [in Anlehnung an WER08 S.4]

Sind die oben genannten Komponenten Zielfunktion, Entscheidungsvariablen und Restriktionen formal explizit ausgedrückt, kommt ein Optimierungsmodell, auch Vorhersagemodell genannt, in Frage [vgl. SUM09 S. 8 f.]. Eine Optimierungsmethode kann dann eine optimale Lösung unter Berücksichtigung der Zielfunktion und aller Nebenbedingungen generieren. Für den Fall, dass es keine klare Zielsetzung gibt, werden mit Hilfe eines Simulationsmodells verschiedene Entscheidungsvarianten durchgespielt.

3.2 Optimierungsmethoden des Operations Research

Operations Research befasst sich insbesondere mit Entscheidungsmodellen und der Ermittlung einer optimalen Lösung. Es werden quantitative Erklärungsmodelle zur Entscheidungsunterstützung eingesetzt. Diese führen meist zu einer Ergebnisermittlung und Bewertung jeweils einzelner Alternativen. Eine Methode wird in diesem Zusammenhang als das Vorgehen einschließlich Modellierung und Problemlösung unter Einsatz von Algorithmen

[8] Isomorph bedeutet von gleicher Gestalt und Struktur

verstanden. Gelegentlich werden Methoden und Algorithmen auch synonym verwendet [vgl. WER08 S.8 ff.].

Modelle, die im Rahmen realer Problemstellungen zu lösen sind, gilt es in der Regel zu optimieren. Unter Optimierung versteht man die Ermittlung derjenigen zulässigen Lösung, die am besten von allen gefundenen Alternativen geeignet ist. Damit wird vorausgesetzt, dass alle zulässigen Alternativen bekannt sind und die Zielvorstellung klar definiert ist. Es gilt nun die Alternativen mit der Zielvorstellung zu bewerten und deren Ergebnisse zu vergleichen. Danach kann eine optimale Lösung gewählt werden. Bei einer Vielzahl von Alternativen wird angestrebt, alle Handlungsalternativen mittels eines mathematischen Optimierungsmodells implizit zu erfassen und unter Einsatz eines Optimierungsalgorithmus die beste Alternative zu berechnen [vgl. WER08 S.8 f].

In der Literatur sind zur Lösung von Optimierungsproblemen mehrere Verfahren bekannt, die unter anderem bei Zimmermann (2008), Neumann (2002) oder Pinedo (2009) detailliert dargestellt sind. In den nachfolgenden Kapiteln werden jedoch nur die lineare und ganzzahlig-kombinatorische Optimierungsmethoden vorgestellt, da diese die Grundlagen für das in Kapitel vier zu erarbeitende Lösungskonzept darstellen sollen.

3.2.1 Lineare Optimierung

Sind eine Zielfunktion und alle zugehörigen Restriktionen eines Optimierungsmodells Linearkombination der Entscheidungsvariablen, können Modellierungs- und Lösungstechnologien der linearen Optimierung (Linear Programming, LP) eingesetzt werden. Dabei wird, wie schon erwähnt, eine gegebene Zielfunktion minimiert oder maximiert, unter der Berücksichtigung von linearen Restriktionen, die sowohl Gleichungen als auch Ungleichungen sein können [vgl. SUM09 S. 8f.]. Ein lineares Optimierungsproblem zeichnet sich also dadurch aus, dass die Zielfunktion eine lineare Funktion der Entscheidungsvariablen ist und dass die Nebenbedingungen in Form linearer Gleichungen und Ungleichungen für die Entscheidungsvariablen gegeben sind [vgl. NEM02 S.36 f.].

Bei einer Vielzahl von wirtschaftlichen und technischen Fragestellungen geht man von linearen Zusammenhängen und Zielkriterien aus. Ein Grund dafür ist die Einfachheit und Verständlichkeit der gegebenen Modelle. Es stehen leistungsfähige Verfahren, die auf modernen Rechnern Aufgaben mit Tausenden von Variablen und Nebenbedingungen mit

vertretbarem Aufwand lösen können, zur Verfügung. Aus diesem Grund werden auch „Nichtlinearitäten“ linearisiert, d.h. durch lineare Approximation[9] ersetzt [vgl. NEM02 S.35]. Gilt es ein in der Realität auftretendes Problem zu lösen, so sind folgende in Abbildung 3.2 aufgeführte Schritte der Reihe nach durchzuführen.

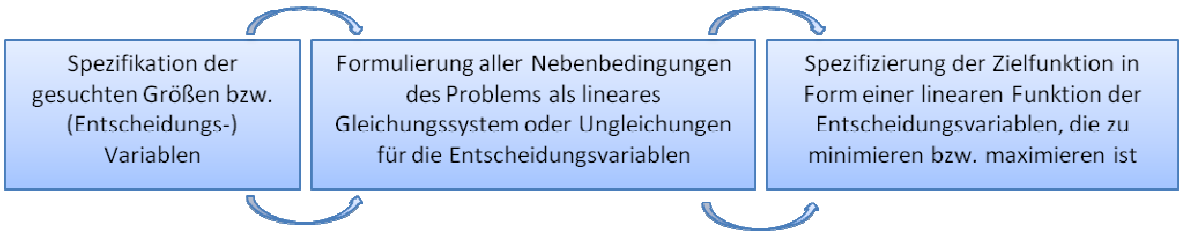

Abbildung 3.2: Aufstellen eines linearen Optimierungsproblems [Eigene Darstellung]

Ein Optimierungsproblem der linearen Optimierung besteht somit aus den folgenden Komponenten [vgl. SUM09 S.34 ff.]:

- *Entscheidungsvariablen, die kontinuierlich Werte zwischen gegebenen Schranken annehmen können,*
- *einer zu maximierenden oder zu minimierenden Zielfunktion und*
- *linearen Restriktionen, die vom Typ größer-, kleiner-gleich oder gleich sind.*

Entscheidungsvariablen

Die Entscheidungsvariablen entsprechen dabei dem Lösungsraum, sprich Entscheidungsfreiraum in der gegebenen Entscheidungssituation. Die Entscheidungsvariablen können beispielsweise Produktionsmengen einzelner Produkte oder Varianten, Mengen an Zutaten in Mischungen oder Flächen in der landwirtschaftlichen Produktion darstellen. Eine Entscheidung ist so zu treffen, dass sie im Sinne der gegebenen Zielfunktion optimal ist [vgl. SUM09 S.34]. Im Allgemeinen werden die Variablen dabei wie folgt bezeichnet:

$$x_j, so\ dass\ l_j \leq x_j \leq u_j\ für\ alle\ j \in 1, \dots, n \qquad [3.1]$$

[9] Approximation bedeutet angenäherte Bestimmung oder Darstellung einer unbekannten Größe, Funktion oder Zielpunkt

Das heißt, es gibt n Variablen, die jeweils eine (reelle) Untergrenze l_j und Obergrenze u_j haben. Die Variablen können auch nach unten oder oben unbeschränkt sein. In dem Fall ist die Unter- bzw. Obergrenze gleich $-\infty$ oder $+\infty$ [vgl. SUM S.34].

Zielfunktion

Die optimale Entscheidung ist von der Zielfunktion abhängig, da diese das Optimierungsproblem formuliert. Bei der linearen Optimierung ist die Zielfunktion eine Linearkombination der Variablen und soll maximiert oder minimiert werden [vgl. DOD02 S.17 ff.]. Beispielhafte Zielfunktionen sind z.B. Ertrags-, Deckungsbeitrags- oder Gewinnmaximierung, Kostenminimierung oder der Maximierung des ROI[10]. In der Praxis gilt es oft mehrere Ziele zu berücksichtigen, aber die lineare Optimierung erlaubt nur eine Zielfunktion. Kommen jedoch mehrere gewünschte Funktionen vor, so kann eine höchste Priorität vergeben werden, nach der dann optimiert wird. Man spricht in diesem Fall von der multikriteriellen Optimierung [vgl. HOO04 S.592 ff.]. Meistens können jedoch nicht alle Aspekte in einem formalen Modell dargestellt werden. Die Zielfunktion wird allgemein wie folgt dargestellt [vgl. SUM S.34 f.]:

$$Maximiere\ bzw.minimiere \sum_{j=1}^{n} c_j\ x_j \qquad [3.2]$$

Restriktionen

Restriktionen können in der linearen Optimierung als Ungleichungen oder Gleichungen dargestellt werden, wobei die „linke Seite“ eine Linearkombination der Entscheidungsvariablen und die „rechte Seite“ eine reelle Konstante ist. Typische Restriktionen sind z.B. Kapazitätsgrenzen in der Produktion, Verfügbarkeit der Rohmaterialien und prognostizierte Absatzmengen. Weiterhin müssen oft logische oder physikalische Gegebenheiten als Restriktionen definiert werden, damit das Modell korrekt bleibt. Bereits das Setzen von Ober- und/ oder Untergrenzen für die einzelnen Variablen sind Restriktionen, weil diese den Lösungsraum einschränken.

[10] ROI bedeutet Return on Invest

Die Restriktionen können folgendermaßen dargestellt werden:

$$\sum_{j=1}^{n} a_{ij}\, x_{ij} \ \leq\geq\ b_i \ f\ddot{u}r\ alle\ i \in \{1, \dots, m\} \qquad [3.3]$$

Somit handelt es sich für jedes i entweder um eine kleiner- oder größer-gleich Restriktion.

Das Grundmodell der Linearen Optimierung lautet daher wie folgt [vgl. UNG10 S25 ff.]:

Zielfunktion

$$maximiere\ bzw. minimiere\ z = c^T x \qquad [3.4]$$

Unter den Nebenbedingungen, dass

$$Ax \leq b$$

$$x \geq 0$$

$$c, x \in \mathbb{R}^n, b \in \mathbb{R}^m, b \geq 0, A_{m,n}\ .$$

Man nennt $z = c^T x$ die Zielfunktion in transponierter[11] Matrixschreibweise, c den Vektor der Zielkoeffizienten, $A_{m,n}$ (eine Matrix mit m Zeilen und n Spalten) die Koeffizientenmatrix, b den Kapazitätsvektor oder „rechte Seite“. Die Beschränkungen $x \geq 0$ (d.h. $x \geq 0$, $i = 1, \dots, n$) werden als Nichtnegativitätsbedingungen bezeichnet [vgl. ZIM08 S.72 ff.].

Ein klassisches Problem aus der Produktionsplanung soll die Vorgehensweise der linearen Optimierung verdeutlichen. Eine Firma produziere aus m Rohstoffen (oder allgemein mit der Hilfe von m Produktionsfaktoren oder Ressourcen) $R_1, \dots, R_m$ die q Produkte $P_1, \dots, P_q$. Für die Erzeugung einer Mengeneinheit oder Gewichtseinheit von P_j seien a_{ij} Einheiten des Rohstoffs R_i nötig ($i = 1, \dots, m; j = 1, \dots, q$). a_{ij} wird auch Produktionskoeffizient (oder Faktor-Inanspruchnahme) des Produktionsfaktors R_i pro Mengeneinheit P_j genannt. Von der

[11] Transponiert bedeutet Spiegelung einer Matrix an der Hauptdiagonalen bzw. hier Zeilenvektor statt Spaltenvektor und wird mittels hochgestelltem T kenntlich gemacht. Ein Vektor ohne weitere Angabe ist stets als Spaltenvektor zu verstehen. c^T bezeichnet also einen Zeilenvektor [vgl. PAP01 Band2 S.6 ff.].

Ressource R_i seien nur b_i Einheiten verfügbar (Kapazitätsrestriktion). Weitere Variable sind wie folgt definiert:

$$\left.\begin{array}{r} p_j\ der\ Verkaufspreis \\ k_j\ die\ variablen\ Kosten \\ g_j := \ p_j - k_j\ der\ Deckungsbeitrag \end{array}\right\}\ pro\ Einheit\ von\ P_j$$

Die Fixkosten bei der Produktion der q Güter $P_1, \dots, P_q$ wird mit k_0 bezeichnet. Es gilt im Folgenden ein optimales Produktionsprogramm zu suchen, d.h. es muss ermittelt werden, wie viele Einheiten P_j zu produzieren sind, so dass der Gesamtgewinn möglichst groß wird. Dabei wird vorausgesetzt, dass die gesamt produzierte Menge abgesetzt werden kann.

Sei x_j die Anzahl der von P_j produzierten Mengeneinheiten. Der Gesamtgewinn beträgt dann $b = \sum_{j=1}^{q} g_j x_j - k_0$. Für die Produktion von x_j Mengeneinheiten von P_j sind $a_{ij}x_j$ Einheiten des Rohstoffes R_i erforderlich und für die Produktion aller q Produkte $P_1, \dots, P_q$ folglich $\sum_{j=1}^{q} a_{ij}x_j$ Einheiten von R_i. Da R_i nur b_i Einheiten zur Verfügung stehen, ergibt sich somit folgendes zu mit Hilfe der linearen Optimierung zu lösendes Optimierungsproblem.

$$Optimierungsproblem \begin{cases} Zielfunktion: & \max \ \sum_{j=1}^{q} g_j\, x_j - k_0 \\ Nebenbedingungen: & \sum_{j=1}^{q} a_{ij}\, x_j \le b_j \ (i = 1, \dots, m) \\ & x_j \ge 0 \ (j = 1, \dots, q)\ . \end{cases} \qquad [3.5]$$

3.2.1.1 Lösungsverfahren für lineare Optimierungsmodelle

Erst nachdem ein Optimierungsmodell korrekt formuliert wurde, kann eine optimale Lösung ermittelt werden, auf dessen Basis ein Ergebnis abgeleitet werden kann. In einem LP-Problem handelt es sich um ein System von Gleichungen und Ungleichungen, das mehrere gültige Lösungen besitzt. Eine zulässige Lösung dieses Systems ist eine Wertekombination der Entscheidungsvariablen, die alle Restriktionen erfüllen [vgl. SUM09 S.44 f.]. Das besondere an LP-Problemen ist, dass sie fast immer unendlich viele Lösungen besitzen. Es gilt nun nach einer, im Sinne der Zielfunktion, optimalen Lösung zu suchen. Hierfür sind

mehrere Vorgehensweisen aus der Literatur bekannt. Der Vollständigkeit halber sind hier gängige Möglichkeiten zur Lösung eines linearen Optimierungssystems aufgeführt:

- *Grafische Lösungsverfahren*

  Sehr kleine Probleme können grafisch in annehmbarer Zeit gelöst werden. Kleine Probleme sind Aufgaben mit wenigen Entscheidungsvariablen und wenigen Restriktionen sowie Aufgaben besonderer Struktur, für die spezielle Lösungsmethoden Anwendung finden. Darüber hinaus können zwei Klassen linearer Probleme gelöst werden. Eine Klasse sind diejenigen, die genau zwei Entscheidungsvariablen haben und die zweite Klasse sind diejenigen, die neben den Nichtnegativitätsanforderungen (-bedingungen) genau zwei Restriktionen haben [vgl. SUM09 S.35 ff.]. Dieses Verfahren wird in Kapitel 3.2.1.2 genauer betrachtet.

- *Simplex-Verfahren*

  Das Simplex-Verfahren ist das am häufigsten verwendete Verfahren zur Lösung von linearen Optimierungsproblemen. Es wird aufgrund der schnellen und einfachen Lösungsfindung häufig eingesetzt. Die Grundidee des Simplex-Verfahrens ist es, sich schrittweise von einer Ecke des Polyeders zu seiner benachbarten zu hangeln, bis es keinen besseren Nachbarn mehr gibt. Da es sich bei der linearen Optimierung um ein konvexes Optimierungsproblem handelt, ist die so gefundene optimale Ecke dann auch global, d.h. es gibt im zulässigen Lösungsraum keine andere Ecke mit einem besseren Zielfunktionswert. Für die Darstellung des genaueren Vorgehens wird auf Neumann (2002), Zimmermann (2008) und Unger (2010) verwiesen.

- *Dualität*

  Eine Optimierungsaufgabe ist vollständig durch ihre Daten - Zielfunktions-koeffizienten, Koeffizientenmatrix und rechte Seite der Nebenbedingungen - charakterisiert. Mit denselben Daten kann ein weiteres Optimierungsproblem konstruiert werden, das wichtige zusätzliche Informationen zum Ausgangsproblem liefert. Die Untersuchung dieses Paares von Aufgaben und der zwischen ihnen bestehenden Zusammenhänge ist Gegenstand der Dualität [vgl. ZIM08 S.92 ff.].

- *Innere-Punkte-Methode*

  Innere-Punkte-Verfahren zeichnen sich durch bessere theoretische Eigenschaften (polynomiale Komplexität) und schnellere Konvergenz für sehr große Probleme aus. Ein Nachteil ist, dass sie vergleichbar ungeeignet zum Lösen einer Serie von Optimierungsproblemen sind [vgl. UNG10 S.59 ff.].

3.2.1.2 Grafische Lösung eines 2-dimensionalen LP-Modells

Bevor nun die Lösungsalgorithmen für komplexe Optimierungsmodelle aufgestellt werden, soll zunächst der einfache Fall der grafischen Lösung eines linearen Optimierungsmodells betrachtet werden. Anhand dieses Vorgehens können grundlegende Vorgehensweisen der Optimierung verständlich gemacht werden, da es sich um sehr kleine Probleme mit wenigen Entscheidungsvariablen und wenigen Restriktionen handelt. Daher auch einfaches oder sehr kleines Modell genannt.

Ein lineares Optimierungsmodell mit nur zwei Variablen definiert also einen zulässigen Entscheidungsbereich. Die Restriktionen bilden bei einem linearen Modell je eine Gerade in der Ebene und definieren somit einen zulässigen Bereich in der Form eines zweidimensionalen Polyeders (ausgenommen der Bereich ist beschränkt). Projiziert man die Variablen auf beide Achsen, kann ein zweidimensionales Modell grafisch mit Hilfe einer Zeichnung gelöst werden [vgl. UNG10 S.9 ff.].

Ein einfaches Beispiel soll die Vorgehensweise einmal verdeutlichen. Ein Unternehmen stellt zwei verschiedene Frühstücksflocken A (Fruchtringe) und B (Schokoflocken) mit einem Deckungsbeitrag von 2,00 € bzw. 1,50 € je kg her. Die Herstellung eines Kilogramms der Fruchtringe benötigt doppelt so viel Zeit bei seiner Herstellung wie die Schokoflocken. Falls nur Schokoflocken produziert würden, könnte das Unternehmen 1000 kg pro Tag fertigen. Die Zutatenbelieferung erlaubt jedoch nur die Produktion von 800 kg Frühstücksflocken pro Tag (Typ A und B zusammen). Für A- und B-Flocken sollen verschiedene Verpackungen verwendet werden. Es stehen täglich 400 Verpackungen vom Typ A und 700 Verpackungen vom Typ B zur Verfügung. Eine sich in der Realität ergebene Fragestellung wäre, wie viele Frühstücksflocken vom Typ A und Typ B müssen produziert werden, um den maximalen gesamten Deckungsbeitrag zu erzielen.

Modellierung des LP-Modells

Entscheidungsvariablen:

x_1*: Anzahl der zu produzierenden Frühstücksflocken vom Typ A*

x_2*: Anzahl der zu produzierenden Frühstücksflocken vom Typ B*

Zielfunktion

Maximierung des gesamten Deckungsbeitrags in €, also

$$max\ z = 2x_1 + 1{,}5x_2 \qquad [3.6]$$

Nebenbedingungen

Zutatenbelieferung $x_1 + x_2 \leq 800$ [3.7]

Verpackungen $x_1 \leq 400,\ x_2 \leq 700$ [3.8], [3.9]

Zeitrestriktion Ein Produkt A benötigt 2t Zeiteinheiten pro Kilogramm; ein Produkt vom Typ B nur 1t Zeiteinheiten pro Kilogramm, und es stehen nur 1000t Zeiteinheiten pro Tag zur Verfügung, also

$$x_1(2t) + x_2(1t) \leq 1000t \qquad [3.10]$$

und durch t dividiert (t > 0)

$$2x_1 + x_2 \leq 1000 \qquad [3.10a]$$

Nichtnegativität $x_1, x_2 \geq 0$ (negative Produktionsmengen ergeben keinen Sinn) [3.11]

Daraus folgt das LP-Modell

Zielfunktion

$$\max\ z = 2x_1 + 1{,}5x_2 \qquad [3.6]$$

Nebenbedingungen

$$2x_1 + x_2 \leq 1000\ (a) \qquad [3.10a]$$

$$x_1 + x_2 \leq 800\ (b) \qquad [3.7]$$

$$x_1 \leq 400\ (c) \qquad [3.8]$$

$$x_2 \leq 700\ (d) \qquad [3.9]$$

$$x_1, x_2 \geq 0\ (e) \qquad [3.11]$$

Zur grafischen Lösung des Problems werden folgende Schritte durchlaufen. In Abbildung 3.3 ist die grafische Lösung dargestellt. Die Restriktionsgeraden (gleich statt kleiner- oder größer-gleich) des LP-Problems (Koordinaten zweier Punkte auf einer Geraden bestimmen, dann Gerade zeichnen) sind zu zeichnen. Diese sind die Randgeraden der durch die Restriktionen dargestellten Halbebenen. Durch Einsetzen der Koordinaten eines nicht auf der Gerade liegenden Punktes in die Restriktionsgleichung wird die Halbebene immer richtig bestimmt [vgl. NEM08 S.36 ff.]. Der Durchschnitt der Halbebenen ist der zulässige Bereich. In diesem Fall läuft die Gerade zur Restriktion (a) durch die Punkte (0, 1000) und (500, 0). Da Punkt (0, 0) die Ungleichung (a) erfüllt, wird diese durch die Halbebene unterhalb der Gerade dargestellt.

In Schritt zwei wird die Richtung der Zielfunktion bestimmt. Lösungen gleichen Wertes liegen auf sogenannten Isogewinn-Hyperebenen (im 2-dimensionalen Fall Isogewinngeraden) [vgl. SUM09 S.36 ff.]. Der Wert der Zielfunktion ist auf eine Konstante zu setzen $z = z_0$, damit die Isogewinnlinie gezeichnet werden kann. In diesem Fall ist z.B. $z_0 = 600$ zu wählen und die Isogewinngerade $2x_1 + 1{,}5x_2 = 600$ kann gezeichnet werden.

Der dritte Schritt ist die Verschiebung der Isogewinngeraden parallel zu einer optimalen Ecke. Für das Beispiel wird die Isogewinngerade parallel nach oben bewegt, bis die optimale Ecke C mit $x_1 = 200;\ x_2 = 600$ erreicht ist. Der optimale Zielfunktionswert ist in Gleichung 3.12 beschrieben.

$$z = 2 * 200 + 1{,}5 * 600 = 1300. \qquad [3.12]$$

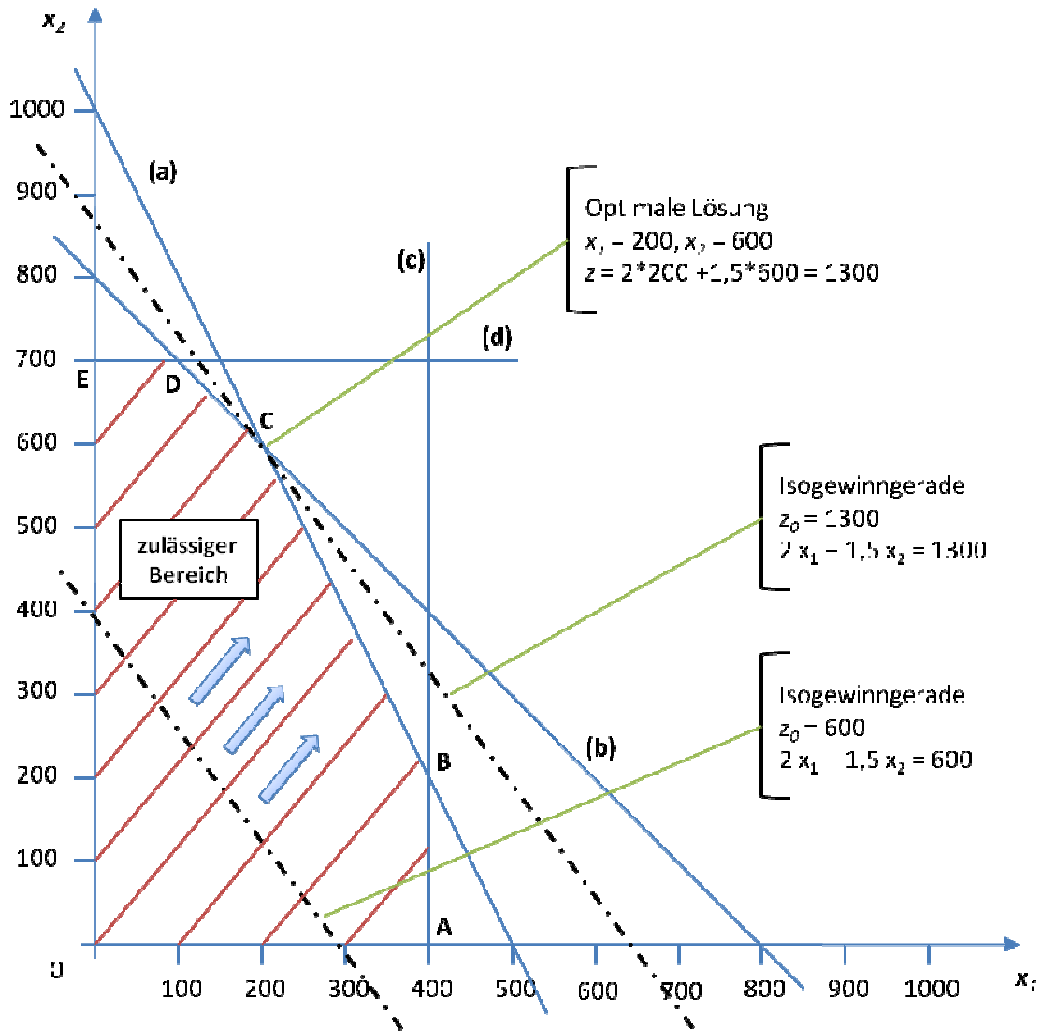

Abbildung 3.3: Grafische Lösung eines 2-dimensionalen LP-Modells [Eigene Darstellung]

3.2.1.3 Eigenschaften des zulässigen Bereichs

Ein Bereich $S \subseteq R^n$ heißt konvex, falls für jede zwei Punkte $X, Y \in S$ alle Punkte auf der geradlinigen Verbindung zwischen X und Y auch in S liegen [vgl. NUM02 S.43 ff.]. In Abbildung 3.4 sind Beispiele für konvexe und nichtkonvexe Punktemengen gegeben. Der zulässige Bereich S ist durch die rote Schraffur oder blaue Füllung dargestellt sowie die zu betrachtenden Punktemengen durch die Gerade XY. Im Allgemeinen ist der zulässige Bereich eines LP-Problems mit n Variablen konvex (nicht Typ 6 oder 7) und linear bzw. geradlinig abgegrenzt (nicht Typ 4 oder 5). Falls der zulässige Bereich beschränkt, d.h. von allen Richtungen eingegrenzt ist (Typ 1 und Typ 3, nicht Typ 2) heißt er konvexer Polyeder, der durch die Ecken des zulässigen Bereiches aufgespannt wird [vgl. SUM09 S.38]. In Abbildung 3.3 wird der zulässige Bereich durch die Verbindungsstrecken 0A, AB, BC, CD,

DE und E0 eingegrenzt. Somit ist der durch 0, A, B, C, D und E aufgespannte Polyeder konvex.

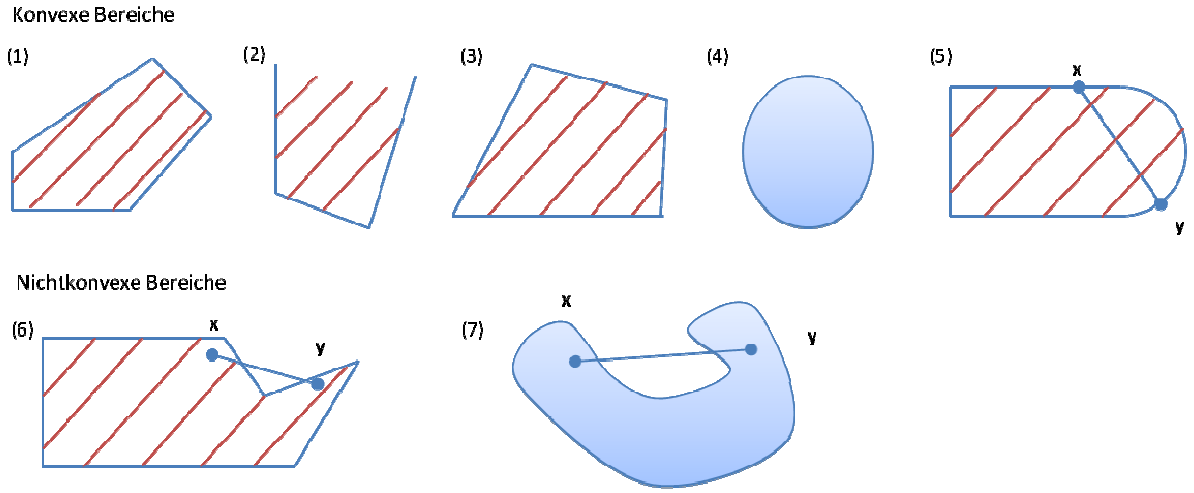

Abbildung 3.4: Konvexe und nichtkonvexe Bereiche [vgl. SUM09 S.38]

Viele Fälle der LP-Modelle sind jedoch nicht mit einer eindeutigen Lösung versehen. Diese Fälle kommen häufig in der Praxis vor. Modelle ohne zulässige Lösungen bilden beispielweise eine leere Schnittmenge, da der durch die Restriktionen zulässige Bereich leer ist. Unbeschränkte Modelle verschieben die Isogewinnlinie unendlich weit, so dass sich die Lösungen dabei grenzenlos verbessern. In diesem Fall gibt es also keine endliche Optimallösung. Die Form 2 in Abbildung 3.4 stellt so einen Fall dar. Falls bei der grafischen Lösung die Isogewinngerade am Ende des Verschiebevorganges zwischen zwei Ecken liegt, so besitzt das Modell unendlich viele gleichwertige Lösungen. Das Modell ist dann mehrdeutig. Zur Lösung von Problemen dieser Art bedient man sich mathematischen Verfahren, wie beispielsweise der Simplex-Methode. Für die Darstellung des genaueren Vorgehens wird, wie oben auf Neumann (2002), Zimmermann (2008) und Unger (2010) verwiesen.

3.2.2 Ganzzahlige und kombinatorische Optimierung

Viele in der Praxis auftretende Probleme sind nur mit ganzzahligen Variablen sinnvoll zu lösen, da eine Teilbarkeit z.B. von Ressourcen oft nicht gegeben ist. Rein-ganzzahlige (Integer Programming, IP) oder gemischt-ganzzahlige (Mixed Integer Programming, MIP) Optimierung unterscheidet sich von der (kontinuierlichen) linearen Optimierung nur durch zusätzliche Bedingungen. Es unterliegen hier einige oder alle Variablen der sogenannten

Ganzzahligkeitsbedingung. Die Zielfunktion und die Restriktionen sind linear, die Variablen sind allerdings nicht mehr kontinuierlich sondern diskret. Das bedeutet, dass es sinnlos ist den Variablen beliebige reelle Werte zuzuordnen. Es ist beispielsweise notwendig Menschen, Fahrzeugen oder Maschinen ganzzahlige Mengen bestimmter konkurrierender Möglichkeiten zuzuordnen. Außerdem lassen sich binäre ja/ nein-Entscheidungen über die Durchführung potentieller Vorhaben mit Hilfe von 0/1-Variablen darstellen, wobei logistische Abhängigkeiten von Vorhaben mit dieser speziellen Variablen als lineare Restriktionen darstellbar sind [vgl. SUM09 S.9 f.]. Ganzzahlige Variablen und damit ganzzahlige Optimierungsprobleme treten also auf, wenn kein Kontinuum von Alternativen vorliegt. Ist speziell die Menge der möglichen Alternativen und damit der zulässige Bereich endlich, spricht man von einem kombinatorischen Optimierungsproblem. Die Kombinatorik befasst sich mit den verschiedenen Anordnungen endlich vieler Objekte, z.B. den möglichen Permutationen endlich vieler Zahlen. Bewertet man diese Anordnungen und liegt damit eine Zielfunktion fest, so liegt ein kombinatorisches Optimierungsproblem vor [vgl. NEM02 S.380 f.].

Praxisrelevante kombinatorische Optimierungsprobleme, wie die kostenminimale Belegung von Maschinen mit Jobsequenzen, lassen sich mit Hilfe der ganzzahlig-gemischten Optimierung darstellen. Aus einer Menge von diskreten Elementen (Aufträgen, Maschinen) ist allgemein ein Konstrukt aus diesen Elementen, also eine Teilmenge (Reihenfolgebildung, Wegeplan), zu konstruieren. Diese Teilmenge muss gewisse Nebenbedingungen erfüllen und diesbezüglich einer Zielfunktion optimal sein. Obwohl der zulässige Bereich des Lösungsraums, d.h. der Bereich der keiner der Nebenbedingungen widerspricht, endlich und beschränkt ist, wächst die Anzahl alternativer Lösungen in der Regel exponentiell bezüglich der diskreten Elemente der Grundmenge [vgl. SUM09 S.9 f.]. Aus diesem Grund sind ganzzahlig-kombinatorische Optimierungsmodelle sehr schwer zu lösen. Die Schwierigkeit der Optimierungsmodelle wird durch die hohe Anzahl an kombinatorischen Lösungs-möglichkeiten hervorgerufen. Schon bei kleinen Modellen entsteht eine sehr hohe Zahl an Wertemöglichkeiten, da optimale Lösungen der LP-Relaxationen (MIP-Modelle ohne Ganzzahligkeitsforderungen) fraktionelle Werte aufweisen können. Daher wurden spezielle Suchstrategien der ganzzahligen und kombinatorischen Optimierung entwickelt. Lösungsmethoden sind das Branch-and-Bound Verfahren und Heuristiken (beide für kombinatorische Optimierungsprobleme) sowie das Verfahren von Gormy zur Lösung ganzzahliger Optimierungsprobleme [vgl. NEM02 S.380 ff.].

3.3 Einführung in die Maschinenbelegungsplanung

Nahezu alle Produktionssysteme der heutigen Zeit setzen sich als Ziele minimale Bestände (work in process) bei hoher Liefertreue (on time delivery) und gleichzeitiger Auslastung aller Ressourcen sowie sehr kurze Durchlaufzeiten zu erreichen [vgl. HOS01 S.488]. Es ist schnell ersichtlich, dass diese Ziele konträr sind und zu einander im Konflikt stehen. So ist es sehr viel einfacher Aufträge in der vorgegebenen Zeit fertigzustellen, wenn die Maschinenauslastung gering ist, da keine Engpässe im System entstehen können. Das Ziel der Maschinenbelegungsplanung ist es nun eine, im Hinblick auf die gewichteten Ziele, optimale Balance zwischen den gegensätzlichen Zielen herzustellen.

In diesem Kapitel sollen zunächst die Grundlagen der Maschinenbelegungsplanung vorgestellt werden. Im Anschluss daran folgt Einführung in die in der Literatur gängige Notation.

Die Maschinenbelegungsplanung war während der letzten Jahrzehnte Gegenstand intensiver Forschung. Eine ausführliche Darstellung von betrachteten Modellen und Lösungsverfahren liefern unter anderem Blazewicz et. Al (1996), Dempster et al. (1982), Domschke et al. (1997) und Pinedo (1995).

3.3.1 Grundlagen der Maschinenbelegungsplanung

Die Maschinenbelegungsplanung, auch Maschinenscheduling genannt, befasst sich mit der optimalen Einplanung von Aufträgen oder Jobs, die auf gewissen Maschinen zu bearbeiten sind. Maschinenbelegungsprobleme treten vor allem im Fertigungsbereich auf. Die zentrale Aufgabe der Maschinenbelegungsplanung ist es, zu ermitteln wann welche Jobs auf welchen Maschinen zu bearbeiteten sind, so dass eine bestimmte Zielfunktion minimal bzw. maximal wird [vgl. DOD02 S.110 ff.]. Ein mögliches Kriterium für die Zielfunktion ist die Zeitspanne bis zum Ende der Bearbeitung des letzten Jobs oder eine Zielfunktion bei der die Minimierung der mittleren Zeit, die ein Auftrag auf seine Bearbeitung warten muss, zu optimieren ist. Hierbei zu berücksichtigende Restriktionen können etwa vorgegebene Bereitstellungstermine, d.h. früheste mögliche Anfangszeiten, oder nicht überschreitbare Fertigstellungstermine für gewisse Jobs sein. Auch eine vorgeschriebene Bearbeitungsreihenfolge für einzelne Aufträge ist denkbar. Ferner ist zu unterscheiden, ob die Bearbeitung eines Jobs unterbrochen und später wieder fortgesetzt werden kann oder ob die Jobs nicht unterbrechbar sind [vgl. HOS01 S.488 ff.].

Typisch für die Maschinenbelegungsplanung ist die Vielfalt der Optimierungsprobleme, die sich durch Kombination unterschiedlicher Maschinenkonfigurationen, Zielfunktionen und Restriktionen ergeben [vgl. NEM02 S.474 f.]. Zur Lösung dieser Optimierungsprobleme sind verschiedenartige Lösungskonzepte entwickelt worden, die meist auf einen speziellen Problemtyp zugeschnitten sind. Charakteristisch für Schedulingaufgaben ist auch, dass eine kleine Abänderung der Problemstellung (z.B. Nichtunterbrechbarkeit oder die Vorgabe von Bereitstellungsterminen für einige Jobs) häufig ein leichtes (polynominal lösbares) Problem in ein schweres überführt [LEC99 S.5 ff.].

Um die Vielzahl verschiedener Schedulingprobleme zu klassifizieren, hat sich in der Literatur zur Maschinenbelegungsplanung ein weitgehend einheitliches Klassifikationsschema durchgesetzt [vgl. NEM08 S.475 ff.]. Es wird generell angenommen, dass n Jobs, von 1 bis n durchnummeriert, auf m Maschinen $M_1, \dots, M_m$ zu bearbeiten seien. Dabei gilt stets, dass zu einem Zeitpunkt auf einer jeden Maschine höchstens ein Job und ein jeder Job höchstens auf einer Maschine bearbeitet werden kann. Für ein Schedulingproblem können folgende Daten gegeben sein [vgl. PIN08 S.13 ff.]:

- *Bearbeitungsdauer* (engl. processing Time) p_{ij} für Job j auf Maschine M_i $(i = 1, \dots, m; j = 1, \dots, n)$
- *Bereitstellungstermin* (engl. release Date) r_j für Job j, d.h. der Zeitpunkt, ab dem Job j bearbeitet werden kann
- *Fälligkeitstermin* (engl. due date) d_j für Job j, d.h. der Zeitpunkt, bis zu dem die Bearbeitung von Job j abgeschlossen sein sollte
- *Gewicht* (engl. weight) w_j für Job j

Das Gewicht w_j gibt an, wie wichtig oder dringend Job j im Vergleich zu den übrigen Jobs ist. $w_j > w_k$ gilt genau dann, wenn Job j wichtiger oder dringender als Job k ist. Die Größen p_{ij}, r_j, d_j und w_j seien stets nichtnegative ganze Zahlen. Ist für Job j kein Bereitstellungstermin $r_j > 0$ gegeben, so sei der frühestmögliche Bearbeitungstermin der Zeitpunkt 0.

Sind die Zeitintervalle in denen die Jobs $1, \dots, n$ auf den einzelnen Maschinen bearbeitet werden festgelegt, so wird von einem Bearbeitungsplan gesprochen [vgl. WIE08 S.326 f]. Ein Plan heißt zulässig, wenn er allen Restriktionen des Schedulingproblems genügt, und

optimal, wenn er zulässig ist und die Zielfunktion minimiert [vgl. PIN08 S.13 ff.]. Eine Jobreihenfolge (engl. sequence) auf einer Maschine bedeutet die Bearbeitungsreihenfolge der Jobs auf der Maschine. Jeder Jobreihenfolge entspricht einer Menge von möglichen Plänen (bezogen auf eine Maschine), bei denen die Anfangs- und Endzeitpunkte der Bearbeitung jedes Jobs festgelegt sind [vgl. NEM02 S.475 f.]. Zur Veranschaulichung eines Plans werden Balken- oder Ganttdiagramme verwendet [vgl. LÖD08 S.90 f.]. Jeder Maschine entspricht genau ein Balken über der Zeitachse. In jedem Balken sind die Zeitintervalle, in denen die einzelnen Jobs auf der betreffenden Maschine bearbeitet werden, eingetragen. Abbildung 3.5 zeigt einen Plan für eine Maschinenbelegung mit vier Aufträgen und drei Maschinen. Job eins beispielsweise auf Maschine M_2 im Zeitintervall [0,2], auf M_1 im Zeitintervall [2,5] und auf M_3 im Zeitintervall [8,10] bearbeitet wird.

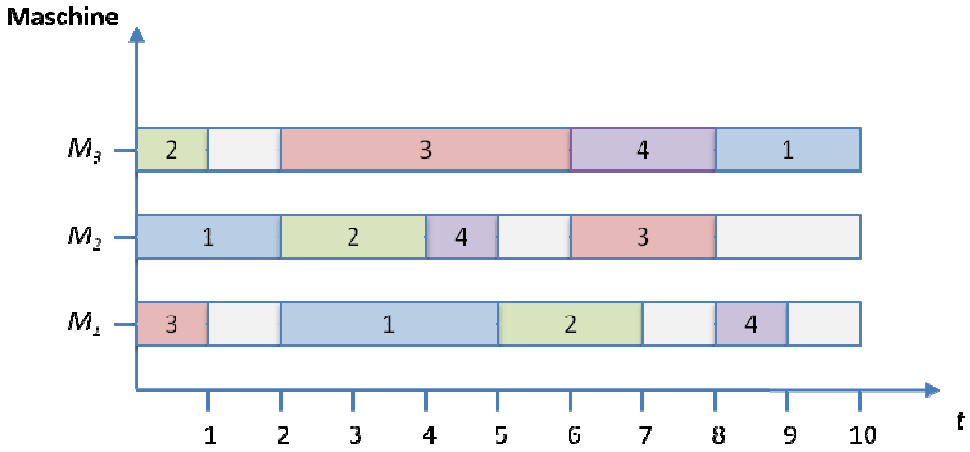

Abbildung 3.5: Eine Maschinenbelegungsplanung mit vier Aufträgen und drei Maschinen [Eigene Darstellung]

Weitere Randbedingungen, die bei der Maschinenreihenfolgeplanung gelten sind [vgl. HOS01 S.488 f.]:

- *Alle Jobs stehen zum Start der Periode zur Verfügung,*
- *Prozesszeiten sind deterministisch,*
- *Maschinenausfälle kommen nicht vor,*
- *es gibt keine Unterbrechungen der Aufträge, d.h. wird ein Auftrag gestartet, so muss er auch beendet werden und*
- *bestehende Aufträge werden nicht abgebrochen.*

3.3.2 Notation und Resultate der Maschinenbelegungsplanung

Zur Kurzbeschreibung von Schedulingsproblemen ist eine Notation eingeführt worden, die im Folgenden kurz vorgestellt werden soll [vgl. PIN08 S.13 ff.]:

Ein-Maschinen-Problem

Ein-Maschinen-Probleme sind aus vielerlei Hinsicht wichtig für die Maschinenreihenfolgeplanung. Das Ein-Maschinen-Problem ist sehr einfach im Vergleich zu komplexen Multimaschinenproblemen, da nur auf eine Maschine geachtet werden muss und das Optimierungsmodell folglich nicht so komplex ist. Trotz der Reduzierung auf ein Ein-Maschinen-Problem kann, basierend auf der Problematik des Problems, die Lösung auf ein System mit beliebig vielen Maschinen projiziert werden. So ist es sinnvoll das Schedulingproblem auf ein Subproblem zu übertragen und im Anschluss daran Lösungen für ein ganzes System abzuleiten [vgl. PIN08 S.35 ff.]. Aufgrund der Komplexität und der damit verbundenen Schwierigkeit Schedulingprobleme zu lösen, ist es also ratsam das Problem weiter einzugrenzen und es in kleine Stücke aufzuteilen. Diese Möglichkeit impliziert die Reihenfolgebildung auf einer Maschine, also dem Ein-Maschinen-Problem und nimmt nur die Bildung von Auftragssequenzen auf einer Maschine vor, während der Maschinenbelegungsplan (Scheduling) sagt, wann ein Job j auf einer Maschine M_i bearbeitet werden muss. Die so betrachtete Auftragsmenge ist also kleiner als die des gesamten Systems. Das Scheduling wird nur aus der optimalen Sequenz auf einer Maschine abgeleitet, da nicht für das gesamte System eine Maschinenbelegungsplanung erfolgt. Die Betrachtungsweise einer Maschine stellt also einen Spezialfall dar, der sich auf viele Situationen anwenden lässt. Somit bilden sie die Grundlage für komplexere Schedulingaufgaben [vgl. PIN08 S.35 f.]. Beispielsweise lässt sich ein Multimaschinenproblem mit Hilfe einer Engpassuntersuchung vereinfachen. Ist der Engpass eines Systems gefunden, so gibt die optimale Reihenfolge auf dem Engpass die Systemleistung vor, da so der Durchsatz maximal ist [vgl. GLR05 S.22 f.]. In diesen Fällen ist es sinnvoll die Reihenfolge auf dem Engpass zuerst festzulegen und alle restlichen Reihenfolgen in Abhängigkeit dieser Reihenfolge auf dem Engpass im Anschluss zu bilden [vgl. HOS01 S.513 ff.]. So wird für die Stationen vor dem Engpass eine Rückwärtsterminierung und für die Stationen nach dem Engpass eine Vorwärtsterminierung durchgeführt [vgl. GLR05 S.22]. Diese Vorgehensweise der Komplexitätsreduktion gibt also vor ein Multimaschinenproblem auf ein Ein-

Maschinen-Problem zu reduzieren und dann Lösungen auf das gesamte System abzuleiten.

Mehrere parallele Maschinen

Viele Produktionen in der Realität bestehen aus mehreren Arbeitsstationen oder Abteilungen, die wiederum aus einer bestimmten Anzahl von technisch gleichartigen oder identischen Maschinen bestehen [vgl. PIN08 S.111 f.]. Man kann sich eine klassische Fertigung mit einer Dreherei, Fräserei und Schleiferei vorstellen. Diese Abteilungen bestehen in vielen Fällen aus identischen Maschinen oder solchen, die den gleichen Anforderungen genügen. Wenn Job j nun auf einer Maschine M_i der Schleiferei bearbeitet werden soll, so kann aus der Menge aller zur Verfügung stehenden Maschinen der Schleiferei frei gewählt werden. Eine Ausnahme bildet hier die Tatsache, dass eine Maschine gleichen Typs älter ist und somit andere Bearbeitungszeiten generiert oder sich in einem schlechteren Zustand befindet. In diesem Fall wird eine zusätzliche Variable eingefügt, die die Bearbeitungszeit im Hinblick auf die Prozesszeit p_{ij} korrigiert [vgl. PIN08 S.14], um zu im Sinne der Zielfunktion optimalen Ergebnissen zu kommen.

Job-Shop- und Flow-Shop-Scheduling

Ein Flow-Shop-Problem bezeichnet ein System mit m Maschinen in Serie. Jeder Job j folgt der gleichen Bearbeitungsreihenfolge auf jeder der m Maschinen, d.h. alle Jobs folgen der gleichen Reihenfolge [vgl. PIN08 S.15f]. Beispielweise wird jeder Job zuerst auf Maschine M_1 danach auf M_2 usw. gefertigt. Nach der Bearbeitung auf einer Maschine wird der Job in die Pufferstation der nächsten Maschine weitergeleitet, i.d.R. erfolgt die Bearbeitung nach dem FIFO-Prinzip[12]. Ein Beispiel aus der Industrie stellt hier die Serienfertigung dar. Hier sind in der Regel immer die gleichen Arbeitsstationen in der gleichen Reihenfolge zu durchlaufen.

Ein Job-Shop-Problem bezeichnet ein Szenario, in dem jeder Job j einer festen Route oder Bearbeitungsreihenfolge auf m Maschinen folgt [vgl. Ebenda S.15]. Es wird unterschieden in Probleme in denen ein Job j nur einmal auf einer Maschine m bearbeitet wird und zwischen Problemen in denen ein Job j mehrmals auf einer

[12] FIFO bedeutet First-in-First-out und besagt, dass Aufträge in der Reihenfolge abgearbeitet werden, in der sie bei einem Arbeitssystem erscheinen [vgl. LÖD05 S.445].

Maschine m bearbeitet werden kann. Letzteres ist für den klassischen Maschinen- und Anlagenbau sicherlich zutreffender. Die Arbeitspläne verweisen hier öfter auf dieselbe Maschine, aber zu verschiedenen Zeitpunkten im Arbeitsplan. So kommt es, wie auf Seite 27 oben beschrieben, zu sehr komplexen und rechenaufwändigen Optimierungsmodellen.

α : Maschinenkonfiguration

Es werden Ein-Maschinen-Probleme, Probleme mit parallelen Maschinen und Shop-Probleme unterschieden. Bei Einmaschinenproblemen und Problemen mit parallelen Maschinen besteht jeder Job aus genau einem Arbeitsgang. Bei den Shop-Problemen dagegen setzt sich jeder Job aus mehreren Operationen (Arbeitsgängen) zusammen, die auf verschiedenen Maschinen ausgeführt werden. Je nach Vorschrift, in der die Maschinen zu durchlaufen sind, wird zwischen den folgenden Modellen unterschieden:

J	Job-Shop-Probleme vorgegeben)	(Reihenfolge für jeden Job fest)
F	Flow-Shop-Probleme	(Reihenfolge für jeden Job identisch)
O	Open-Shop-Probleme	(keine Reihenfolge vorgegeben).

β : Jobeigenschaften

Das zweite Feld in der Notation beschreibt zusätzliche Vorgaben eines gegebenen Schedulingproblems. Dieses Feld kann mehrere Einträge enthalten. Mögliche Einträge sind beispielsweise die folgenden:

r_j	Für jeden Job j sind Bereitstellungstermine $r_j > 0$ gegeben
$prec$	Zwischen den Jobs bestehen Anordnungsbeziehungen (precedence constraints)

Die folgenden beiden Jobeigenschaften werden ausschließlich bei Flow-Shop-Problemen untersucht:

$block$	Die Lagerkapazität zwischen zwei aufeinanderfolgenden Maschinen ist begrenzt

$nowait$ Sobald die Bearbeitung eines Jobs begonnen hat, muss dieser Job ohne Unterbrechung durchgängig bearbeitet werden. Eine Wartezeit zwischen zwei Maschinen ist nicht erlaubt.

γ : Zielfunktion

In diesem Feld wird die zu minimierende Zielfunktion beschrieben. Für einen gegebenen Maschinenbelegungsplan ist für jeden Job j eine Completion-Time (Fertigstellungszeitpunkt) C_j durch Beendigung der letzten zu bearbeitenden Operation festgelegt. Sind zusätzlich noch Due-Dates (Fälligkeitstermine) $d_j \geq 0$ gegeben, so ist die Lateness (Verspätung) von Job j definiert als [vgl. HOS01 S.488 f.]:

$$L_j \coloneqq C_j - d_j. \qquad [3.13]$$

Die Verspätung ist positiv, wenn der Job erst nach seinem Fälligkeitstermin beendet ist und negativ, wenn der Job zu früh fertig ist. Die Tardiness (Terminüberschreitung) wird definiert als [vgl. HOS01 S.488 f.]:

$$T_j \coloneqq max\{L_j, 0\} \qquad [3.14]$$

Das Modell $min \sum T_j$ erfuhr während der letzten Jahrzehnte eine besondere Aufmerksamkeit in der Literatur, da es bis 1990 aufgrund seiner Komplexität nicht in annehmbarer Zeit zu lösen war [vgl. PIN09 S.51]. Die Besonderheit des Modells liegt in der hohen Erreichung der Ziele der Produktionslogistik (vgl. dazu auch Kapitel 2 und Kapitel 3.3). Man kann sich leicht vorstellen, dass nur die Minimierung der Anzahl der verspäteten Jobs (vgl. Formel 3.13) alleine nicht eine optimale Lösung liefern kann. Es kann vielmehr dazu kommen, dass ein Auftrag unakzeptabel lang vor einer Bearbeitungsstation warten muss und hohe Bestände sowie hohe Durchlaufzeiten verursacht. Die Minimierung der Tardiness (Terminüberschreitung) über alle Aufträge hingegen, impliziert dass alle Aufträge gleich gewichtet werden und keiner von Ihnen lange vor einer Bearbeitungsstation warten muss. Dieses spricht für relativ kurze Durchlaufzeiten bei gleichzeitiger Zielerreichung einer optimalen Auslastung, da alle Aufträge mit einem konkreten Kundenbedarf in der vorgesehenen Zeit fertiggestellt werden. Somit lässt sich mit der Zielfunktion $\min \sum T_j$ eine optimale Reihenfolge, die einen hohen Durchsatz und Liefertermintreue erfüllt, generieren.

4 Entwicklung eines Lösungsansatzes zur Berücksichtigung der Arbeitsgangreihenfolge

Nachdem in Kapitel zwei die Durchsatzkennlinientheorie anhand des DePlaVis-Algorithmus vorgestellt wurde, gilt es nun, aufbauend auf den Erkenntnissen des dritten Kapitels, einen Lösungsvorschlag zur Berücksichtigung der Arbeitsgangreihenfolge zu entwickeln. Zunächst wird in grundsätzlichen Überlegungen auf das Ziel der Maximierung des Durchsatzes unter Berücksichtigung der Reihenfolgebildung eingegangen. Im Anschluss daran wird ein neuer Algorithmus formuliert und anhand eines Beispiels ausführlich erläutert. Danach wird mit Hilfe eines Flussdiagrammes ein Implementierungsvorschlag in den bestehenden DePlaVis-Algorithmus entwickelt. Ein in Matlab programmierter Demonstrator wird in Kapitel 4.5 vorgestellt. In dem letzten Kapitel werden die Ergebnisse bewertet.

4.1 Allgemeine Überlegungen zur Arbeitsgangreihenfolgebildung

In Kapitel 2.2 konnte erläutert werden warum es wichtig ist die Arbeitsgang-reihenfolgebildung zu berücksichtigen. So kommt es aufgrund der diskreten Arbeitsinhalte der einzelnen Aufträge zu Wartezeiten vor den Bearbeitungsstationen. Ein Auftrag blockiert während seiner Bearbeitung die jeweilige Arbeitsstation für die nachfolgenden Aufträge. Kommen weiterhin Aufträge an der Arbeitsstation an, so bildet sich eine Warteschlange. Die Warteschlange behindert und stoppt ggf. den Materialfluss. Dieses Verhalten tritt genau dann ein, wenn zwei Materialströme auf einer Arbeitsstation durchgesetzt werden müssen oder die Arbeitsinhalte verschieden groß sind. Nach Goldratt kann ein Produktionssystem nur maximal so viel durchsetzen wie seine beschränkenden Elemente, also die Engpässe eines Systems durchsetzen können [vgl. GOL08 S.5 ff.]. Der DePlaVis-Algorithmus begründet sich ebenfalls auf dieser Annahme [vgl. KRS09 S.2 ff.]. Die Folge aus der nicht Berücksichtigung der Arbeitsgangreihenfolge ist, dass der Durchsatz des Gesamtsystems unter Umständen also nicht maximal ist. Dies ist dann der Fall, wenn sich eine Warteschlange im System und damit auch vor dem Engpass bildet. Berücksichtigt man hingegen die diskreten Arbeitsinhalte und Fertigstellungszeitpunkte eines jeden Auftrags, so kann auf dieser Basis eine optimale Bearbeitungsreihenfolge abgeleitet werden. Alle in einer Periode relevanten Aufträge mit diskretem Arbeitsinhalt müssen somit berücksichtigt werden.

Die Bildung der optimalen Reihenfolge impliziert, dass die jeweiligen Fertigstellungstermine und die Bearbeitungstermine, d.h. die diskreten Inhalte, in die Reihenfolgebildung einfließen.

Wie oben gezeigt können nur auf diese Weise die Ziele der Produktionslogistik auch ganzheitlich erfüllt werden [vgl. WIE08 S.3 f.]. Die Fertigstellungstermine sind ein Kriterium für den Start- und Endzeitpunkt der Bearbeitung eines Auftrags. Die Bearbeitungszeiten hingegen dienen der Ermittlung der jeweiligen Einlastung auf einer Maschine und der Ermittlung der Fertigstellung des Auftrages. Liegen diese Zeiten vor, so kann eine Reihenfolgeplanung unter Optimierung einer Zielfunktion durchgeführt werden.

Ein weiteres Problem bei nicht Berücksichtigung der Arbeitsgangreihenfolge ist die Bildung von dynamischen Engpässen im Produktionssystem. Der bereits bekannte DePlaVis-Algorithmus stützt sich auf die Annahme, dass der größte Engpass auch die stärkste Auswirkung auf einen Materialstrom besitzt [vgl. GOL08 S.5 ff]. Der Durchsatz hängt demnach nur von der Abarbeitungswahrscheinlichkeit an der Durchsatzschranke ab. Die Aufträge im DePlaVis-Algorithmus werden zu Beginn einer Periode gestartet und nach dem FiFo-Prinzip durch das Produktionssystem geschleust. Wie in Kapitel 2.2 gezeigt kommt es so zu der Fragestellung, ob die Aufträge den größten Engpass des Systems überhaupt erreichen.

Die Annahme den Auftragsstrom als kontinuierlichen Fluss anzusehen stellt sich als weiteres Problem dar. Demnach lässt sich dieser in infinitesimal kleine Stücke zerlegen. So entstehen keine Wartezeiten vor den Bearbeitungsstationen. In der Praxis jedoch handelt es sich um eine endliche Anzahl an diskreten Aufträgen mit diskretem Arbeitsinhalt. Eine sich ergebende Problemstellung ist also, wann ein diskreter Auftrag gestartet werden soll, damit dieser den Engpass erreicht. Ebenso gilt es zu gewährleisten, dass der Auftrag auch abgearbeitet werden kann und fertiggestellt wird. Somit müssen die diskreten Arbeitsinhalte der Aufträge berücksichtigt werden, um eine möglichst realitätsnahe Abbildung zu erhalten.

Einen neuen Ansatz zur Lösung dieses Problems liefert die Eingrenzung des Gesamtsystems auf ein Ein-Maschinen-Problem [vgl. PIN08 S.35 ff.]. Dabei stellt der Engpass den Betrachtungsmittelpunkt, also das Ein-Maschinen-Problem, dar. Diese Betrachtungsweise ist legitim, weil der Engpass das durchsatzbestimmende Element des Gesamtsystems ist [vgl. GOL08 S.3 ff.]. Werden also die Aufträge, die auf dem Engpass durchgesetzt werden sollen in eine optimale Reihenfolge gebracht, so kann die Leistung des Gesamtsystems verbessert bzw. der Engpass voll ausgelastet werden. Ein Vergleich der sich so ergebenen Einlastung und der tatsächlichen Kapazität führt zu der Erkenntnis, welche Aufträge in einer Periode gestartet werden können und welche ggf. fremdvergeben

werden sollten. Die resultierenden freizugebenden Aufträge können nach der Theorie of Constraints von dem Gesamtsystem durchgesetzt werden, weil von freien Kapazitäten auf den Nicht-Engpassmaschinen auszugehen ist. Folglich können alle Aufträge, die den Engpass passieren, auch vom Gesamtsystem durchgesetzt werden. Sind die Aufträge, die innerhalb einer Periode freigegeben werden sollen, ermittelt, kann eine Rückwärts-terminierung für alle vorgelagerten Arbeitsstationen und eine Vorwärtsterminierung für alle nachgelagerten Arbeitsstationen vorgenommen werden. Es ergibt sich der Zeitpunkt zudem ein Auftrag auf dem Engpass bearbeitet und dieser den Engpass spätestens erreichen muss. Mittels der Rückwärtsterminierung kann der Startzeitpunkt für den Auftrag generiert werden. Es ist daher sinnvoll das Multimaschinenproblem auf das Ein-Maschinen-Problem einzugrenzen, da die Festlegung der optimalen Reihenfolge auf dem Engpass die Systemleistung vorgibt [vgl. GLR05 S.22 f.]. Im Anschluss daran wird ein Maschinenbelegungsplan für das gesamte System aus der Sequenz des Engpasses abgeleitet.

4.2 Formulierung des neuen Algorithmus unter Berücksichtigung der Arbeitsgangreihenfolge

In diesem Kapitel wird nun der neue Algorithmus zur Berücksichtigung der Arbeitsgangreihenfolge vorgestellt. Wie bei dem bekannten DePlaVis-Algorithmus stützt sich der neue Algorithmus auf die These, dass der größte Engpass auch die stärkste Auswirkung auf den Materialstrom hat (vgl. Kapitel 4.1). Ähnlich der ersten zwei Schritte des DePlaVis-Algorithmus 2.0 wird zunächst der maximale Belastungsquotient aller Arbeitsstationen bestimmt. Zur Erinnerung geschieht dieses mit Hilfe der Formel 2.5.

$$BQ_i = \frac{AF_{ij}}{K_{AS_i}} \qquad [2.5]$$

Nach Ermittlung des größten Systemengpasses gilt es die auf ihm eingeplanten Aufträge zu identifizieren. An dieser Stelle wird das Ein-Maschinen-Problem abgegrenzt. Nur die Aufträge, die über den größten Systemengpass laufen sind relevant für die Sequenzbildung [vgl. GOL08 S.3 ff.] Die Ausgangsbetrachtung des neuen Algorithmus besagt nun, dass alle auf dem Engpass geplanten Aufträge in eine optimale Reihenfolge zu bringen sind. Die Prüfung, welche Aufträge tatsächlich durchgesetzt und somit freizugeben sind, erfolgt erst nach der Reihenfolgebildung. Dieses ist sinnvoll, da die Reihenfolgebildung nicht bereits im

Vorfeld beeinflusst werden darf. Das Weglassen von Aufträgen ohne Berücksichtigung der Zielfunktion führt nicht zu optimalen Ergebnissen. Sind die Aufträge, die auf dem Engpass eingeplant sind, identifiziert kann mittels des neuen Algorithmus die optimale Reihenfolge berechnet werden. Das Ein-Maschinen-Problem stellt somit die Ausgangsbasis für den neu entwickelten Algorithmus dar.

Wie bereits in Kapitel 2 und 3 gezeigt, ist eines der Hauptziele in der Produktion die Liefertermine einzuhalten. Dieses resultiert üblicherweise aus zwei Quellen. Zum einen gibt der Kunde einen Wunschliefertermin vor und zum anderen generiert der Fertigungsprozess selbst einen Bereitstellungstermin [vgl. HOS01 S.488 f.]. Das Ziel des DePlaVis-Algorithmus ist es nun den maximalen Durchsatz an der Durchsatzschranke zu erzielen. Ein Auftrag, der einen frühen Liefertermin und ein kleines Arbeitsvolumen hat, sollte vor einem Auftrag mit späterem Liefertermin und großem Arbeitsvolumen gestartet werden. Der DePlaVis-Ansatz hingegen zielt an dieser Stelle auf den größten möglichen Durchsatz ab. Er empfiehlt den Auftrag mit einem größeren Arbeitsvolumen freizugeben. In der Praxis kann dieses jedoch so nicht gewünscht sein. Aufträge mit einem kleinen Volumen müssten ständig warten und bilden hohe Bestände am Engpass. Verbindet man jedoch die Ziele miteinander, eine hohe Liefertermintreue bei maximalem Durchsatz zu erreichen, so kann man folgenden Schluss ziehen. Im Sinne der Zielerreichung der Produktionslogistik sollten diejenigen Aufträge freigegeben werden, die einen Liefertermin einhalten müssen. Unter diesen Aufträgen gilt es dann die maximale Anzahl durchzusetzen. Das Optimierungsmodell $1 \parallel \sum T_j$ (Total Tardiness) ist, wie in Kapitel 3.3.2 gezeigt, besonders gut geeignet die beiden genannten Ziele miteinander zu vereinbaren [vgl. HOS01 S.488 f.]. Das Optimierungsmodell $1 \parallel \sum T_j$ optimiert die Summe der gesamt Tardiness, d.h. der Terminüberschreitungen einer bestimmten Auftragsreihenfolge [vgl. PIN08 S.50 ff.]. So kann es nicht dazu kommen, dass ein Auftrag, im Verhältnis zu anderen Aufträgen, besonders hohe Liegezeiten hat.

Die Eingangsdaten zur Bildung der optimalen Auftragssequenz beinhalten die Nummer des Auftrages j, die Bearbeitungsdauer p_j und den Fälligkeitstermin d_j. Folgende Grundannahme ist für diesen Algorithmus zu beachten. Wenn $p_j \leq p_k$ und $d_j \leq d_k$ gibt es eine optimale Sequenz, die Job j vor Job k plant. Um eine Reihenfolge zu berechnen wird das Grundprinzip des Branch-and-Bound-Verfahrens genutzt. Dieses Verfahren ist eine Methode, um eine optimale Lösung zu ermitteln. In der linearen und ganzzahlig-kombinatorischen Optimierung wird das Optimierungsproblem mittels des Branch-and-Bound-Verfahrens in immer neue Teilprobleme zerlegt (Branch) [vgl. ZIM08 S.230 ff.]. Diese

sind wiederrum mit Hilfe des Simplex-Algorithmus zu lösen. Mittels geeigneter Schranken (Bound) sollen viele suboptimale Lösungen frühzeitig erkannt und ausgesondert werden, so dass der zu durchsuchende Lösungsbaum kleingehalten wird.

Der zu entwickelnde Lösungsvorschlag bedient sich einer impliziten enumerativen Technik auf der Grundlage des Branch-and-Bound-Verfahrens. Sie ist sehr gut geeignet alle kombinatorischen Möglichkeiten einer Menge M zu überprüfen [vgl. NEM02 S.392 ff.]. Dies geschieht mit Hilfe des sogenannten Suchbaumes, d.h. Wurzelbaumes dessen Knoten immer Lösungen des Gesamtproblems, also einer Teilmenge von M entsprechen. Dabei bezieht sich „Branch“ auf das Verzweigen des Suchbaumes, wodurch neue Teilmengen generiert werden. „Bound“ weist auf die Verwendung unterer und oberer Schranken für Zielfunktionswerte hin, mit deren Hilfe man „uninteressante“ Teilmengen von M eliminieren kann. Bei einem Branch-and-Bound-Algorithmus speichert man neben dem Suchbaum, die „beste“ bisher gefundene zulässige Lösung ab. Diese Lösung ist diejenige mit dem kleinsten Zielfunktionswert. Nachdem alle Knoten auf einer Ebene des Suchbaumes untersucht wurden, ist nur an der Stelle mit dem kleinsten Zielfunktionswert der Baum weiter abzuschreiten. Alle anderen Zweige scheiden aus und werden nicht weiter betrachtet. Dieses Vorgehen wird bis zur optimalen Lösung aller Möglichkeiten durchgespielt. In Abbildung 4.1 ist ein Suchbaum exemplarisch dargestellt.

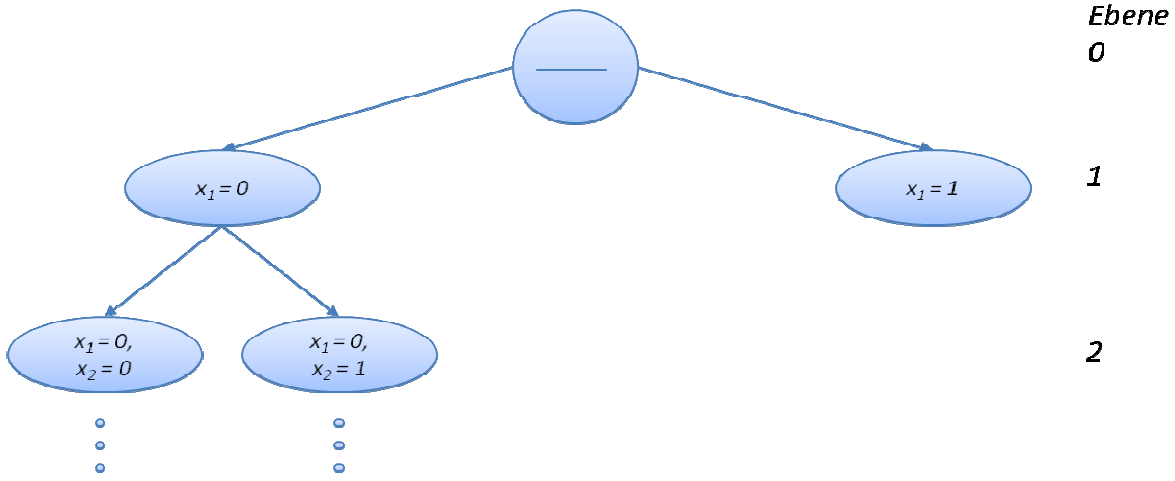

Abbildung 4.1: Branch-and-Bound-Verfahren [Eigene Darstellung]

Die Größe des Suchbaumes wird von der Anzahl der Kombinationsmöglichkeiten vorgegeben. Bei einer Anzahl von n Aufträgen existieren $n!$ mögliche Lösungs-kombinationen [vgl. HOS01 S.495 ff.]. Der Suchbaum muss demzufolge auch n Ebenen

untersuchen. Um nun die Reihenfolge mit der kleinsten Terminüberschreitung zu bestimmen muss folgende Zielfunktion erfüllt werden.

Zielfunktion

$$z = min \sum T_j \qquad [4.1]$$

Initialisierung

Zur Ermittlung der ersten Sequenz muss davon ausgegangen werden, dass es noch keine Reihenfolge gibt. Folglich sieht die erste Reihenfolge so aus (*,*,*,…,*). Die Sterne stehen hier für einen beliebigen Auftrag, der noch nicht in eine Reihenfolge gebracht wurde. Das bedeutet, dass kein Auftrag an einer bestimmten Stelle zu einer bestimmten Zeit bearbeitet werden muss.

Schritt 1

Die Festlegung einer ersten Reihenfolge beginnt an der ersten Stelle der n Aufträge. So werden alle möglichen Kombinationen überprüft. Dieses wären die Reihenfolgen (1,*,*,*,…,*), (2,*,*,*,…,*),…, ($n-1$, *,*,*,…,*) und (n,*,*,*,…,*).

Schritt 2

Um den zweiten Auftrag in der Sequenz zu ermitteln muss jeder optimale Ast des Suchbaumes weiter abgebildet werden. Auf dieser Ebene gilt es nun wieder für alle Kombinationen die optimale Lösung im Sinne der Zielfunktion 4.1 zu ermitteln. Dieses ist so lange durchzuführen, bis alle Aufträge in einer optimalen Reihenfolge sind. Ist beispielsweise die optimale Reihenfolge aus Schritt eins (3,*,*,*,…,*), so müssen alle Optionen (3,1,*,*,…,*), (3,2,*,*,…,*),…, (3, n-1,*,*,…,*) bis (3,n,*,*,…,*) durchgespielt werden. Dieser Prozess wird von einem Suchbaum dargestellt. Jeder Eintrag auf dem Suchbaum in einem Knoten gehört zu einer möglichen Reihenfolge. Diese Reihenfolge enthält geplante und ungeplante Aufträge. Eine vollständige Enumeration aller Aufträge beinhaltet nun, dass in jedem Schritt der tiefere Ast (engl. Lower Bound) berechnet wird.

Das Vorgehen für das Optimierungsproblem $\min \sum T_j$ und die partielle Reihenfolge ab Schritt k nachdem k Aufträge schon auf die ersten k Positionen zugewiesen wurden ist in Abbildung 4.2 dargestellt.

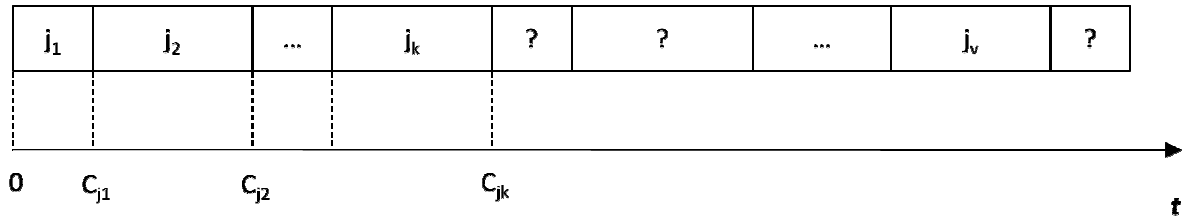

Abbildung 4.2: Bereits zugewiesene und noch nicht zugewiesene Aufträge [Eigene Darstellung]

Für die Aufträge, die auf die ersten k Positionen zugewiesen wurden berechnet sich die Tardiness nach der Formel 4.2.

$$T_{ju} = \max \{C_{ju} - d_{ju}, 0\} \qquad [4.2]$$

Für jeden weiteren Auftrag, der noch nicht auf eine bestimmte Reihenfolge zugewiesen wurde berechnet sich die Tardiness nach der Formel 4.3.

$$T_{jv} \geq \max \{C_{jv} - d, 0\} \qquad [4.3]$$

Die Tardiness T_{jv} kann also niemals weniger als $max\{C_{jv} - d, 0\}$ betragen. Daher ist d als der maximale Fertigstellungstermin aller nicht zugewiesenen Aufträge zu wählen.

$$d = \max \{d_{j_{k+1}}, d_{j_{k+2}}, \dots, d_{j_n}\} \qquad [4.4]$$

Das Minimum der Terminüberschreitung unter Berücksichtigung der Fertigstellungstermine kann gefunden werden, indem die nicht zugewiesenen Aufträge nach der SPT-Regel[13] sortiert werden. Die SPT-Regel dient der Reihenfolgebildung der noch nicht zugewiesenen Aufträge [vgl. HOS01 S.141 f.]. So werden die Aufträge mit der kürzesten Bearbeitungszeit

[13] SPT bedeutet shortest process time

zuerst bearbeitet. Die Verwendung der Regel dient dem einheitlichen Vorgehen bei allen möglichen Kombinationen. Die SPT-Regel hat sich in der Praxis bei der Maschinenbelegungsplanung sehr bewährt [vgl. NEM02 S.505]. Die Aufträge die noch nicht zugewiesen wurden, sind aufsteigend nach ihren Prozesszeiten zu sortieren. So kann für die Berechnung nach Formel 4.3 eine vorläufige Reihenfolge gebildet werden. Diese Reihenfolge dient also nur der vorläufigen Ergebnisfindung und stellt nicht die optimierte Reihenfolge dar. Erst nachdem alle Terminüberschreitungen der Aufträge mit der Formel 4.1 berechnet wurden, ist die optimale Reihenfolge festgelegt. Somit setzt sich die jeweilige gesamt Tardiness aus der Summe der einzelnen Terminüberschreitungen von zugewiesenen und nicht zugewiesenen Aufträge zusammen (siehe Formel 4.5). Zur Veranschaulichung ist das Vorgehen des Algorithmus in Abbildung 4.3 einmal dargestellt.

$$T = \underbrace{\left(T_{j1} + T_{j2} + \cdots + T_{jk}\right)}_{\text{zugewiesene Aufträge}} + \underbrace{\left(T_{j_{k+1}} + T_{j_{k+2}} + \cdots + T_{j_n}\right)}_{\text{nicht zugewiesene Aufträge}} \qquad [4.5]$$

Schritt 3

Nach Ermittlung der optimalen Reihenfolge auf dem größten Engpass des Systems erfolgt eine erneute Ermittlung der noch nicht zugewiesenen Aufträge. Die berechnete Reihenfolge muss zuerst in das System übertragen werden. Dabei werden die zu berücksichtigenden Aufträge auf alle ihnen zugewiesenen Maschinen eingeplant. Die benötigte Kapazität auf allen betroffenen Maschinen ist daraufhin anzupassen. Sind weitere noch nicht zugewiesene Aufträge vorhanden, lässt sich erneut das maximale Verhältnis von Belastung zu Kapazität ermitteln und somit der neue Systemengpass definieren. Hier gilt es wieder unter Verwendung des vorgestellten Algorithmus die optimale Reihenfolge abzuleiten bis jeder Engpass im System optimal ausgenutzt ist, d.h. jeder Auftrag zugewiesen ist.

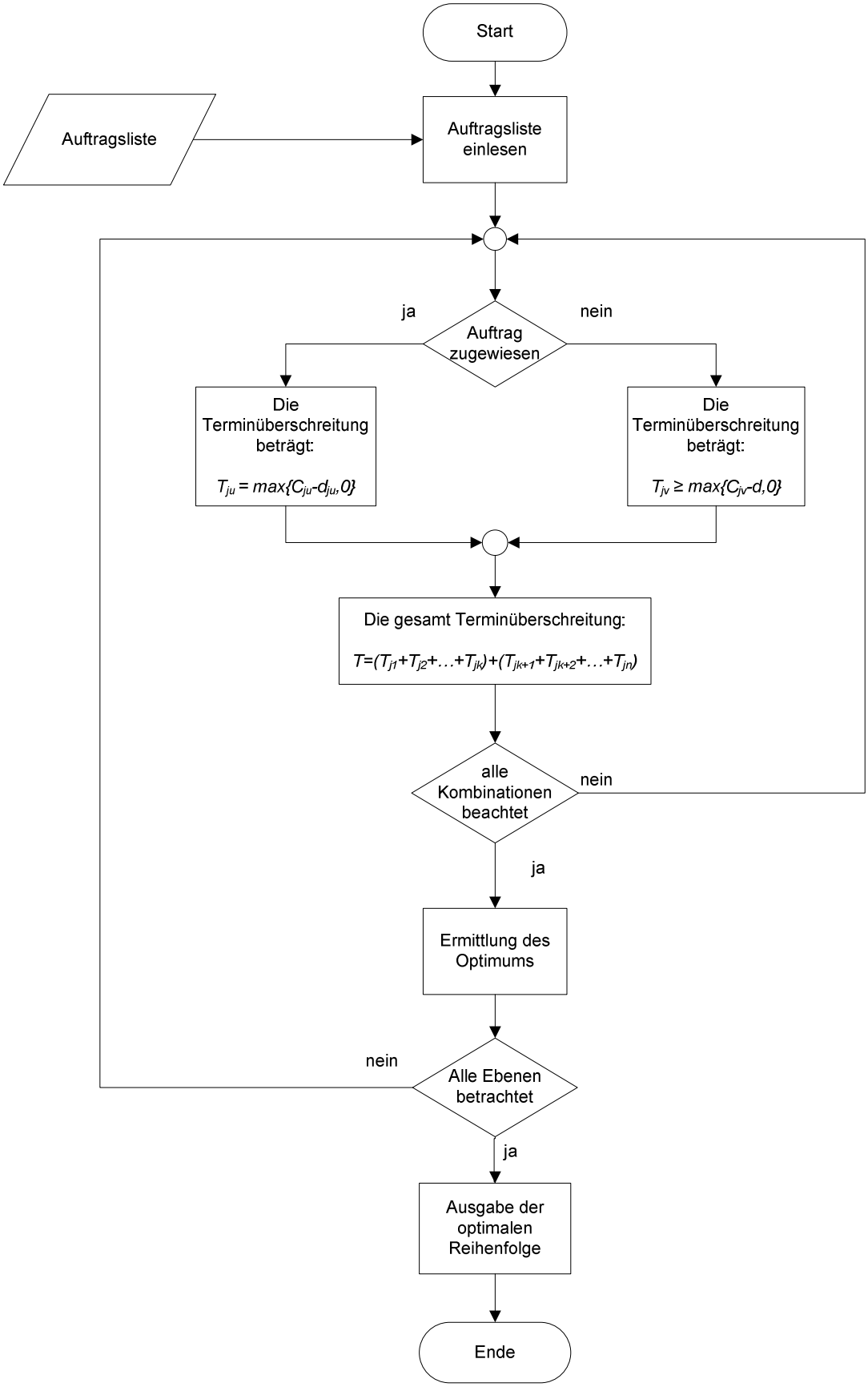

Abbildung 4.3: Vorgehensweise des Algorithmus zur Minimierung der gesamt Terminüberschreitung [Eigene Darstellung]

4.3 Anwendungsdemonstration des Algorithmus anhand eines Fallbeispiels

Zur Verdeutlichung soll nun die Anwendung des Algorithmus anhand eines konkreten Fallbeispiels dargestellt werden. Die Auftragsnummer j, die Fertigstellungstermine d_j sowie die jeweiligen Bearbeitungszeiten p_j sind dem Arbeitsplan zu entnehmen. Die Bearbeitungszeit p_j ist in Minuten angegeben. Der Fertigstellungstermin d_j ergibt sich aus dem Datum. Zur Berechnung ist dieser Wert ebenfalls in Minuten angegeben. Ein Tag hat eine Kapazität von 8 Stunden, also 480 Minuten. So kann ab dem Start der Periode bis zu ihrem Ende ein gültiger Fertigstellungstermin generiert werden. 960 Minuten entsprechen also zwei Tagen. Da ab Start der Periode gerechnet wird ist demzufolge der erste Fertigstellungstermin von 960 Minuten am Ende des zweiten Tages.

Das zu betrachtende Fertigungssystem ist der Abbildung 4.4 zu entnehmen. Dabei sind denen als Rechtecke dargestellten Arbeitsstationen AS_i jeweils eine Kapazität K_{AS_i} zugeordnet. Die Aufträge AF_i sind als farbliche Verbindungen zwischen den jeweiligen Arbeitsstationen dargestellt. Es gilt nun zunächst das maximale Verhältnis von Belastung zu Kapazität zu bestimmen. Hierzu wird wieder die Formel 2.5 angewendet.

$$BQ_i = \frac{AF_{ij}}{K_{AS_i}} \qquad [2.5]$$

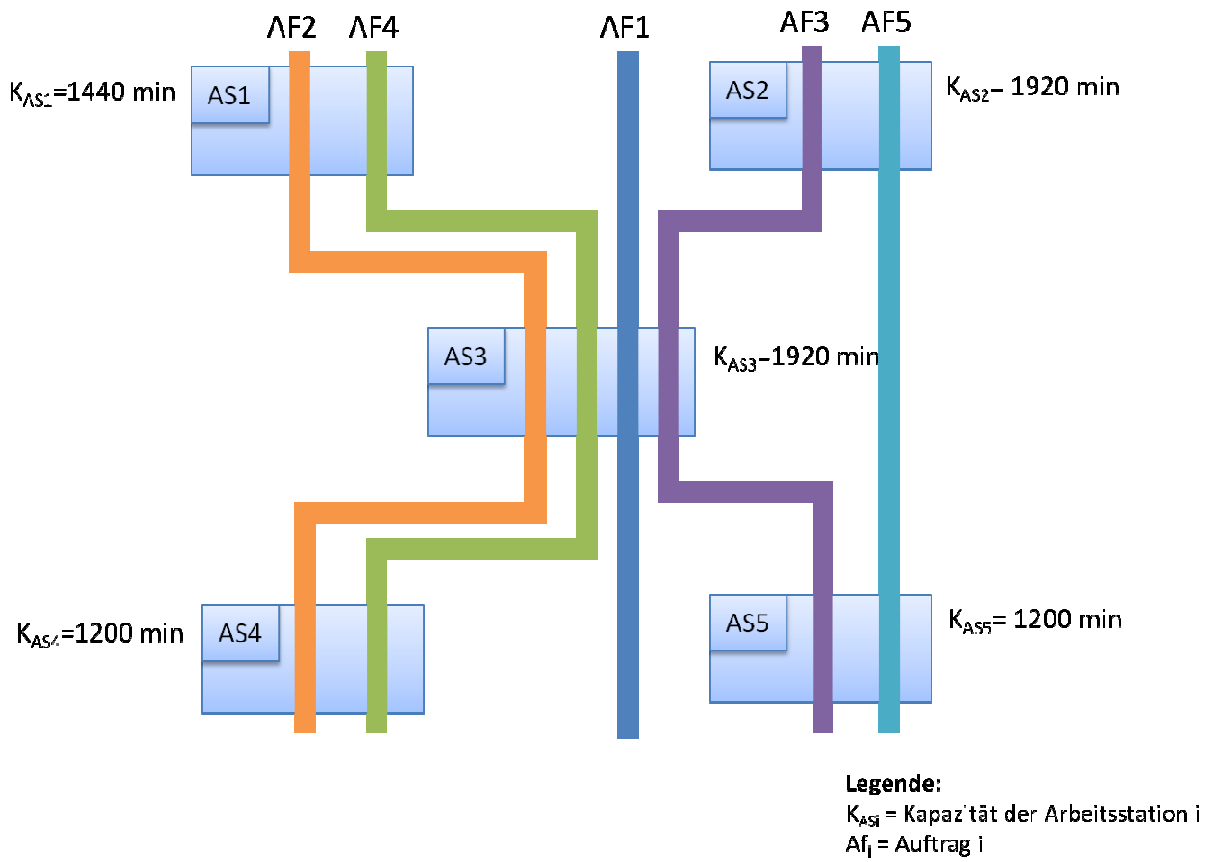

Abbildung 4.4: Übersicht des Fertigungsnetzwerks mit Arbeitsstationen inklusive der Kapazität und der Zuordnung der Aufträge [Eigene Darstellung]

Die Arbeitsinhalte belasten die Arbeitsstationen dabei wie folgt:

- *AF 1*

  AS3 = 960 min

- *AF 2*

  AS1 = 960 min

  AS3 = 595 min

  AS4 = 480 min

- *AF 3*

  AS2 = 960 min

  AS3 = 610 min

  AS5 = 610 min

- *AF 4*

  AS1 = 610 min

  AS3 = 1700 min

  AS4 = 480 min

- *AF5*

  AS2 = 800 min

  AS5 = 480 min

Somit ergeben sich für die Arbeitsstationen folgende Belastungsquotienten, die in Tabelle 4.1 aufgeführt sind. In dem Fallbeispiel beträgt die Kapazität der größten Engpassmaschine 1920 Minuten. Die sich ergebende Einlastung der vier betrachteten Aufträge ergibt 3865 Minuten. Der resultierende Belastungsquotient lautet 2,01 nach der Formel 2.5.

$$BQ_i = \frac{AF_{ij}}{K_{AS_i}} = \frac{3865\ min}{1920\ min} = 2{,}01 \qquad [2.5a]$$

Tabelle 4.1: Belastungsquotienten der einzelnen Arbeitsstationen

Arbeitsstation	$\boldsymbol{BQ_i = \frac{AF_{ij}}{K_{AS_i}}}$
AS 1	1,09
AS 2	0,92
AS 3	2,01
AS 4	0,80
AS 5	0,91

Damit ist der größte Systemengpass gefunden. Jetzt gilt es die auf diesem Engpass eingeplanten Aufträge zu identifizieren. Die Aufträge und ihre zugehörigen Bearbeitungszeiten sowie Fertigstellungstermine sind der Tabelle 4.2 zu entnehmen.

Tabelle 4.2: Eingangsdaten zur Ermittlung der optimalen Reihenfolge auf Arbeitsstation 3

Auftrag	1	2	3	4
p_j / min	960	595	610	1700
d_j / min	960	1440	1920	3360

Die Berechnung startet mit der Initialisierung auf der Ebene null des Branch-an-Bound-Verfahrens. Hier sind noch keine Aufträge zugewiesen. Im nächsten Schritt erfolgt die Bildung der ersten möglichen Reihenfolgen. Die erste Reihenfolge lautet daher (1,*,*,*), wobei die Sterne wieder für die nicht zugewiesenen Aufträge stehen. Nach der Formel 4.2 berechnet sich folgender Wert für die Tardiness.

$$T_1 = max\ \{C_1 - d_1, 0\} = max\ \{960\ - 960\ ,0\} = 0\ min \qquad [4.2a]$$

Die Tardiness beträgt null, da keine Verspätung zu diesem Zeitpunkt vorliegt. Es werden also nur positive Terminüberschreitungen berücksichtigt. Die gesamte Terminüberschreitung der noch nicht zugewiesenen Aufträge 2, 3 und 4 ist nicht weniger als die kleinste. Daher wird diese nun mit dem aus Formel 4.4 größtem Fälligkeitsdatum der noch verbliebenen Aufträge berechnet. Hier beträgt dieses d_4 = 3360 Minuten.

$$d = max(d_2,\ d_3\ und\ d_4) = ma\,x(1440, 1920\ und\ 3360) = 3360\ min \qquad [4.4a]$$

Die Reihenfolge zur Ermittlung der sich so ergebenden Tardiness erfolgt unter Berücksichtigung der SPT-Regel. In dem Beispiel bedeutet das, dass die Aufträge in der Reihenfolge (1,2,3,4) berechnet werden. Somit ergibt sich nach Formel 4.3 die restliche Tardiness der nicht zugewiesenen Aufträge zu:

$$T_2 + T_3 + T_4 \geq max\ \{C_2 - d, 0\} + max\ \{C_3 - d, 0\} + max\ \{C_4 - d, 0\} \qquad [4.3a]$$

$$= max\ \{1555 - 3360\} + max\ \{2165 - 3360\} + max\ \{3865 - 3360\}$$

$$= 0 + 0 + 0 + 505$$

$$= 505\ min.$$

Die gesamte Terminüberschreitung der Reihenfolge (1,*,*,*) ergibt sich somit zu 505 Minuten.

$$T = T_{j1}\ +\ T_2 + T_3 + T_4 = 0 + 0 + 0 + 505 = 505\ min. \qquad [4.5a]$$

In analoger Vorgehensweise lassen sich so die Tardinesswerte für die Reihenfolgen (2,*,*,*) = 505, (3,*,*,*) = 505 und (4,*,*,*) = 3305 ermitteln. Da die Reihenfolge (4,*,*,*) den höchsten und damit im Sinne der Zielfunktion schlechtesten Ausgangspunkt bildet, wird dieser Ast nicht weiter verfolgt. Auf der Ebene 1 kommt es also pro Knoten zu jeweils drei neuen Kombinationsmöglichkeiten. Die erste Ebene des Suchbaumes und die sich ergebenen Knoten sind in Abbildung 4.5 dargestellt.

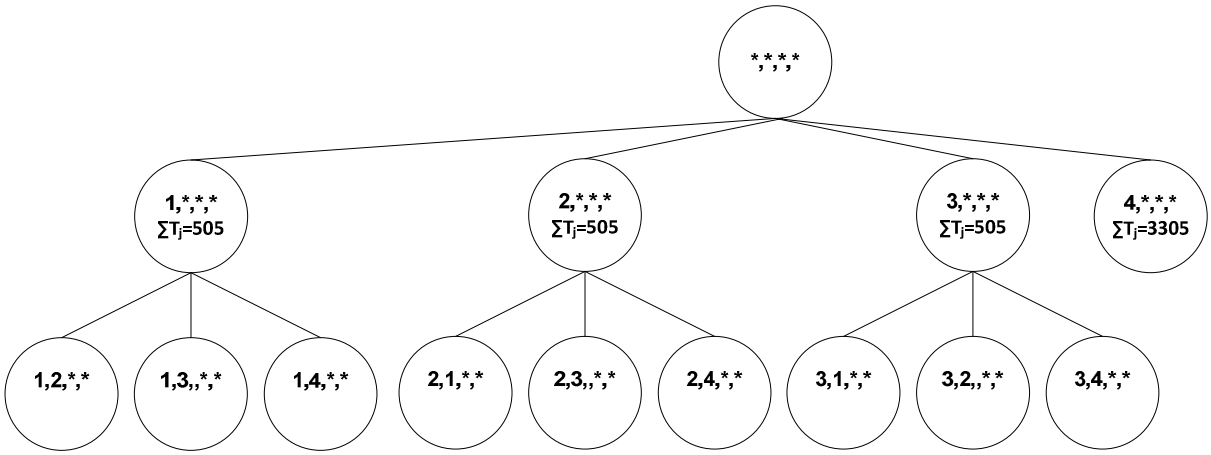

Abbildung 4.5: Darstellung der ersten Ebene des Suchbaumes und der sich ergebenden Knoten [Eigene Darstellung]

Im nächsten Schritt des Algorithmus werden für die ermittelten Optima der ersten Ebene wieder die Terminüberschreitungen ausgerechnet. Exemplarisch wird nur für einen Fall das Verfahren erneut durchgespielt. Zu beachten ist an dieser Stelle, dass jetzt zwei Aufträge zugewiesen sind. Die Tardiness für den Knoten (1,2,*,*) auf der nächsten Ebene berechnet sich nun auf die im nachfolgenden gezeigte Weise.

Zugewiesene Aufträge

$$T_1 + T_2 = max\ \{C_1 - d_1, 0\} + max\ \{C_2 - d_2\} \qquad [4.2b]$$

$$= max\ \{960 - 960{,}0\} + max\ \{1555 - 1440\}$$

$$= 0 + 115$$

$$= 115\ min$$

Ermittlung eines neuen d

$$d = max(\ d_3\ und\ d_4) = max\ (1920\ und\ 3360) = 3360\ min \qquad [4.4b]$$

Nicht zugewiesene Aufträge

$$T_3 + T_4 \geq max\ \{C_3 - d, 0\} + max\ \{C_4 - d, 0\} \qquad [4.3b]$$

$$= max\ \{2165 - 3360\} + max\ \{3865 - 3360\}$$

$$= 0 + 505$$

$$= 505\ min$$

Die gesamte Terminüberschreitung der Reihenfolge (1,2,*,*) ergibt sich somit zu 620 Minuten.

$$T = T_{j1} + T_{j2} + T_3 + T_4 = 0 + 115 + 0 + 505 = 620\,min \quad [4.5b]$$

Die in gleicher Vorgehensweise ermittelten Terminüberschreitungen aller Reihenfolgen ergeben erneut drei Optima (siehe Abbildung 4.6). An diesen Optima werden die Äste weiter verfolgt. Die anderen Knoten werden nicht weiter betrachtet, da hier keine besseren Lösungen mehr möglich sind. Es müssen also alle Kombinationsmöglichkeiten für alle noch aktiven Äste des Suchbaumes auf der jeweiligen Ebene untersucht werden. Der Übersichtlichkeit halber ist der gesamte Suchbaum in Abbildung 4.6 dargestellt. Die Berechnung erfolgt dabei auf immer die gleiche Weise, bis alle Aufträge zugewiesen sind. Die optimale Sequenz nach abschreiten jeden optimalen Astes ergibt sich somit zu 1,3,2,4 mit einer Terminüberschreitung von 1230 Minuten.

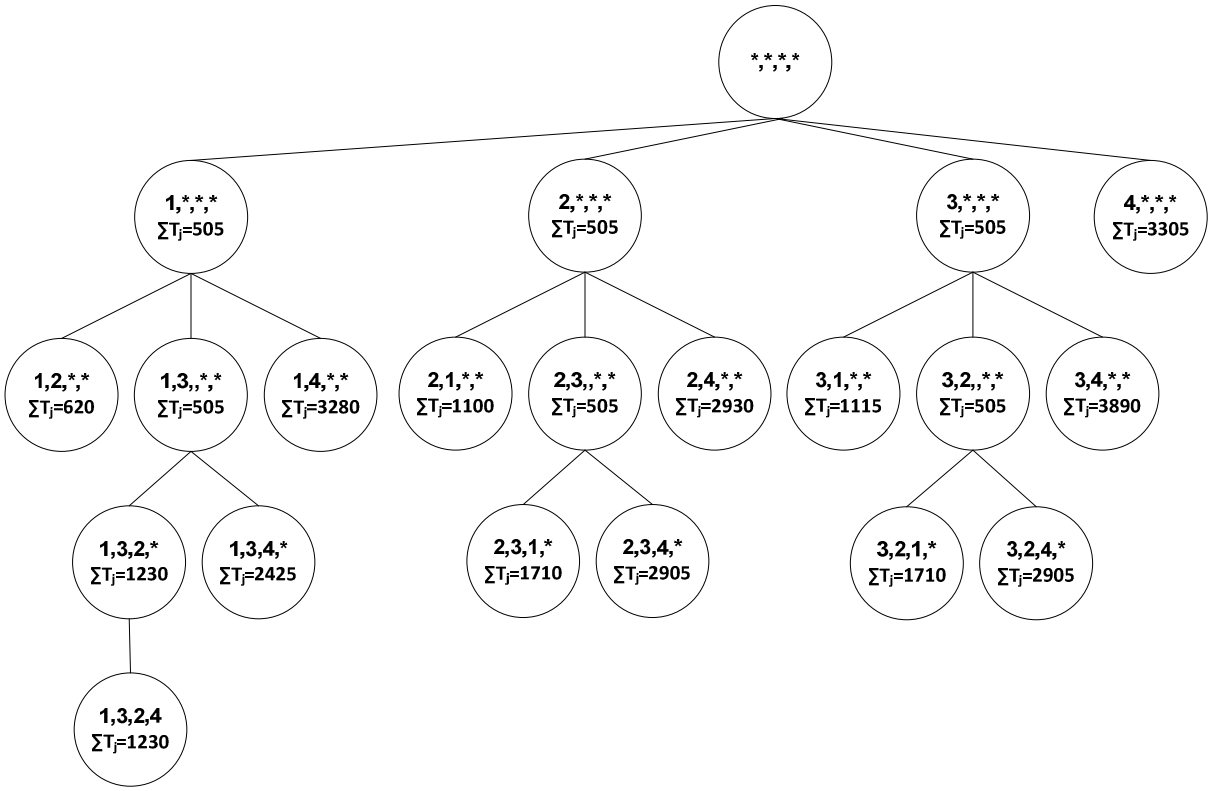

Abbildung 4.6: Branch-and-Bound für die Ermittlung der optimalen Reihenfolge [Eigene Darstellung]

Nach Ermittlung der optimalen Sequenz auf dem Engpass erfolgt die Prüfung der Durchführbarkeit der Aufträge. Die Summe aller Prozesszeiten muss mit der Kapazität auf dem Engpass verglichen werden. Ist diese kleiner, so können alle Aufträge durchgesetzt werden. Ist diese jedoch größer, so muss ermittelt werden, welche Aufträge noch durchgesetzt werden können. Da der Engpass nur über eine Kapazität von 1920 Minuten verfügt, können die Aufträge 1 und 3 vollkommen durchgesetzt werden. Die sich ergebende Einlastung von den Aufträgen eins und drei beträgt 1570 Minuten. Der Auftrag 2 hingegen kann nicht vollständig durchgesetzt werden. Dieser sollte jedoch auch noch in dieser Periode gestartet werden, da er in der folgenden Periode abgearbeitet wird. Auftrag 4 hingegen kann in dieser Periode nicht mehr durchgesetzt werden und sollte ggf. fremdvergeben werden. Die resultierende Reihenfolge ergibt sich damit zu der in Tabelle 4.2 dargestellten Auftragsliste inklusive dem spätesten Bereitstellungstermin r_j auf dem Engpass.

Tabelle 4.3: Resultierende Auftragsliste für den Engpass (Arbeitsstation 3)

Auftrag	1	3	2
p_j / min	960	610	595
d_j / min	960	1920	1440
r_j / min	0	960	1570

Im nächsten Schritt muss geprüft werden, welche Aufträge noch nicht zugewiesen wurden. Die Aufträge eins bis drei sind eingeplant, während der Auftrag vier nicht gestartet werden kann. In dem Fallbeispiel ist jetzt lediglich noch Auftrag fünf zu betrachten. Dieser läuft über die Arbeitsstationen zwei und fünf. Die korrigierte Kapazität von Arbeitsstation zwei beträgt 960 Minuten, da der Arbeitsinhalt von Auftrag drei subtrahiert werden muss. Die Einlastung von Auftrag fünf beträgt auf Arbeitsstation zwei 800 Minuten. Somit ergibt sich ein Belastungsquotient von 0.83. Auf Arbeitsstation fünf ergibt sich in analoger Vorgehensweise ein Belastungsquotient von 0,81. Daraus folgt, dass der Auftrag fünf ohne weiteres durchgesetzt werden kann. Eine Reihenfolgebildung ist nicht mehr nötig, da kein Engpass mehr vorliegt. Im Falle einer neuen Reihenfolgeplanung wäre Auftrag fünf so einzuplanen, dass der größte Systemengpass nicht beeinflusst würde. Die ermittelte Reihenfolge auf dem bereits geplanten Engpass darf also nicht verändert werden. Jetzt kann für das gesamte System ein Maschinenbelegungsplan abgeleitet werden. Dieser ist in Abbildung 4.7 dargestellt.

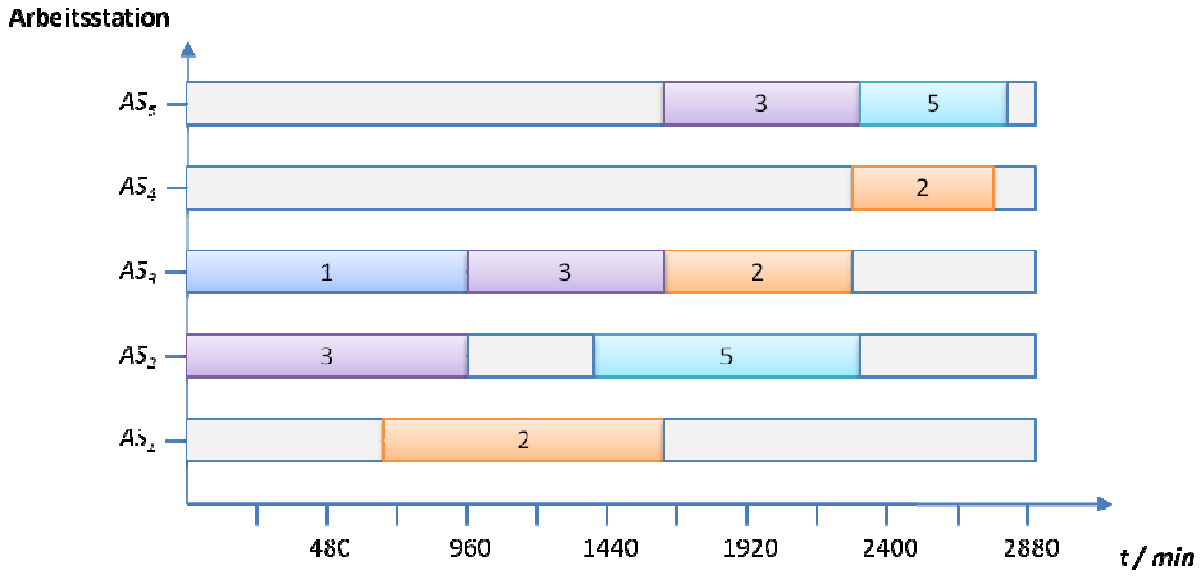

Abbildung 4.7: Maschinenbelegungsplan für das gesamte Fertigungssystem

4.4 Implementierungsvorschlag des neuen Algorithmus

Der neue Berechnungsalgorithmus zur Reihenfolgebestimmung auf dem Engpass soll in diesem Kapitel in den bestehenden Algorithmus implementiert werden. Dazu wird ein Flussdiagramm (siehe Abbildung 4.8) erstellt, welches die Implementierung in den bestehenden DePlaVis-Algorithmus darstellt. Im Falle einer möglichen Umsetzung des Algorithmus in die Praxis soll dieser Implementierungsvorschlag das Vorgehen erleichtern.

Wie auch bei dem bisherigen Algorithmus beginnt auch dieser Algorithmus mit der Bestimmung des größten Engpasses über einen Belastungsabgleich. Zuerst muss die Einlastung auf den Arbeitsstationen bestimmt werden. Dazu werden die Informationen aus der Auftragsliste eingelesen. Ist die Einlastung ermittelt, so wird in einem Abgleich mit der jeweiligen Kapazität der Belastungsquotient gebildet. Im Anschluss daran sind das maximale Verhältnis von Belastung zu Kapazität und die zugehörige Arbeitsstation bestimmt. Alle Aufträge, die über diese Arbeitsstation laufen, werden für die Reihenfolgebildung berücksichtigt und in einer separaten Auftragsliste abgelegt.

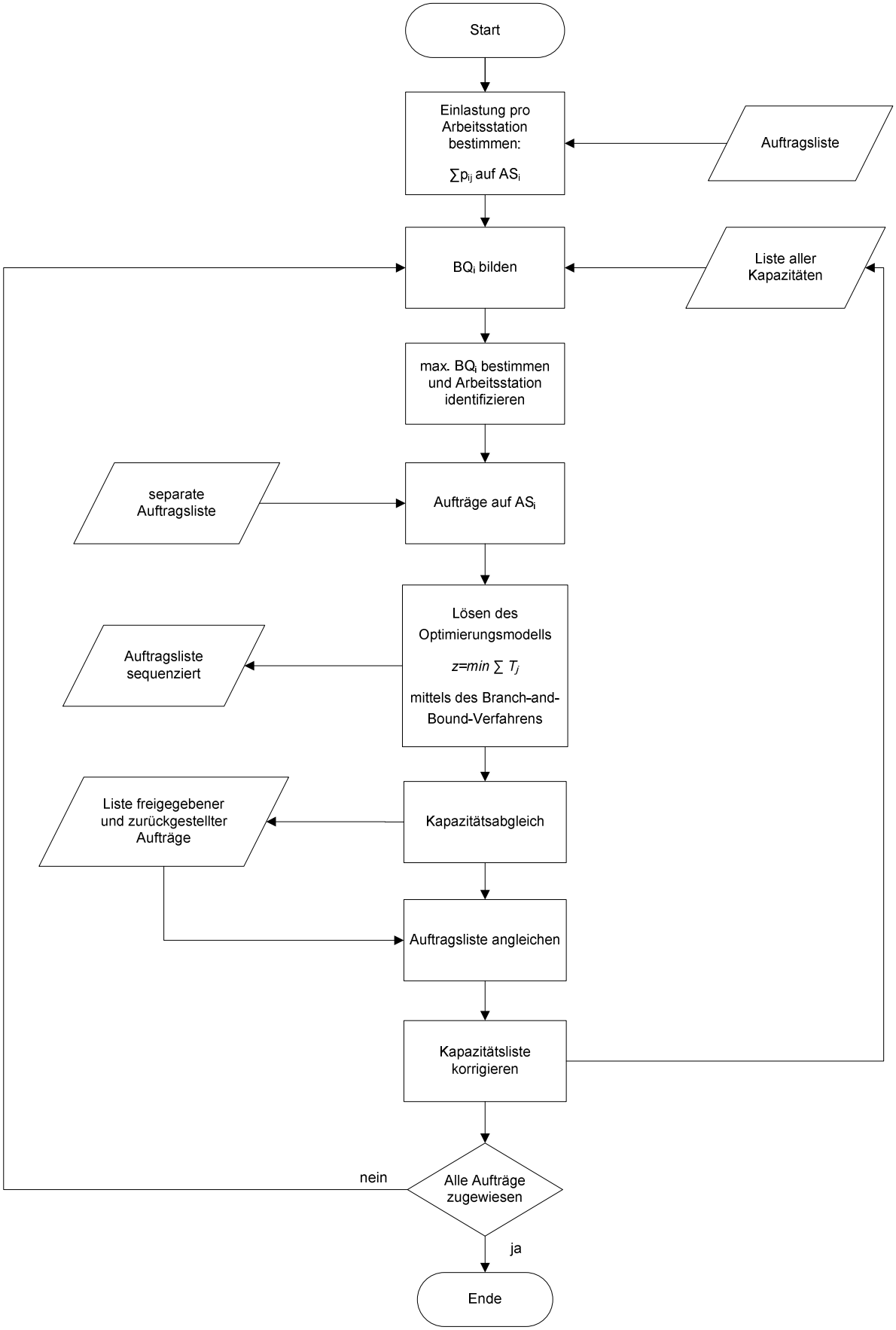

Abbildung 4.8: Implementierungsvorschlag des neuen Algorithmus [Eigene Darstellung]

Im nächsten Schritt ist das neue Optimierungsmodell $z = min \sum T_j$ aufzustellen und mittels des Branch-and-Bound-Vorgehens zu lösen. Es wird die optimale Reihenfolge ausgegeben. Die Reihenfolge ergibt die resultierende Einlastung auf dem Engpass durch summieren der Bearbeitungszeiten. Diese Einlastung wird der Kapazität der Engpassmaschine gegenüber gestellt. Jetzt sind nur die Aufträge freizugeben, die innerhalb einer Periode durchgesetzt werden können. Daraus lässt sich eine neue Auftragsliste generieren. Diese Auftragsliste beinhaltet die optimale Reihenfolge und die Bereitstellungstermine r_j auf dem betrachteten Engpass.

Nach dem Durchlauf für die Arbeitsstation mit dem größten Belastungsquotienten werden die Kapazität und die Einlastung des gesamten Systems angeglichen. Die Aufträge, die bereits freigegeben werden, erhalten eine reservierte Kapazität auf allen Maschinen. Diese besetzte Kapazität ist anzugleichen. Demzufolge muss also auch die Einlastung angeglichen werden. Die neuen Kapazitäten und die neue Einlastung werden wieder in einer Tabelle abgelegt.

Im letzten Schritt muss geprüft werden, ob alle Aufträge berücksichtigt wurden. Ist dieses nicht der Fall, so werden erneut alle Belastungsquotienten mit den angeglichenen Daten ermittelt. Es gilt wieder die Arbeitsstation optimal zu belegen, die das größte Verhältnis von Belastung zu Kapazität aufweist. Dieses Vorgehen ist so lange durchzuführen, bis alle Aufträge berücksichtigt wurden. Ist dies der Fall, so endet der Prozess und alle Engpass-maschinen sind optimal belegt. Jetzt kann der Durchsatz berechnet werden. Alle Werte werden in einer Tabelle abgelegt. Schließlich können daraus noch der direkte und indirekte Durchsatz abgeleitet werden. Anschließend wird die Durchsatzkennlinie gezeichnet.

4.5 Erstellung eines Demonstrators in Matlab

Die Erstellung eines Demonstrators in Matlab soll die Anwendbarkeit des neu entwickelten Algorithmus darstellen. Ein Programm-Code kann dem Anhang entnommen werden. Matlab dient im Gegensatz zu Computeralgebrasystemen nicht der symbolischen, sondern primär der numerischen (zahlenmäßigen) Lösung von Problemen. Die Software wird in der Industrie und an Hochschulen vor allem für numerische Simulation sowie Datenerfassung, Datenanalyse und -auswertung eingesetzt. Programmiert wird unter Matlab in einer proprietären[14] Programmiersprache, die auf der jeweiligen Maschine (Computer) interpretiert

[14] Proprietäre Programmiersprache bedeutet in einer eigenen von Matlab verwendeten Codierung

wird. Kleinere Programme können als so genannte Skripts oder Funktionen zu atomaren Einheiten verpackt werden. Zur Programmierung des Branch-and-Bound-Verfahrens wird Matlab hier eingesetzt, da es an der HAW-Hamburg in den Informatik-Vorlesungen gelehrt und frei zur Verfügung gestellt wird. Die endgültige Umsetzung des Algorithmus sollte jedoch in einer Datenbankanwendung umgesetzt werden. Matlab ist, wie gezeigt, ein Programm welches primär zur Lösung von numerischen Problemen eingesetzt werden sollte. Eine Datenbankanwendung zur Umsetzung des Algorithmus scheint hier deutlich besser geeignet.

Am Start des Programmes sind die Eingangsdaten in einem Array einzulesen. Das Array beinhaltet alle benötigten Informationen für die Bildung der optimalen Reihenfolge. Dazu gehören die Auftragsnummer j, die Bearbeitungszeit p_j und der Fertigstellungszeitpunkt d_j. Jetzt können die Eingangsdaten für die Ermittlung der optimalen Reihenfolge aus dem Array eingelesen werden. Das Array wird nach den Bereitstellungsterminen sortiert und nummeriert, damit der Algorithmus angewendet werden kann. Mittels Zählschleifen kann nun die erste Ebene und alle ihr zugehörigen Kombinationen des Suchbaumes abgebildet werden. Zur Berechnung der Tardiness ist eine Funktion geschrieben worden, die auf die in Kapitel 4.2 erläuterte Vorgehensweise rechnet. Für jede Kombinationsmöglichkeit der jeweiligen Ebene gibt die Funktion den Tardiness-Wert zurück in das Programm.

Die berechneten Terminüberschreitungen werden in einem Tensor abgelegt. Dieser Tensor beinhaltet für jede Ebene j und jeden Knoten alle Tardiness-Werte. Für jeden ermittelten Tardiness-Wert wird in einem neuen Tensor die Reihenfolge gespeichert. Ist auf einer Ebene des Suchbaumes jeder Tardiness-Wert ermittelt, so müssen die Optima gefunden werden. Dieses geschieht mit Hilfe einer Untersuchung auf Minima. Sind die Optima gefunden werden die Reihenfolgen aus dem Reihenfolge-Tensor übernommen. Nun kann die nächste Ebene des Suchbaumes untersucht werden. Alle nicht optimalen Tardiness-Werte aus dem Tensor werden nicht weiter untersucht. Dieses ist mit Hilfe eines Abbruchkriteriums berücksichtigt worden.

Auf der nächsten Ebene ist zu beachteten, dass bereits zu jedem Tardiness-Wert eine optimale Reihenfolge existiert. Die Funktion zur Ermittlung der Tardiness muss also die Ebene des Suchbaumes übermittelt bekommen und die bereits zugewiesenen Aufträge. Nach dem in Kapitel 4.2 vorgestelltem Algorithmus, muss zwischen zugewiesenen und nicht zugewiesenen Aufträgen unterschieden werden. Dieses geschieht über die Zählvariable j.

Wie oben erwähnt steht diese für die betrachtete Ebene des Suchbaumes. Die Zählvariable j wird also mit jeder Ebene des Suchbaumes hochgezählt, bis alle n Ebenen abgebildet werden. Nun kann die Funktion die Tardiness-Werte an das Programm zurückgeben. Alle Tardiness-Werte und die ihnen zugehörigen Reihenfolgen werden erneut in dem jeweiligen Tensor gespeichert. Es findet an dieser Stelle wieder eine Untersuchung auf Optima statt. Nach der Untersuchung können ein oder mehrere Optima ermittelt werden, die wiederum verfolgt werden müssen. Dieses geschieht auf die oben gezeigte Weise.

Sind alle optimalen Reihenfolgen ermittelt, wird die Machbarkeit überprüft. Hierzu wird die Einlastung mit der Kapazität verglichen und angepasst. Das Ergebnis wird in einem Array gespeichert. Als Ausgabe wird der Arbeitsplan für die betrachtete Maschine mit den freizugebenden Aufträgen in der richtigen Reihenfolge ausgegeben.

Aufgrund der Programmstruktur und der iterativen Vorgehensweise kommt es zu sehr langen Berechnungszyklen. Man kann sich leicht vorstellen, dass eine Auftragsliste mit 40 oder mehr Aufträgen eine hohe Anzahl an Kombinationsmöglichkeiten bietet. Je mehr Aufträge also vorliegen, desto größer wird auch der Suchbaum. Demzufolge steigt ebenfalls die Berechnungszeit [vgl. HOS01 S.493 ff.].

Um ein Gefühl für das Wachstum der Kombinationsmöglichkeiten, und damit auch der Berechnungszeit, zu bekommen soll folgendes Beispiel dienen. Man überlege sich wie viele mögliche Reihenfolgen bei drei Aufträgen existieren. Jeder der drei Aufträge könnte an erster Stelle stehen. Folgend sind zwei Aufträge übrig, die an der zweiten Stelle stehen könnten und einer der die letzte Stelle belegt. Daher beträgt die Anzahl der Berechnungs-möglichkeiten, wie in Kapitel 4.2 bereits gezeigt $3 * 2 * 1 = 6$. Das bedeutet $3!$, sprich „drei Fakultät“. Ist nun nach der optimalen Reihenfolge gesucht, muss das Programm jede dieser Lösungen kennen. Es sind bereits in diesem kleinen Beispiel sechs mögliche Reihenfolgen. Bei einem Ein-Maschinen-Problem wächst die Anzahl der Lösungen exponentiell: $3! = 6$, $4! = 24$, $5! = 120$, $6! = 720$, usw. Je höher also die Anzahl der Aufträge, desto extremer steigt die Anzahl der Kombinationsmöglichkeiten und damit die Berechnungszeit. So ergeben sich bei der Anzahl von 25 Aufträgen bereits

$$25! = 15{,}511{,}210{,}043{,}330{,}985{,}984{,}000{,}000$$

mögliche Lösungen.

Für den in Matlab programmierten Demonstrator bedeutet das, dass nur Probleme mit wenigen Aufträgen in annehmbarer Zeit gelöst werden können. In der Tabelle 4.3 sind die Berechnungszeiten des Programmes aufgezeigt.

Tabelle 4.4: Datenermittlung der Rechenzeit des Demonstrators

Anzahl der Aufträge	**Anzahl der möglichen Kombinationen**	**Rechenzeit / sec**	
		Mittelwert	Standartabweichung
5	120	0,0492	± 0,0003
6	720	0,0617	± 0,0060
7	5040	0,0772	± 0,0056
8	40320	0,1482	± 0,0089
9	362880	0,2741	± 0,0141
10	3628800	0,7788	± 0,0115
11	3,992E+07	2,6741	± 0,0685
12	4,790E+08	3,9031	± 0,0537
13	6,227E+09	13,0660	± 0,0977
14	8,718E+10	17,7467	± 0,0278
15	1,308E+12	233,5169	± 0,3620

Zur Ermittlung der Mittelwerte wurden je 10 Messungen durchgeführt und die zugehörige Standartabweichung berechnet. Es ist zu erkennen, dass die Berechnungszeit ebenfalls mit den Kombinationsmöglichkeiten exponentiell steigt. Das im Programm eingebundene Abbruchkriterium verkürzt die Berechnungszeit, da nur noch die optimalen Lösungen berechnet werden. Vergleicht man die erfassten Daten mit denen einer Studie [vgl. HOS01 S.495], so ist zu erkennen, dass die erfassten Daten sehr nah an denen aus der Literatur liegen (siehe Tabelle 4.5).

Tabelle 4.5: Rechenzeit für die Reihenfolgebildung auf einer Maschine [vgl. HOS01 S.495]

Anzahl der Aufträge	Anzahl der möglichen Kombinationen	Rechenzeit
5	120	0,012 microsec
6	720	0,72 microsec
7	5040	5,04 microsec
8	40320	40,32 microsec
9	362880	362,88 microsec
10	3628800	3,63 millisec
11	3,992E+07	39,92 millisec
12	4,790E+08	479,00 millisec
13	6,227E+09	6,23 sec
14	8,718E+10	87,18 sec
15	1,308E+12	21,79 Minuten
⋮	⋮	⋮
20	2,433E+18	77,147 Jahre

Eine weitere Möglichkeit die Rechenzeit zu verkürzen ist die dynamische Programmierung [vgl. ZIM08 S.233 ff.]. Dynamische Programmierung kann dann erfolgreich eingesetzt werden, wenn das Optimierungsproblem aus vielen gleichartigen Teilproblemen besteht und eine optimale Lösung des Problems sich aus optimalen Lösungen der Teilprobleme zusammensetzt [vgl. PIN08 S.50 ff.]. Das Verfahren der dynamischen Programmierung besteht darin, zuerst die optimalen Lösungen der kleinsten Teilprobleme direkt zu berechnen und diese dann geeignet zu einer Lösung eines nächstgrößeren Teilproblems zusammenzusetzen. Einmal berechnete Teilergebnisse werden in einer Tabelle gespeichert. Bei nachfolgenden Berechnungen gleichartiger Teilprobleme wird auf diese Zwischenlösungen zurückgegriffen, anstatt sie jedes Mal neu zu berechnen. Wird die dynamische Programmierung konsequent eingesetzt, vermeidet sie zeitaufwändige Rekursionen, weil bekannte Teilergebnisse wiederverwendet werden. Die detaillierte Vorgehensweise der dynamischen Programmierung kann unter anderem bei Zimmermann (2008) und Pinedo (2008) entnommen werden.

4.6 Bewertung der Ergebnisse

Der neu entwickelte Lösungsansatz zur Berücksichtigung der Arbeitsgangreihenfolge erfolgt mittels der Minimierung aller Terminüberschreitungen. Hierzu wird die Abgrenzung des Problems auf ein Ein-Maschinen-Problem vorgenommen. So kann jeweils auf dem größten Systemengpass die optimale Sequenz ermittelt werden. Diese Vorgehensweise ist besonders sinnvoll, weil der Engpass das durchsatzbestimmende Element der Produktion ist [vgl. GOL90 S.5 ff.]. Die Zielfunktion $z = min \sum T_j$ zielt auf die Minimierung der Terminüberschreitungen aller Aufträge ab. Dies führt zu einer Optimierung des Durchsatzes hinsichtlich der Liefertermine. Nur die Aufträge mit einem zeitlich nahen Liefertermin werden freigegeben. Es ist dabei egal, wie groß ihr Arbeitsinhalt ist. So ist zu erwarten, dass der gesamt Durchsatz des Systems nicht gesteigert wird, lediglich die Berücksichtigung der Arbeitsgangreihenfolge erfolgt auf diese Weise.

Die Ziele der Produktionslogistik können teilweise abgebildet werden. Die Einhaltung der Liefertermintreue ist das Hauptziel des Algorithmus. Dieses wird in der Zielfunktion formuliert. Sekundär wird die optimale Auslastung des Engpasses verfolgt. Diese gilt daher nicht für das gesamte System, da immer nur so viel eingeplant wird, wie auf dem größten Engpass durchgesetzt werden kann. Nicht-Engpässe werden also nicht zwingend optimal ausgelastet. Die Minimierung der Durchlaufzeiten wird hingegen berücksichtigt. Aufgrund des Liefertermintreuekriteriums dürfen Aufträge keine langen Wartezeiten haben. So kann eine kurze Durchlaufzeit gewährleistet werden. Die Minimierung der Bestände und damit der Kapitalbindung im Produktionsprozess ist nicht berücksichtigt in der Zielfunktion. Da jedoch die Vorgehensweise darauf abzielt die Engpässe optimal zu belegen, so dass sich keine Warteschlange im System bildet kann indirekt darauf Einfluss genommen werden. Es bleibt jedoch festzuhalten, dass die Zielfunktion nur direkt auf die Liefertermintreue einwirken kann und somit dieses Ziel der Produktionslogistik erfüllt.

Die Berücksichtigung der diskreten Arbeitsinhalte hingegen kann gut abgebildet werden. Der Lösungsalgorithmus bildet die optimale Reihenfolge unter Berücksichtigung aller Lieferterminüberschreitungen und leitet die optimale Sequenz daraus ab. Im Anschluss daran wird der Engpass optimal belegt. Nur die Aufträge, die auch durchgesetzt werden können, sind freizugeben. Dies führt dazu, dass der Engpass optimal im Sinne der Zielfunktion ausgelastet ist. Die daraus resultierenden nicht freizugebenden Aufträge sollten unter Umständen fremdvergeben werden. Dies ist ein Vorteil für die Produktionsplanung, da schon im Vorfeld Entscheidungen über die Fremdvergabe von Aufträgen sinnvoll abgeleitet

werden können. Der neue Algorithmus berücksichtigt die Arbeitsgangreihenfolge und lastet den Engpass dementsprechend optimal aus. Zur Berechnung der optimalen Reihenfolge bezieht sich der Algorithmus auf die diskreten Arbeitsinhalte der Aufträge. Dies sind beispielsweise die Bearbeitungszeit p_j oder der Fertigstellungstermin d_j. Basierend auf diesen Informationen wird die optimale Reihenfolge ermittelt.

Die Vorgehensweise des Branch-and-Bound-Verfahrens ist sehr rechenaufwendig. Wie in Kapitel 4.5 gezeigt sind die Berechnungszyklen der iterativen Reihenfolgebildung sehr hoch. Sie steigen exponentiell, je mehr Aufträge zu berücksichtigen sind. Für die Praxis bedeutet dies, dass im realistischen Fall von 20 oder mehr Aufträgen pro Arbeitsstation keine Berechnung mehr durchführbar ist. Somit kann der programmierte Algorithmus nur begrenzt eingesetzt werden. Eine dynamische Programmierung hingegen gewährleistet den Einsatz des neuen Lösungsalgorithmus. So kann eine Vielzahl von Aufträgen auf einer Arbeitsstation in die optimale Reihenfolge gebracht werden. Aus diesen Ergebnissen kann somit ein Maschinenbelegungsplan erstellt werden. Die Programmierung des Branch-and-Bound-Verfahren sollte also dynamisch erfolgen.

Die Entwicklung eines Algorithmus zur Berücksichtigung der Arbeitsgangreihenfolge und dessen Programmierung in Matlab wurden im Rahmen dieser Bachelorthesis durchgeführt. Eine Validierung der Ergebnisse unter Berücksichtigung der Arbeitsgangreihenfolge konnte nicht durchgeführt werden. Die aufgezeigten Berechnungszeiten lassen eine Simulation nicht zu. Es ist nicht möglich ein Schedulingsproblem in annehmbarer Zeit zu lösen. Hier sollte zunächst eine dynamische Programmierung erfolgen. Die sehr anspruchsvolle dynamische Programmierung des Algorithmus muss also vor Beginn einer Simulationsstudie erfolgen.

In einer Simulationsstudie ist zu überprüfen, ob der neue Algorithmus tatsächlich den gesamt Durchsatz des Produktionssystems steigert. So gilt es zu prüfen, ob eine feste Reihenfolge auf dem Engpass, seine optimale Auslastung nach sich zieht. Für die Umsetzbarkeit der ermittelten Reihenfolge muss gewährleistet sein, dass die geplanten Aufträge den Engpass auch erreichen. Es ist zu erwarten, dass in einem realen Modell zu viele Engpässe vorhanden sind, die das System verstopfen. Dieses muss in einer Simulationsstudie geprüft werden. Zuerst sollten die Ergebnisse der Reihenfolgebildung an einem kleinen Modell überprüft werden. Die Ergebnisse müssen daraufhin mit den Ergebnissen des DePlaVis-Algorithmus verglichen werden. Aufbauend auf diesem Modell gilt es dann, ein ganzes Produktionssystem mit reellen Daten zu simulieren. Hierzu ist, wie oben gezeigt eine

dynamische Programmierung des Branch-and-Bound-Verfahrens umzusetzen. Erst nach der Auswertung der Simulationsergebnisse kann die Umsetzbarkeit des neuen Lösungsvorschlags validiert werden. Zu überprüfen gilt es, dass die optimale Reihenfolgebildung auf dem Engpass diesen optimal auslastet und die Leistung des Systems verbessert.

5 Zusammenfassung und Ausblick

Maschinenbelegungsprobleme sind aus vielerlei Gründen besonders anspruchsvoll. Zum einen stehen die Ziele der Produktionslogistik im gegenseitigen Konflikt. Auf der anderen Seite kann die Mathematik zur Lösung eines Schedulingproblems sehr komplex werden. Der hier diskutierte Lösungsansatz versucht das Optimierungsproblem zu vereinfachen. Zunächst erfolgt eine Abgrenzung des gesamten Systems auf ein Ein-Maschinen-Problem. Dieses ist der Systemengpass. Ist dieser optimal belegt, so kann der maximale Durchsatz des Systems erreicht werden. Danach werden die Ergebnisse auf das gesamte System übertragen.

Bisher wird bei der Berechnung der Durchsatzkennlinie die Arbeitsgangreihenfolge nicht berücksichtigt. Auf Arbeitsstationen, die zwar überlastet, aber nicht der größte Engpass des Systems sind, wird lediglich Kapazität für diesen Arbeitsstrom reserviert. Dieses verletzt die Voraussetzung der gleich proirisierten Aufträge. Zudem wird bisher nicht der diskrete Arbeitsinhalt eines Auftrages berücksichtigt. Der Durchsatz eines Auftrages hängt lediglich vom Durchsatz am Engpass ab.

Zur Lösung des Ein-Maschinen-Problems wird ein Algorithmus entwickelt, der die optimale Reihenfolgebildung auf dem Engpass vornimmt. Für jede Arbeitsstation des Produktionssystems wird der Belastungsquotient gebildet. Dieser ergibt sich aus dem Verhältnis von Einlastung zu Kapazität auf der jeweiligen Arbeitsstation. Sind alle Belastungsquotienten berechnet, so werden das maximale Verhältnis von Belastung zu Kapazität und damit der Engpass des Systems bestimmt. Mit Hilfe des neu entwickelten Algorithmus kann nun die optimale Reihenfolge auf dem Engpass berechnet werden. Das Optimierungskriterium ist es die Terminüberschreitungen aller berücksichtigten Aufträge minimal zu halten.

Die Zielsetzung des Optimierungssystems ist die Erreichung einer hohen Liefertermintreue bei maximalem Durchsatz auf dem Engpass. Ist der Abgleich von Belastung zu Kapazität erfolgt werden die durchsetzbaren Aufträge in das System eingeplant und der Engpass damit optimal ausgelastet. Sind alle Aufträge berücksichtigt, so wird der Maschinenbelegungsplan erstellt. Der entwickelte Lösungsansatz zielt damit auf ein kombiniertes Optimierungsmodell ab. Die Zielfunktion minimiert die Anzahl der Terminüberschreitungen. Danach wird das Ziel verfolgt, den maximalen Durchsatz zu erreichen. Dies ist durch die Einplanung der optimalen Reihenfolge auf dem durchsatzbestimmenden Element gewährleistet. Das Produktions-

system kann nun den maximalen Durchsatz erzielen. Das Optimierungsmodell berücksichtigt jedoch keine Kundenwunsche oder die Deckungsbeitragssteigerung. Diese Ziele der Produktionslogistik werden in dem Optimierungsmodell nicht abgebildet.

Der entwickelte Algorithmus zielt auf die Optimierung des gesamten Systems ab. Es wird jedoch nur der Engpass des Systems betrachtet. Dieser wird optimal im Sinne der Zielfunktion belegt. Auf den Nicht-Engpassmaschinen wird lediglich die Kapazität der freizugebenden Aufträge reserviert. Eine Planung findet hier jedoch nicht explizit statt. Die Berücksichtigung der gesamt Terminüberschreitung aller Aufträge hingegen ist zulässig. Es kann gezeigt werden, dass die Berücksichtigung der einzelner Aufträge zu langen Wartezeiten für diese führen kann.

Aus der vorliegenden Arbeit ergeben sich zudem weitere Aufgaben zur Verbesserung des DePlaVis-Algorithmus. So kann der neue Ansatz sehr gut zur Ermittlung der freizugebenden Aufträge genutzt werden. Als Ergebnis können die Aufträge ermittelt werden, die fremdvergeben werden sollten. Diese bilden somit keine Warteschlange im Produktionssystem mehr, was den gesamten Durchsatz des Systems erhöhen würde. Ein erster Implementierungsvorschlag wurde in der Arbeit skizziert und in Teilen in Matlab umgesetzt.

Eine Validierung des erarbeiteten Algorithmus muss erarbeitet werden. Dazu muss eine Simulationsstudie durchgeführt werden. Ein erstes Szenario sollte nur wenige Arbeitsstationen betrachten. Um die Umsetzbarkeit des neuen Algorithmus zu prüfen, muss ein Durchlauf mit dem bisherigen Vorgehen des DePlaVis-Algorithmus durchgeführt werden. Ein zweiter Durchlauf muss den neuen Algorithmus anwenden. Nun können die Durchsatzkennlinien verglichen werden. Ein zweites Szenario sollte die, dem DePlaVis-Projekt zur Verfügung stehenden, Praxisdaten verwenden. Hier gilt es zu prüfen, ob der entwickelte Algorithmus in der Praxis anwendbar ist. Es gilt zu untersuchen, ob die Trägheit des gesamten Systems die positiven Auswirkungen des neuen Algorithmus nicht zum Tragen kommen lässt.

6 Literaturverzeichnis

ALT08 Altfeld, Nils (2008): „Erweiterung des DePlaVis Ansatzes durch Integration von Optimierungsmethoden“, Diplomarbeit HAW Hamburg, Hamburg

BLA06 Blackstone, John H.; Gardiner, Stanley C.; Watson, Kevin J.(2006): „The evolution of a management philosophy: The theory of constraints“, in Journal of Operations Management, S.387-S.402, Elsevier

BOY04 Boyd, Lynn; Gupta, Mahes: „Constraints Management (2004): What is the theory?“, in International Journal of Operation & Production Management, S.350-S.371, Emarald, Lousiville

DOD02 Domschke, Wolfgang; Drexl, Andreas (2002): „Einführung in das Operations Research“, 5.Auflage, Springer-Verlag, Berlin Heidelberg New York

GLR05 Glynn, W. Peter; Pinedo, L. Michael; Robinson, M. Stephen (2005): „Planning and Scheduling in Manufacturing and Services”, Springer Verlag, New York

GOE02 Goemans, Michel X.; Queranne, Maurice; Schulz, Andreas S.; Skutella, Martin; Wang, Yaoguang (2002): „Single machine scheduling with realease dates“, in Society for industrial and Mathematics (SIAM) Vol. No.2, S.165-S.192

GOL90 Goldratt, Eliyahu (1990): „Theorie of Constraints, What is this thing called theory of constraints and how should it be implemented?“, 1.Auflage, Great Barrington

GOL08 Goldratt, Eliyahu (2008): „Das Ziel: Ein Roman über Prozessoptimierung“, 4.Auflage, Campus Verlag, Frankfurt/New York

GUT76 Gutenberg, Erich (1976): „Grundlagen der Betriebswirtschaftslehre, Die Produktion“, 22.Auflage Berlin

HAB99 Haberlandt, Karlheinz (1999): „Engpassorientierte Werkstattfertigung” in PPS-Management Nr.4, S.17-S.22

HOO04 Hoogeveen, Han (2004): „Multicriteria scheduling“, in European Journal of operational research, S.592-S.623, Elsevier, Utrecht

HOS01 Hopp, J.Wallace; Spearman, L. Mark (2004): „Factory Physics: Foundations of manufacturing management“, 2.Auflage, Irwin McGraw-Hill, Singapore

KRE95 Kreutzfeldt, Jochen (1995): „Planen mit Bearbeitungsalternativen in der Teilefertigung“, VDI-Verlag, Düsseldorf

KRE07 Kreutzfeldt, Jochen (2007): „BMBF-Antrag im Programm „IngenieurNachwuchs 2007“, DePlaVis Durchsatzsteigerung im Anlagenbau und -betrieb durch engpassorientierte Planung und Visualisierung“, Hamburg

KRS09 Kreutzfeldt, Jochen; Schultheiss, Johannes (2009): „Performance improvement in production systems through practice oriented bottleneck management“, 4th European Conference on Technology Management, Glasgow

LEC99 Lechleiter, Iris (1999): „Maschinenbelegungspalung in der Variantenfertigung: Job-Shop-Scheduling mit Fälligkeitsterminen und Flow-Shop-Scheduling mit begrenzten Zwischenlägern“, Deutscher Universitäts-Verlag, Wiesbaden

LÖD08 Lödding, Hermann (2008): „Verfahren der Fertigungssteuerung“, 2.Auflage, Springer-Verlag, Berlin Heidelberg

NEM02 Morlock, Martin; Neumann, Klaus (2002): „Operations Research“, 2.Auflage, Hanser Verlag, München Wien

NYH03 Nyhuis, Peter (2003): „Logistische Kennlinien: Grundlagen, Werkzeuge und Anwendungen“, 2.Auflage, Berlin

NYH04 Nyhuis, Peter (2004): „Logistische Terminkennlinie - Ein Werkzeug zur Beherrschung der Terminsituation von Produktionssystemen“ in: PPS-Management 9, S.38-S.40.

NYH05 Nyhuis, Peter (2005): „Applying Simulation and Analytical Models for Logistic Performance Prediction“ in CIRP Annals, Oxford

NYH08 Nyhuis, Peter (2008): „Beiträge zu einer Theorie der Logistik“, Springer Verlag, Berlin Heidelberg

PAP01 Papula, Lothar (2001): „Mathematik für Ingenieure und Naturwissenschaftler“, Band 2, Vieweg Verlag, Wiesbaden

PIN08 Pinedo, L. Michael (2008): „Scheduling: Theory, Algorithm, and Systems“, 3.Auflage, Springer Verlag, New York

SCH08 Schultheiss, Johannes (2008): „Darstellung und Bewertung des Potentials und der Grenzen des DePlaVis-Ansatzes anhand von Fallbeispielen“, Masterprojekt HAW Hamburg, Hamburg

SCH09 Schultheiss, Johannes (2009): „Ausarbeitung und Verifikation eines Vorschlags zur Berücksichtigung der Bearbeitungsreihenfolge im DePlaVis-Algorithmus“, Masterthesis HAW Hamburg, Hamburg

SUM09 Mellouli, Taïeb; Suhl, Leena (2009): „Optimierungssysteme: Modelle, Verfahren, Software, Anwendungen“, 2.Auflage, Springer-Verlag, Berlin Heidelberg

TAI92 Taillard, E. (1992): „Benchmarks for basic scheduling problems“, in European Journal of Operational Research 64, S.278-285, Elsevier, North-Holland

UNG10 Dempe, Staphan; Unger, Thomas (2010): „Lineare Optimierung: Modell, Lösung, Anwendung“, Vieweg und Teubner Verlag, Wiesbaden

WER08 Werners, Brigitte (2008): „Grundlagen des Operations Research“, 2.Auflage, Springer-Verlag, Berlin Heidelberg

WIE92 Wiendahl, Hans-Peter (1992): „Anwendung der belastungsorientierten Auftragsfreigabe“, Hanser Verlag, München

WIE08 Wiendahl, Hans-Peter (2008): „Betriebsorganisation für Ingenieure“, 6.Auflage, Hanser Verlag, München Wien

ZIM08 Zimmermann, Hans-Jürgen (2008) : „Operations Research“, 2.Auflage, Vieweg Verlag, Wiesbaden

Anhang A

```
%---Matlab Demonstrator---Scheduling-----------------------------------------
%Zweck        :  Mit Hilfe des B&B-Verfahrens werden alle optmalen Äste
%                abgetastet. Die Funktion Tardiness liefert die
%                Tardinesswerte des jeweiligen Astes.
%                Die optimale Reihenfolge wird von dem Programm ausgegeben.
%Autor        :  Simon Schütt
%Datum        :  20.08.2010
%-----------------------------------------------------------------------------
clc                                          %clear command window
clear all                                    %leert den workspace

%Ausgangsdaten werden aus Excelsheet importiert
Auftragsliste=xlsread('Reihenfolge_V1.xlsx', 2);


%Initialisierung der Variablen
A=Auftragsliste;
%Die Matrix A enthält in der ersten Spalte die Auftragsnummer, in zweiter
%Spalte die processtime und in dritter Spalte das due-date.

n=length(A);                                 %n steht für die Anzahl der Jobs

%Hilfsvariable v zum Überspringen der ersten Minimabestimmung, da im
%ersten Durchlauf noch keine existieren können.
v=0;


%Diese Zählschleife bildet den gesamten Suchbaum des Verfahrens ab. Sie
%zählt von der Ebene 2 bis zur letzten Ebene n+1. Auf der Ebene zwei werden
%die ersten Aufträge zugewiesen, deshalb der Start bei j=2.
for j=2:n+1

%In der folgenden if-Bedingung werden die Terminüberschreitungen ermittelt.
%Diese Bedingung ist erst nach Bildung der ersten Reihenfolge erfüllt, d.h.
%für v>0. Zu Beginn ist v gleich Null.
    if v>0
        minT=min(TTensor(1,[1:v-1],j-2));
        minimum=find(TTensor(1,:,j-2)==minT);  %Index des Minimums
        c=length(minimum);                     %c ist die Anzahl der Minima
        v=1;

%In der folgenden Zählschleife werden für die ermittelten Minima die
%Auftragslisten erstellt. Die Schleife läuft für alle existierenden
%Minima (c) durch. Das Resultat ist eine neue Auftragsliste, die die
%ermittelte Reihenfolge darstellt.
        for m=1:c
           A=ReihenfolgeTensor(:,minimum(1,m,1),j-2);
                 for l=1:length(A)
                   for i=1:length(A)
                       if A(l)==Auftragsliste(i)
                          A(l,2)=Auftragsliste(i,2);
                          A(l,3)=Auftragsliste(i,3);
                       end
                   end
                 end
```

```
%Jetzt können die neuen ReihenfolgeTensoren gebildet werden. Dies erfolgt
%mittels eines Arrays. So können die Tardinesswerte für die neue
%Reihenfolge ermittelt werden (Abruf der T-Funktion). Diese werden in dem
%TTensor und ReihenfolgeTensor abgelegt.
            for i=j-1:n
                B=A([j-1],:);                           %Hilfszeile B
                A([j-1,n],:) = A([i,n],:);              %A wird neu sortiert
                A([i],:) = B;                           %B wird in A eingesetzt
                ReihenfolgeTensor(:,v,j-1)=A(:,1);      %Reihenfolge aus A
                T=Tardiness(A,j,n);                     %Abruf der Funktion T
                TTensor(1,v,j-1)=T;                     %Tardiness in TTensor
                v=v+1;
            end
        end
    end

%Schleife läuft nur zur Initialisierung. Diese Schleife läuft nur für die
%erste Ebene ab (v=0), in der noch keine Reihenfolge gebildet und
%demzufolge auch keine optimale Reihenfolge vorliegen kann.
    if v<=0
        v=1;
        for i=j-1:n
            B=A([j-1],:);                               %Hilfszeile B
            A([j-1,n],:) = A([i,n],:);                  %A wird neu sortiert
            A([i],:) = B;                               %B wird in A eingesetzt
            ReihenfolgeTensor(:,v,j-1)=A(:,1);          %Reihenfolge aus A
            T=Tardiness(A,j,n);                         %Abruf der Funktion T
            TTensor(1,v,j-1)=T;                         %Tardiness in TTensor
            v =v+1;
        end
    end
end

%Schleife erzeugt die Ergebnissausgabe und ermittelt die durchsetzbaren
%Aufträge. Dies geschieht mit Hilfe des Kapazitätsabgleiches (Kapa)
fprintf('Die min Tardiness beträgt %3d bei %4d optimalen
Reihenfolgen.\n\n\n', minT, c)
for g=1:c
    Kapa=1960;                                  %Kapazität der Engpassmaschine
    Summe=0;
    K=ReihenfolgeTensor(:,minimum(1,g,1),j-2);
    for l=1:length(K)
        if Summe<=Kapa
            for i=1:length(K)
                if K(l)==Auftragsliste(i)
                    K(l,2)=Auftragsliste(i,2);
                    K(l,3)=Auftragsliste(i,3);
                end
            end
            Summe=Summe+K(l,2);
        end
    end
    fprintf('-----------------------------------------------------------------
---------------\n')
    fprintf('Die Summe der Prozesszeiten der durchführbaren Aufträge der
%3d. Reihenfolge\nbei einer Kapazität von %4d lautet %4d.\n', g, Kapa,
Summe)
    fprintf('\nDie %2d.optimale Arbeitsplan auf dem Engpass lautet
folglich:\n\n', g)
    disp(K)
end
```

Anhang B

```
function T=Tardiness(A,j,n)
%---Tardiness------Tardiness-----------------------------------------------
%Zweck        :  Funktion zur Berechnung der Tardiness T einer vorgegebenen
%               Reihenfolge.
%               Ausgangsdaten sind die Anzahl der Jobs (n), die Ebene
%               innerhalb des B&B-Verfahrens (j) und eine Daten-Matrix mit
%               der Anzahl der Jobs, der Prozesszeiten und der due-dates.
%Autor        :  Simon Schütt
%Datum        :  26.07.2010
%-------------------------------------------------------------------------

%Initialisierung der Variablen für Ts-Zählschleife
T=0;                    %Tardiness
Ts = 0;                 %Tardiness scheduled
Tplus=0;                %Tardinesshilfsvariable
C = 0;                  %Completion Time
z = 1;                  %Zählvariable

%Zählschleife zur Bestimmung von Ts
for i=1:j-1             %Bezug zu allen Aufträgen, die bereits geplant sind
    C=C+A(z,2);
    Tplus=C-A(z,3);

    if Tplus<0
       Tplus=0;
    end

    Ts=Ts+Tplus;
    z=z+1;
end

%Initialisierung der Variablen für Tus-Zählschleife
Tus=0;                  %Tardiness unscheduled
Tplus=0;                %Tardiness hilfsvariable wird auf null gesetzt
z = 1;                  %Zählvariable
Cus = C;                %Completiontime unscheduled
B=A(j:n,1:3);           %B enthält die nicht geplanten Aufträge aus A
B=sortrows(B,2);       %B wird nach den Prozesszeiten sortiert (2.Spalte)
dh=A(j:n,3);            %Hiflsvariable zur Ermittlung des größten due-dates
                        %(Vektor der 3.Spalte)
dmax=max(dh);           %Ermittlung des größten due-dates aus dh

%Zählschleife zur Bestimmung von Tus
for i=j:n               %Alle Aufträge, die noch nicht geplant sind
    Cus=Cus+B(z,2);
    Tplus=Cus-dmax;

   if Tplus<0
        Tplus=0;
   end
   Tus=Tus+Tplus;
   z=z+1;
end

%Bestimmung der Tardiness aus Ts und Tus
T=Ts+Tus;
```