- Bachelorarbeit -

Ganzzahlige Optimierung für ein Zeitablaufsteuerungsproblem aus der Farbmittelindustrie

im Studiengang Bachelor of Business Administration

vorgelegt von:	Richard Spiegelberg
	Brahmsstraße 4
	50181 Bedburg
Geburtsdatum:	22. Juli 1986
Matrikelnummer:	27 46 25
Erstgutachter:	Prof. Dr. Marco Lübbecke
Abgabe:	20. März 2014

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Betriebliche Abläufe haben großen Einfluss auf die Wirtschaftlichkeit eines Unternehmens. Dabei stehen nicht selten zeitliche Einflussfaktoren im Vordergrund. Durch modelltheoretische Ansätze aus der Betriebswirtschaftslehre im Bereich 'Operations Research' werden solche zeitlichen Abläufe, die sich durch ein mathematisches Modell beschreiben lassen, analysiert und optimiert. Im Rahmen dieser Arbeit soll ein Modell für ein Zeitablaufsteuerungsproblem aus der Farbmittelindustrie entwickelt werden. Dabei stehen die Übertragung der in der Realität existierenden Produktionsanlage durch ein möglichst realistisches Modell im Vordergrund. Im Anschluss soll mit Hilfe einer benutzerfreundlichen Bedienoberfläche eine einfache Möglichkeit geschaffen werden, Änderungen im Produktionsplan vornehmen zu können, um gezielt Abläufe anpassen zu können. Dabei ist die Parametrierung ebenso wichtig, wie die Analyse der Simulationsergebnisse. Durch das Modell soll der benötigte Zeitaufwand und die damit verbundenen Kosten beim Betriebsablauf minimiert werden.

Das folgende Kapitel gibt eine kurze Einführung in die Grundlagen von Optimierungsproblemen. Dabei wird der Begriff "Scheduling", sowie Möglichkeiten zur Modellierung von Schedulingproblemen vorgestellt. Im Anschluß daran wird ein aus der Literatur entnommenes Schedulingmodell vorgestellt, welches als Vergleichsmodell für das in Kapitel 3 entwickelte Modell dient. Weiter wird eine, ebenfalls im Rahmen dieser Arbeit programmierte, Software vorgestellt, die eine komfortable Konfiguration der Modellparameter ermöglicht. Abschließend werden die Analysemöglichkeiten der Software anhand eines Beispiels erläutert.

2 Grundlagen von Optimierungsproblemen

Das zugrundeliegende Problem stammt aus der Farbmittelindustrie. Hierbei sollen die Fertigungsabläufe im Hinblick auf die gesamt benötigte Bearbeitungszeit optimiert, im Speziellen minimiert, werden. Die Produktion der Farben findet dabei auf mehreren Maschinen statt. Der Produktionsplan ergibt sich durch die Aufteilung der Herstellungsprozesse auf die Maschinen innerhalb festzulegender Zeitintervalle. Diese Zuordnung wird auch "Scheduling" genannt.

2.1 Scheduling

Das Scheduling befasst sich mit der Verteilung von Jobs auf Maschinen, die in einem bestimmten Zeitintervall auf diesen ausgeführt werden. Die folgende Definition führt die Menge aller Maschinen und die auf diesen zu verteilenden Jobs ein.

> "Es sei eine Menge $M = \{M_1, ..., M_m\}, m \in \mathbb{N}$ von Maschinen sowie eine Menge $J = \{J_1, ..., J_n\}, n \in \mathbb{N}$ von Jobs gegeben. Ein Belegungsplan bzw. Schedule ist für jeden Job eine Zuordnung von einem oder mehreren Zeitintervallen zu einer oder mehreren Maschinen" [7].

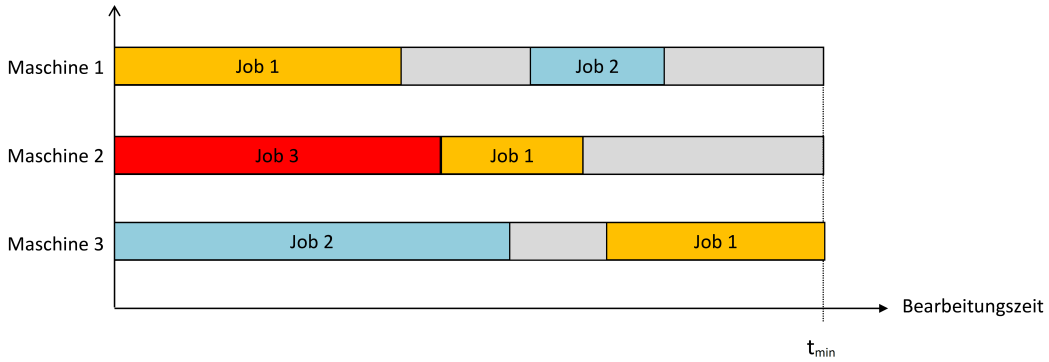
Um zu verhindern, dass mehrere Jobs zur gleichen Zeit auf einer Maschine bzw. ein Job gleichzeitig auf mehreren Maschinen bearbeitet wird, existieren Restriktionen. Sie werden benutzt, um unerwünschtes Verhalten zu eliminieren.

Weiterhin besteht jeder Job aus Operationen, die im Folgenden definiert sind.

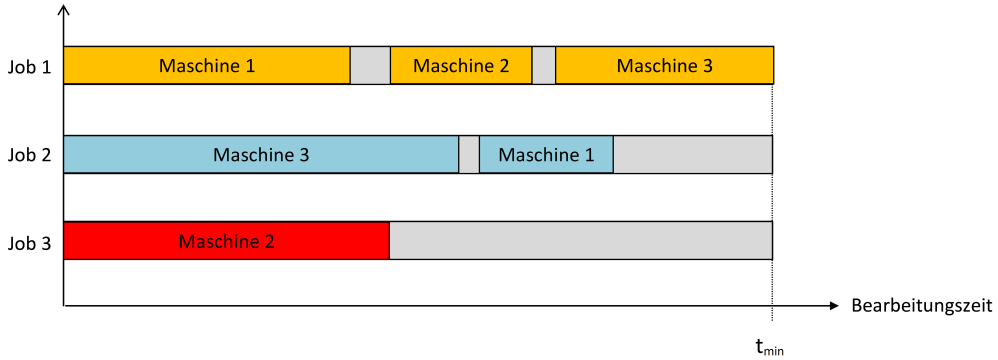
> "Jeder Job J besteht aus einer Anzahl $n \in \mathbb{N}$ von Operationen $o_1, o_2, ...$ Diese Operationen entsprechen einzelnen Arbeitsschritten des Jobs. Des Weiteren wird jeder Operation o eine Bearbeitungszeit p zugewiesen" [7].

Den Zeitintervallen entsprechend lassen sich die Zuordnungen maschinen- oder joborientiert darstellen. Dazu werden Gantt-Diagramme verwendet (siehe Abbildung 2.1).

Nach Schuster [7] existieren verschieden Formen der Klassifikation von Scheduling-Problemen. Es lassen sich drei Formen der Klassifikation unterscheiden. Bei der *Klassifikation nach Jobeigenschaften* lassen sich job-spezifische Eigenschaften, die während der gesamten Bearbeitung gelten, festlegen. Typisches Beispiel dafür stellt die *Präemption* dar. Hierbei handelt es sich um die Festlegung, ob die Bearbeitung



(a) maschinenorientierte Darstellung der Jobs



(b) joborientierte Darstellung der Maschinen

Abbildung 2.1: Gantt-Diagramm

eines Jobs bzw. einer Operation unterbrochen werden und zu einem späteren Zeitpunkt fortgesetzt werden darf. Weiterhin existiert die *Klassifikation nach Maschinenumgebung*. An dieser Stelle soll nur auf das in [7] dargestellte Job-Shop-Problem eingegangen werden.

Job-Shop-Problem Dabei existiert ein Scheduling-Problem mit m Maschinen und n Jobs, wobei mindestens ein Job aus mehreren Operationen besteht. Die Maschinen, auf denen die Jobs ausgeführt werden sind eindeutig. Weiter obliegt die Bearbeitung der Operationen $o_{i,j}$ einer vorgegebenen Reihenfolge für jeden Job $i = 1, ..., n$ der Form $o_{i,1}, o_{i,2}, ..., o_{i,n_i}$ mit n_i als Anzahl für die Operationen von Job i. Soll Präemption verhindert werden, bezeichnet man das Problem als 'no-wait Job-Shop-Problem', welches die Grundlage für das in dieser Arbeit entwickelte Modell darstellt.

Die letzte Art der *Klassifikation nach Optimalitätskriterien* befasst sich mit den Ergebnissen des Lösungsalgorithmus der Modellierungsumgebung. Mögliche Optimalitätskriterien können die insgesamt benötigte Bearbeitungszeit, welche als 'makespan' bezeichnet wird, sein. Andere mögliche Kriterien sind die gesamte bzw. gewichtete Flusszeit, welche sich durch die Wechsel eines Jobs von einer Maschine zur nächsten

ergibt. Gewichtete Flusszeiten können dabei die für einen Stationswechsel benötigte Kosten mit in die Optimalitätskriterien einbeziehen.

Um ein Schedulingproblem modellieren und schließlich lösen zu können, bedarf es mathematischer Optimierungsmodelle. Im Folgenden Kapitel werden diese vorgestellt.

2.2 Optimierungsmodelle

Bei der Lösung eines Optimierungsproblems handelt es sich um eine mathematische Optimierungsaufgabe, "bei der es darum geht, aus der Menge der Lösungen eines Restriktionssystems eine Lösung zu bestimmen, der durch eine Zielfunktion $x_0 = f_0(x_1, x_2, ..., x_n)$ ein Zielwert zugeordnet ist, der von dem Zielwert keiner anderen Lösung übertroffen oder unterschritten wird" [5]. Dazu wird die Lineare Optimierung bzw. Programmierung, ein Hauptbestandteil des Operations Research, zur Lösung solcher Probleme verwendet. Der Zielwert wird dabei meist durch lineare Gleichungen und Ungleichungen eingeschränkt. Grundlage für ein Optimierungsproblem sind oft reale Systeme, bei denen beispielsweise ein Transport- oder Zuordnungsproblem vorliegt. Dabei wird zwischen einem Minimierungs- (zu optimierender Zielwert möglichst gering) oder Maximierungsproblem (zu optimierender Zielwert möglichst groß) unterschieden. Durch Verändern der Vorzeichen lässt sich jedes Minimierungsproblem in ein Maximierungsproblem (und umgekehrt) überführen [5]. Im folgenden Abschnitt wird auf die grundlegenden Unterteilungen von Optmierungsmodellen eingegangen.

2.2.1 Lineares Programm

Gegeben sei eine Matrix $A \in Mat(m \times n, \mathbb{R})$ und Vektoren $b \in \mathbb{R}$ sowie $c \in \mathbb{R}^n$. Unter einem linearen Optimierungsmodell oder einem linearen Programm (kurz LP) versteht man das folgende Problem:

$min(c^T x)$	$Zielfunktion$
unter den Nebenbedingungen:	
$Ax \geq b$	$Restriktionen$
$x \in \mathbb{R}^n_+$	$zulässiger Lösungsbereich$

Die Menge der Vektoren x für die $Ax \geq b$ gilt, seien zulässige Lösungen und gehören zur Menge $S := \{x \in \mathbb{R}_+^n | Ax \geq b\}$ des linearen Programms. Ein Vektor $x_{opt} \in S$ für den $c^T x_{opt} \leq c^T x$ für alle $x \in S$ gilt, wird als optimale Lösung bezeichnet.

Wird ein möglichst geringer Wert der Zielfunktion ($min(c^T x)$)angestrebt, so spricht man von einem Minimierungsproblem, strebt man dagegen einen möglichst großen Zielwert ($max(c^T x)$) an, so spricht man von einem Maximierungsproblem.

2.2.2 Gemischt ganzzahliges lineares Programm

Die ganzzahlige lineare Optimierung (auch ganzzahlige Optimierung), verhält sich wie die Lineare Programmierung. Wobei in der ganzzahligen Optimierung einige oder alle Variablen nur ganzzahlige Werte annehmen dürfen.

Es sei eine Matrix $A \in Mat(m \times n, \mathbb{R})$ und Vektoren $b \in \mathbb{R}$ sowie $c \in \mathbb{R}^n$ gegeben. Unter einem ganzzahligen linearen Optimierungsmodell oder einem ganzzahligen linearen Programm (engl. Integer Linear Programming, ILP) versteht man das Problem:

$min(c^T x)$	$Zielfunktion$
unter den Nebenbedingungen:	
$Ax \geq b$	$Restriktionen$
$x \in \mathbb{Z}_+^n$	$zulässiger Lösungsbereich$

Unter Verwendung des vorgestellten gemischt ganzzahligen linearen Programms kann das Schedulingproblem, für das im Rahmen dieser Arbeit betrachtete Problem, modelliert werden. Grund dafür liegt im zulässigen Lösungsbereich von gemischt ganzzahligen linearen Programmen. Ein Beispiel ist die Betrachtung von Stückzahlen, die sich in der Praxis nur als ganzzahlige Menge realisieren lassen. Daraus ergibt sich das in der Praxis am meisten verwendete *no-wait Job-Shop-Problem* (vgl. [7]) welches für eine Vielzahl von Schedulingprobleme Anwendung findet.

2.3 No-wait Job-Shop-Problem

Das *no-wait Job-Shop-Problem* stellt ein klassisches Job-Shop-Problem (vgl. [7]) erweitert um die Bedingung, dass während der Bearbeitung der einzelnen Operationen

innerhalb eines Job keine Wartezeit vergehen darf (keine Präemption), dar. Diese zusätzliche Bedingung findet in der Praxis eine große Bedeutung, z.B. bei dem im Rahmen dieser Arbeit betrachteten Beispiel aus der Farbmittelindustrie. Möglich wären negative Auswirkungen auf die Qualität der Farbe im Zuge von Wartezeiten, die bei der Befüllung einzelnen Zutaten zur zu mischenden Farbe auftreten würden. Auch aus anderen Industrieanwendungen (siehe [7], S.32) ist eine 'no-wait' Bedingung unumgänglich. Somit eignet sich das *no-wait Job-Shop-Problem* als geeigneter Modellansatz für das im Rahmen dieser Arbeit behandelte Problem. Als Vorlage soll das Modell aus [7], S.34 dienen. Die folgende Tabelle 2.1 stellt die im Modell benötigten Parameter und Variablen vor.

n	$\in \mathbb{N}$	Anzahl der Jobs
m	$\in \mathbb{N}$	Anzahl der Maschinen
J	$= \{J_1, ..., J_n\}$	Menge der Jobs
M	$= \{M_1, ..., M_m\}$	Menge der Maschinen
n_i	$\in \mathbb{N}$	Anzahl der Operationen von Job J_i
N	$= \sum_{i=1}^{n} n_i$	Gesamtanzahl der Operationen aller Jobs
O_i	$= \{o_{i,1}, ..., o_{i,n_i}\}$	Menge der Operationen von Job J_i
$o_{k,l}$	$= (m_{k,l}, p_{k,l}) \in M \times N$	l-te Operation von Job J_k
$m_{k,l}$	$\in M_1, ..., M_m$	l-te Maschine, auf der Job J_k bearbeitet wird
$p_{k,l}$	$\in M$	Bearbeitungsdauer der l-ten Operation von J_k
$P_{k,j}$	$= \sum_{i=1}^{j} p_{k,i}$	kumulierte Bearbeitungszeit von Job J_k bis zur j-ten Operation einschließlich
t_k	$\in \mathbb{N}$	(variable) Startzeit von Job J_k
$t_{k,i}$	$= t_k + P_{k,i}$	Startzeit der i-ten Operation von Job J_k
C_k	$= t_k + P_{k,n_k}$	Fertigstellungszeit des k-ten Jobs
$E_{i,j}$	$= \{\{k, l\} \mid m_{i,k} = m_{j,l}\}$	Zweiermengen von Operationen der Jobs J_i und J_j, die die gleiche Maschine benötigen

Tabelle 2.1: Die Tabelle zeigt die im Modell [7] verwendeten Parameter und Variablen

Das Modell besteht aus einer Menge $J = \{J_1, ..., J_n\}$ Jobs die auf einer Menge $M = \{M_1, ..., M_m\}$ Maschinen gefertigt werden sollen. Dabei besteht jeder Job J_i aus einer Reihe von Operationen $O_i = \{o_{i,1}, ..., o_{i,n_i}\}$. Zu jeder Operation existiert ein Paar $(m_{k,l}, p_{k,l}) \in M \times N$, welches der l-ten Operation von Job k eine Bearbeitungszeit $p_{k,l}$ auf Maschine $m_{k,l}$ zuweist. Weiterhin werden folgende Gleichungen als Restriktionen für das ILP eingeführt:

$min(t_{n+1})$

unter den Nebenbedingungen:

$t_{n+1} - t_i \geq P_{i,n_i}$	$\forall i \in \{1, ..., n\}$
$t_i \geq 0$	$\forall i \in \{1, ..., n\}$
$t_j - t_i \geq P_{i,k} - P_{j,l-1}$	*oder*
$t_i - t_j \geq P_{j,l} - P_{i,k-1}$	$\forall \{k, l\} \in E_{i,j}, i < j \in \{1, ..., n\}$
$t_i \in \mathbb{N}$	$\forall i \in \{1, ..., n+1\}$

Durch die Einführung der binären Variablen y und y^* müssen die folgenden Restriktionen als Ersatz für die obige oder-Verknüpfung dienen, damit sich das Problem als ganzzahliges lineares Programm modellieren lässt:

$$
\begin{aligned}
t_i - t_j + P_{i,k} - P_{j,l-1} &\leq M(1 - y) \\
t_j - t_i + P_{j,l} - P_{i,k-1} &\leq M(1 - y^*) \\
y + y^* &= 1 \\
y, y^* &\in \{0, 1\} \\
M &= 2 \sum_{k=1}^{n} P_{k,n_k}
\end{aligned}
$$

M stellt hierbei eine große Konstante dar und wird im Zusammenhang mit der *bigM-Methode* (siehe [8]) verwendet. Zu beachten ist, dass die obigen Restriktionen verhindern, dass eine Maschine zur gleichen Zeit mehrere Jobs bearbeiten kann. Die Ungleichungen gelten hier für $\forall \{k, l\} \in E_{i,j}, i < j \in \{1, ..., n\}$. $E_{i,j}$ beinhaltet alle Operationspaare der Jobs J_i und J_j, die auf der gleichen Maschine gefertigt werden.

3 Modell

Nun soll mit Hilfe der in Kapitel 2 vermittelten Theorie ein Modell entwickelt werden. Grundlage dafür ist ein Beispiel aus der Farbmittelindustrie. Hierbei werden verschiedene Farben für die Bedruckung von Verpackungsmaterialien hergestellt. Zur Produktion einer Farbe werden in der Praxis eine Vielzahl von Maschinen benötigt. Dabei sind in der Regel unterschiedliche Zutaten (Roh-, Hilfs- und Betriebsstoffe) auf den Maschinen vorhanden (siehe Abbildung 3.1).

3.1 Problembeschreibung

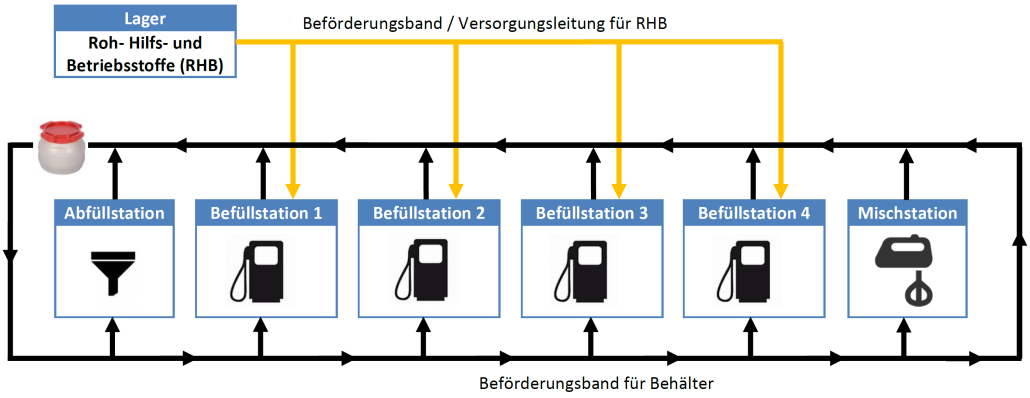


Abbildung 3.1: Modellübersicht

Zur Herstellung einer Farbe werden Lösungsmittel, Farbstoffe und andere Zusatzstoffe benötigt. Farben werden in einem Behälter gemischt und fahren auf Förderbändern die jeweiligen Maschinen ab. Es stehen eine begrenzte Anzahl an Befüllungsstationen zur Verfügung. Daraus resultiert ein Optimierungsproblem hinsichtlich der Parallelfertigung von Farben.

3.2 Aufbau

Das in diesem Kapitel entwickelte Modell versucht den Ablauf der Fertigung von Rezepten auf verschiedenen Maschinen zu optimieren, um so die benötigte Gesamtzeit für eine vorgegebene Menge an zu produzierenden Rezepten zu minimieren. Dabei müssen die Rezepte nach einer Befüllung durch die Stationen gemischt werden. Zum Mischen steht eine Mischstation zur Verfügung. Alternativ können Rezepte während

der Produktion durch den Einsatz eines Behälters mit integrierter Mischfunktion gemischt werden, um so einen Stationswechsel einzusparen. Abschließend wird der Behälter eines Rezeptes an der Abfüllstation entleert. Ein mehrmaliges Anfahren einer Station sollte wenn möglich vermieden werden, da dies zu zeitintensiv wäre. Die Zeit bis zur Fertigstellung aller Rezepte soll minimiert werden. Zusätzlich kann innerhalb eines Rezeptes die Reihenfolge der anzufahrenden Stationen vorgegeben werden. Die folgenden Abbildungen zeigen den zeitlichen Zusammenhang der im Modell benötigten Variablen und Parameter. Um ein Rezept auf einer Station fertigen zu können, wird der Startzeitpunkt dieses Rezeptes auf der jeweiligen Station benötigt. Der Endzeitpunkt dieses Rezeptes entspricht dem Zeitpunkt, an dem das Rezept die Station verlässt. Abbildung 3.2 verdeutlicht den Zusammenhang zwischen Start- und Endzeitpunkt eines Rezeptes auf einer Station. Die Dauer eines Rezeptes auf einer Station wird zum einen durch die Menge der benötigten Zutaten und zum anderen durch eine ggf. vorhandene fixe Stationswartezeit bestimmt. Jede Station verfügt über eine Auswahl an Zutaten. Diese können den Rezepten zugewiesen werden. Dabei entsprechen Mengenangaben äquivalenten Zeiteinheiten, die für die Befüllung einer Zutat benötigt werden. Handelt es sich bei einer Station bspw. um eine Mischstation, so werden dort keine Zutaten befüllt. Prozesse, die auf solchen Stationen durchgeführt werden, unterliegen annahmegemäß einer fixen Bearbeitungszeit. Im Falle einer Mischstation ist die für die Mischung vorgesehene Zeit von der Menge der Zutaten unabhängig und wird somit durch eine fixe Stationswartezeit realisiert.

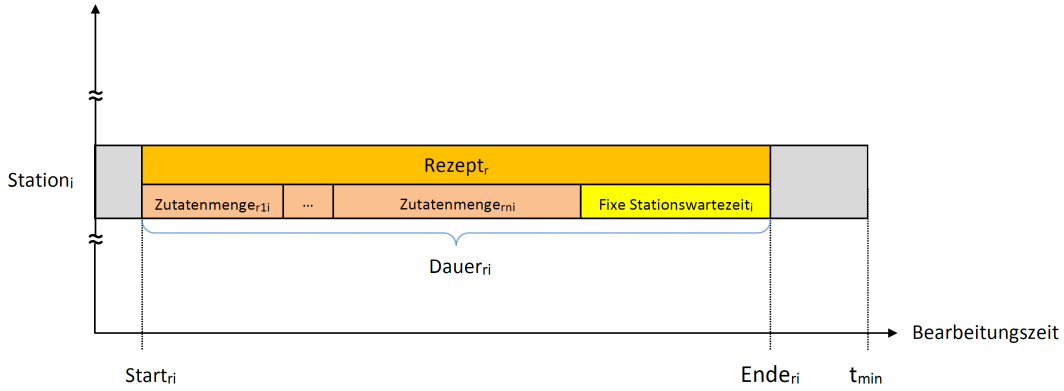


Abbildung 3.2: Rezeptdauer

Grundsätzlich lässt sich anhand der Abbildung erkennen, dass der Startzeitpunkt einer Station zeitlich gesehen nicht nach dem Endzeitpunkt eintreten kann bzw. darf. Weiterhin müssen der Start- und Endzeitpunkt zeitlich vor dem Ende der Bearbeitung aller Rezepte eintreten. Die mindestens benötigte Zeit t_{min} stellt das Ende Bearbeitung dar. Die genannten Bedingungen lassen sich mit Hilfe der folgenden Gleichungen modellieren:

$$
\begin{aligned}
start(r,s) \leq & t_{min} && \forall r \in Rezepte, \forall s \in Stationen \\
ende(r,s) \leq & t_{min} && \forall r \in Rezepte, \forall s \in Stationen \\
dauer(r,s) = & \sum_{z \in Zutaten} menge(r,z,s) + && \forall r \in Rezepte, \forall s \in Stationen \\
& fixeStationszeit(s)
\end{aligned}
$$

Die Variablen müssen weiter verknüpft werden, damit ein zeitlicher Zusammenhang hergestellt wird. Start- und Endzeitpunkt werden über die benötigte Dauer des Rezeptes definiert. Die folgende Gleichung stellt den Zusammenhang dar:

$$
\begin{aligned}
start(r,s) + dauer(r,s) \leq & ende(r,s) + && \forall r \in Rezepte \\
& bigM(1 - rezeptAnStation(r,s)) && \forall s \in Stationen
\end{aligned}
$$

Zu erkennen ist, dass ein weiterer Parameter $rezeptAnStation(r,s)$ benötigt wird. Dieser muss später vorgegeben werden, damit die Bedingung nur eintritt, sofern das Rezept auf der Station bearbeitet werden muss.

Weiterhin müssen Bedingungen zur Kollisionsvermeidung von mehreren Rezepten auf einer Station gesetzt werden. Es muss gewährleistet werden, dass Rezepte innerhalb einer Station nicht kollidieren. Abbildung 3.3 verdeutlicht den Sachverhalt. Jedes Rezept verfügt über eine Start- und Endzeit auf dieser Station. Durch den Bezug der Start- und Endzeitpunkte lässt sich eine Kollision vermeiden. Abhängig von der Reihenfolge der Rezepte auf der Station ergeben sich unterschiedliche Beziehungen der Zeitpunkte. Um die Reihenfolge bestimmen zu können, wird eine Variable $rezeptReihenfolge(s, r_1, r_2)$ benötigt.

Die Kollisionsvermeidung kann durch die folgenden Gleichungen modelliert werden.

$$
\begin{aligned}
ende(r_1,s) \leq & start(r_2,s) + && \forall r_1, r_2 \in Rezepte\{r_1 \neq r_2\} \\
& bigM(1 - reihenfolge(s, r_1, r_2)) + && \forall s \in Stationen \\
& bigM(2 - rezeptAnStation(r_1,s) - \\
& rezeptAnStation(r_2,s))
\end{aligned}
$$

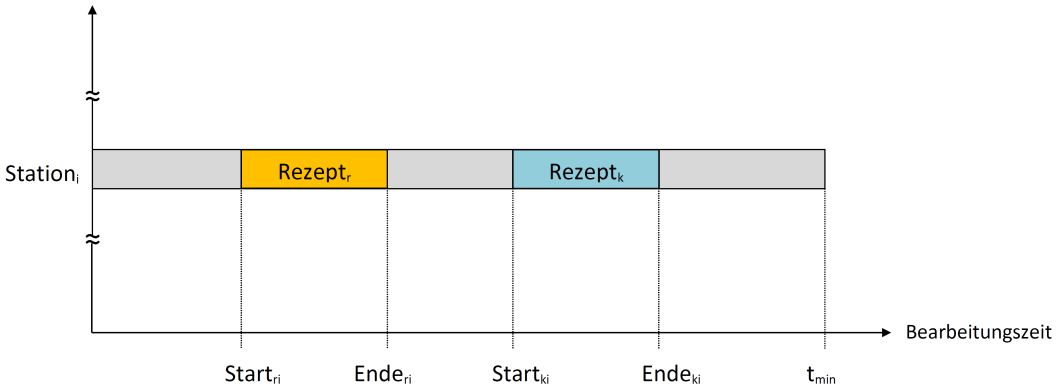


Abbildung 3.3: Rezeptreihenfolge

$$
\begin{aligned}
ende(r_2, s) \leq & start(r_1, s) + && \forall r_1, r_2 \in Rezepte \{r_1 \neq r_2\} \\
& bigM(1 - reihenfolge(s, r_2, r_1)) + && \forall s \in Stationen \\
& bigM(2 - rezeptAnStation(r_1, s) - \\
& rezeptAnStation(r_2, s))
\end{aligned}
$$

Hierbei werden die möglichen Reihenfolgen durch die Variable $reihenfolge(s, r_1, r_2)$ und $reihenfolge(s, r_2, r_1)$ mit $s \in Stationen$ und $r_1, r_2 \in Rezepte$ realisiert. Durch eine bigM-Methode werden die Ungleichungen in Abhängigkeit der jeweiligen Reihenfolge aktiviert bzw. deaktiviert. Weiterhin können beide Bedingungen erst aktiviert werden, sofern beide Rezepte auf den betrachteten Station bearbeitet werden müssen. Dies wird über den Parameter $rezeptAnStation(r \in Rezepte, s \in Stationen)$ realisiert. Dieser Parameter wird exogen vorgegeben und ist abhängig von dem Vorhandensein gewählter Zutaten auf den einzelnen Stationen.

Für die $reihenfolge(s \in Stationen, r_1 \in Rezepte, r_2 \in Rezepte)$ müssen weitere Bedingungen eingeführt werden. Nimmt die boolesche Variable den Wert $true$ an, so wird Rezept r_1 auf der Station s vor dem Rezept r_2 abgearbeitet. Nimmt Sie den Wert $false$ an, wird Rezepte r_2 zuerst bearbeitet. Aus den Überlegungen ergibt sich folglich für die Variable $reihenfolge(s, r_2, r_1)$ das invertierte Verhalten zu $reihenfolge(s, r_1, r_2)$. Durch die folgenden Gleichungen kann das Verhalten modelliert werden:

$$rezeptReihenfolge(s, r_1, r_2) + rezeptReihenfolge(s, r_2, r_1) = 1 \quad \{r_1 \neq r_2\}$$
$$rezeptReihenfolge(s, r_1, r_2) + rezeptReihenfolge(s, r_2, r_1) = 0 \quad \{r_1 = r_2\}$$
$$\forall r_1, r_2 \in Rezepte, \forall s \in Stationen$$

Zu beachten ist, dass die Reihenfolge von allen Rezeptpaaren r_1 und $r_2 \in Rezepte$, die an einer Station $s \in Stationen$ bearbeitet werden, festgelegt sein muss.

Muss ein Rezept auf mehreren Stationen gefertigt werden, besteht ebenfalls Bedarf der Kollisionsvermeidung. Es muss verhindert werden, dass ein Rezept zur gleichen Zeit auf mehreren Stationen abgearbeitet wird. Abbildung 3.4 veranschaulicht den Sachzusammenhang. Nachdem ein Rezept auf Station $i \in Stationen$ fertig bearbeitet wurde, muss der Behälter in dem das Rezept produziert wurde zur Station $j \in Stationen$ transportiert werden. Das Rezept darf während dieses Transportes nicht an Stationen bearbeitet werden. Die benötigte Zeit für den Stationswechsel lässt sich über eine Entfernungsmatrix mittels der Bediensoftware einstellen.

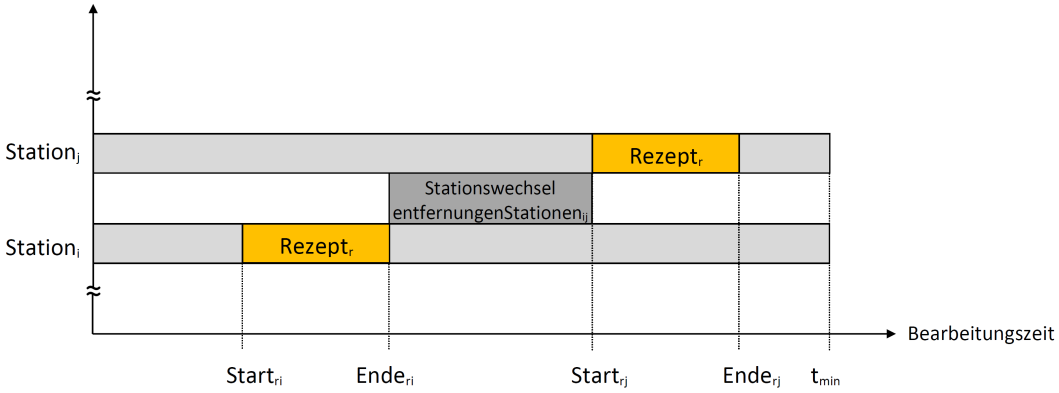


Abbildung 3.4: Stationsreihenfolge

Vergleichbar mit der Kollisionsvermeidung bei der Abarbeitung mehrere Rezepte auf einer Station muss auch hierbei eine Reihenfolge festgelegt werden. Dazu wird der Parameter $reihenfolge(s_1, s_2, r)$ eingeführt. Zu beachten ist, dass es sich hierbei um einen Parameter und um keine Variable handelt. Der Parameter wird exogen gesetzt. Die folgenden Gleichungen modellieren die Kollisionsvermeidung:

$$
\begin{aligned}
ende(r, s_1) + entfernung(s_1, s_2) \leq & start(r, s_2) + \\
& bigM(1 - reihenfolge(s_1, s_2, r)) + \\
& bigM(1 - rezeptAnStation(r, s_1)) + \\
& bigM(1 - rezeptAnStation(r, s_2)) \\
& \\
& \forall r \in Rezepte, \forall s_1, s_2 \in Stationen\{s_1 \neq s_2\}
\end{aligned}
$$

$$
\begin{aligned}
ende(r, s_2) + entfernung(s_2, s_1) \leq & start(r, s_1) + \\
& bigM(1 - reihenfolge(s_2, s_1, r)) + \\
& bigM(1 - rezeptAnStation(r, s_1)) + \\
& bigM(1 - rezeptAnStation(r, s_2)) \\
& \\
& \forall r \in Rezepte, \forall s_1, s_2 \in Stationen\{s_1 \neq s_2\}
\end{aligned}
$$

3.3 Grenzen des Modells

Das beschriebene Modell bietet die Möglichkeit Maschinenbelegungspläne zu erstellen. Das Modell versucht die Realität möglichst genau abzubilden, jedoch sind einige Einschränkungen zu beachten.

Zur Fertigung werden Behälter benötigt, was in der Realität zu einem Engpass führen kann, da eine endliche Menge an Behältern zur Verfügung steht. Dieser Aspekt findet in der Modellbildung keine Beachtung. Das Verfahren eines Behälters von einer Station zur nächsten verursacht zeitliche Verzögerungen im Hinblick auf die zu minimierende Gesamtproduktionszeit. Das Modell beachtet dabei die Laufzeit eines Behälters auf Basis der physikalischen Verbindung beider Stationen durch ein Förderband. Die Laufzeit ergibt sich hierbei aus der tatsächlichen realen Laufzeit des Behälters im Fall keines Konfliktes durch andere Behälter, die ebenfalls Teile des Förderbandes für einen Stationswechsel benutzen. Die Einteilung des Förderbandes in Sektionen wäre denkbar, um eine Kollisionsvermeidung zu realisieren. Dabei wäre jede Sektion nur durch einen Behälter belegt und Staus würden zu einer weiteren Wartezeit führen. Alternativ kann die theoretisch benötigte Zeit mit einem Korrekturterm versehen werden - ein prozentualer Aufschlag wäre denkbar.

Eine weitere Grenze stellt die Rohstoffauswahl auf mehreren Stationen dar. Ist ein Rohstoff auf mehreren Stationen verfügbar, kann das Modell nicht die Station selbstständig auswählen und so ggf. eine kürzere Bearbeitungszeit erreichen. Die stationsspezifische Befüllung der Zutaten obliegen dem Anwender und hat ggf. rudimentären Einfluss auf das Gesamtergebnis. Somit hat eine sinnvolle Vorbelegung von Zutaten auf den Stationen entscheidenden Einfluss auf die Qualität des Ergebnisses. Gleiches gilt für Auswahl von Misch- bzw. Abfüllstationen. Im Falle einer Existenz mehrerer solcher Stationen verbleibt die Entscheidung über die Auswahl der Stationen beim Benutzer.

Jedes Rezept verfügt über eine unterschiedliche Anzahl an Zutaten. In der Realität kann es dazu kommen, das eine bestimmte Reihenfolge beim Befüllen eingehalten werden muss, bspw. eine nach der zu befüllenden Zutatenmenge sortierter Ablauf. Kommt es bei der einzuhaltenden Reihenfolge beim Befüllen dazu, dass eine Station mehrmals angefahren werden muss, so ist dies mit Hilfe des entwickelten Modells nicht möglich. Derzeit kann eine Station höchstens einmal befahren werden.

Nach der Befüllung muss das Rezept gemischt werden. In der Realität stehen zum einem Behälter mit integrierter Mischfunktion und zum anderen Behälter ohne Mischfunktion zur Verfügung. Um eine optimale Lösung des Optimierungsproblems zu finden, muss das Modell entscheiden können, welches Rezept in welchem Behälter produziert wird. Das im Rahmen dieser Arbeit entwickelte Modell benötigt allerdings die Information welches Rezept auf welcher Misch- bzw. Abfüllstation gefertigt wird. Somit kann eventuell keine optimale Lösung gefunden werden.

3.4 Vergleich zur Literatur

Das entwickelte Modell soll mit einem aus der Literatur entnommenen Modell verglichen werden. Dabei interessieren vor allem die Möglichkeiten, sowie die Grenzen beider Modelle. Tabelle 3.1 zeigt eine Gegenüberstellung beider Modelle im Hinblick auf die verwendete Zielfunktion, die vorhandenen Restriktionen und Paramater. Das Modell aus der Literatur stellt dabei ein allgemein gültiges Modell dar, welche das Problem der Verteilung von Jobs auf Maschinen modelliert. Weiterhin besteht jeder Job aus Operationen. Im Vergleich dazu nimmt das im Rahmen dieser Arbeit entwickelte Modell (im weiteren Verlauf als ”entwickeltes Modell” bezeichnet) eine Verteilung von Rezepten (analog zu Jobs) auf Stationen (analog zu Maschinen) vor. Jedes Rezept besteht weiterhin aus Zutaten (analog zu Operationen).

Zielfunktion Sowohl beim allgemeinen Schedulingproblem als auch im hier betrachteten Spezialfall handelt es sich um ein Minimierungsproblem.

In der Literatur wird die Zielfunktion über die Minimierung von t_{n+1} umgesetzt. Hierbei handelt es sich um den Startzeitpunkt von Job n+1. Der Job n+1 ist laut [7] ein "Dummyjob". Dabei entspricht der Startpunkt eines nicht real existierenden Jobs der benötigten Gesamtarbeitszeit. Im Gegensatz dazu wird die Zielfunktion von dem in Kapitel 3 entwickelten Modell über den Endzeitpunkt eines jedes Rezeptes auf allen Stationen dargestellt. Somit entspricht die Gesamtbearbeitungszeit dem Fertigstellungszeitpunkt desjenigen Rezeptes, unabhängig auf welcher Station dieses aktuell bearbeitet wird, welches zeitlich gesehen am spätesten eintritt.

Parameter Durch die Festlegung von Parametern wird das Problem quantitativ beschrieben. Somit lässt sich das Gesamtproblem durch die Kombination von Restriktionen und Parametern unter Hinzunahme einer Zielfunktion vollständig beschrieben.

Das aus der Literatur stammende Modell benötigt drei verschiedene Parameter. Zu beachten ist, dass die Parameter mehrdimensional (n Jobs × m Maschinen) sind. Der erste Parameter $p_{k,l}$ beschreibt die die Bearbeitungsdauer der l-ten Operation von J_k. Hierbei findet eine Aufteilung der Bearbeitungszeit für jede Operation statt. Diese wird job-abhängig gespeichert. Im Vergleich dazu verwendet das entwickelte Modell den dreidimensionalen Parameter *menge(r,z,s)*. Dieser speichert die zur Menge der Zutat benötigte proportionale Bearbeitungszeit z eines Rezeptes r zu jeder Station s. Als nachteilig erweist sich hierbei, dass der Ort der Befüllung einer Zutat bereits festgelegt sein muss. Existiert eine Zutat auf mehreren Stationen, kann dies innerhalb des Modells nicht mehr berücksichtigt werden. Der zweite Parameter $m_{k,l}$ wird zur Festlegung der Reihenfolge der Maschinen eines Jobs benutzt. $m_{k,l}$ gibt dabei die Maschine an, die an l-ter Stelle innerhalb des k-ten Jobs bearbeitet wird. Im Vergleich dazu verwendet das entwickelte Modell den Parameter *reihenfolgeStationen*, der die Reihenfolge für alle Paare an Stationen für ein Rezepte zuweist. $reihenfolgeStationen(s_1, s_2, r)$ ist binär und gibt an, ob s_1 vor s_2 durchlaufen werden soll, sofern Rezept r auf beiden Stationen gefertigt wird. Um dies zu Überprüfen wird ein weiterer Parameter *rezeptAnStation* benötigt, der festlegt, ob ein Rezept auf beide Stationen des Paares aus *reihenfolgeStationen* bearbeitet werden muss. Ist dies nicht der Fall, darf keine Reihenfolge gesetzt werden.

Das entwickelte Modell bietet dabei nicht die Möglichkeit, eine Station mehrmals zu befahren, was hingegen in dem Modell aus der Literatur möglich ist. Da ein Stationswechsel in der Praxis jedoch Zeitverzögerungen verursacht wurde im vorliegenden Modell bewusst auf die Möglichkeit des mehrmaligen Befahrens verzichtet.

Der dritte Parameter vom Modell aus der Literatur $o_{k,l}$ definiert die Reihenfolge der zu bearbeitenden Operationen. Angegeben wird die l-te Operation von Job J_k.

Hierbei kann die Reihenfolge der Operationen innerhalb eines Jobs festgelegt werden. Im entwickelten Modell entfällt diese Möglichkeit, da im betrachteten Praxisproblem eine Befüllung von Zutaten modelltheoretisch als kommutativ betrachtet werden kann. Die Dauer und somit die Belegung der Maschine bzw. Station ist identisch, unabhängig welche Reihenfolge für die Bearbeitung der Operationen gewählt wird.

Restriktionen Die Beschränkungen eines Problems können durch Restriktionen innerhalb eines Modells abgebildet werden. Dabei können Restriktionen als qualitative Beschreibung des realen Problems angesehen werden.

Um eine Zielfunktion minimieren oder maximieren zu können, muss die Variable der Zielfunktion mit Restriktionen verknüpft werden über die sich die Minimierung bzw. Maximierung auswirken kann. Hierbei ist durch die Restriktion $t_{n+1} - t_i \geq P_{i,n_i}$ festgelegt, dass der Startpunkt des "Dummyjobs" zeitlich gesehen später als jeder Startzeitpunkt eines Jobs plus die kumulierte Gesamtbearbeitungszeit aller Operationen dieses Jobs eintritt. Im entwickelten Modell wird dies analog über die Variablen *start*, *dauer* und *ende* realisiert.

Weitere Restriktionen dienen der Vermeidung von paralleler Job bzw. Rezeptbearbeitung und der Vermeidung von parallel genutzter Maschinen bzw. Stationen. Um auszuschließen, dass mehrere Rezepte gleichzeitig auf einer Maschine bearbeitet werden, dient die Restriktion $t_i - t_j + P_{i,k} - P_{j,l-1} \leq M(1-y)$ und $t_j - t_i + P_{j,l} - P_{i,k-1} \leq M(1-y^*)$. Dabei werden die Dauern (Operationen) zweier Jobs mit den jeweiligen Startzeitpunkten verglichen. Die beiden Restriktionen gelten für alle Jobpaare (i,j), die auf der gleichen Maschinen bearbeitet werden. Das entwickelte Modell löst das Problem der Reihenfolge bzw. Parallelbearbeitung von Rezepten bzw. Stationen durch den vorhandenen Parameter $reihenfolge(s_1, s_2, r)$. Somit kann eine Bedingung bezüglich der Start- und Endzeitpunkte jedes Rezeptes auf den Stationen s_1 und s_2 gesetzt werden. Um die Parallelbearbeitung von Rezepten auf einer Station auszuschließen werden ebenfalls Bedingungen mit Hilfe der binären Variablen $reihenfolge(s, r_1, r_2)$ aufgestellt. Hierbei setzt der Modellierungsalgorithmus die Variable und entscheidet darüber, welches Rezept zuerst bearbeitet wird.

Fazit Beide Modelle verfolgen unterschiedliche Ansätze in der Art der Modellierung. Dabei bietet jedes Modell Vor- und Nachteile, die im Folgenden nochmal zusammengefasst werden.

Das im Rahmen dieser Arbeit entwickelte Modell bietet die Vorteile, dass es speziell für das Problem aus der Farbmittelindustrie entworfen wurde. Somit können spezifische Merkmale der realen Fertigung mit eingebunden werden. Beispiel hierfür sind die fixen Stationswartezeiten und die Zeit für den Wechsel von einer Station zur

nächsten. Nachteilig ist bei dem Modell, dass Stationen im Rahmen der Fertigung eines Rezeptes nicht mehrmals durchlaufen werden können. Dies könnte ggf. eine Voraussetzung für die Fertigung einzelner Rezepte sein. Ein weiterer Nachteil ist, dass im Falle von Doppelbelegungen der Zutaten auf mehrere Stationen, das Model nicht ”entscheiden” kann, woher die Zutat optimalerweise zu nehmen ist. Dies muss im Vorfeld durch den Benutzer vorgegeben werden (vgl. Kapitel 3.3).

Das aus der Literatur entnommene Modell beschreibt allgemeine Schedulingprobleme mit unterdrückter Präemption. Das Modell eignet sich generell für die Modellierung des vorhanden Problems, jedoch müssen auf wichtige Parameter verzichtet werden. So bietet das Modell keine Integration von fixen Stationswartezeiten sowie Zeiten für den Stationswechsel, die in der Realität unvermeidbar auftreten. Ggf. ließe sich die das Verhalten über weitere ”Dummyoperationen” annähern bzw. realisieren. Positiv zu bewerten ist hierbei die gegebene Möglichkeit des mehrmaligen Anfahrens von Maschinen bzw. Stationen. Jedoch ist anzumerken, dass das Modell bei der Fertigung eines Jobs, diesen nur auf einer Maschine fertigen kann. Ein Wechsel während eines Jobs von einer Maschine zur nächsten ist nicht möglich. Das Modell unterstellt hierbei, dass es sich um identische Maschinen handelt. Da in dem hier betrachteten Fall mehrere Arten von Maschinen / Stationen existieren, müssten bei Verwendung des Modells aus der Literatur einige Modifikationen vorgenommen werden. Denkbar wäre die Aufteilung eines Rezeptes in mehrere maschinenbezogene Jobs, die wiederum in Operationen, was den einzelnen Zutaten entspricht, aufgeteilt werden müssten. Das Zuordnungsproblem von Zutaten, die auf mehreren Stationen vorhanden sind, betreffend ergeben sich die gleichen Probleme wie bei dem in dieser Arbeit entwickelten Modell.

Modell aus Literatur	**entwickeltes Modell**
Zielfunktion	
$min(t_{n+1})$	$min(ende(r,s))$
Benutzervorgaben (Parameter)	
nicht möglich	$fixeStationswartezeit(s)$
$p_{k,l}$	$menge(r,z,s)$
nicht möglich	$entfernungenStationen(s_1,s_2)$
$m_{k,l}$	$reihenfolgeStationen(s_1,s_2,r)$
nicht benötigt	$rezeptAnStation(r,s)$
$o_{k,l}$	nicht möglich
Restriktionen	
$t_{n+1} - t_i \geq P_{i,n_i}$	$start(r,s) + dauer(r,s) \leq ende(r,s) +$ $bigM(1 - rezeptAnStation(r,s))$ $dauer(r,s) = \sum_z menge(r,z,s) +$ $fixeStationszeit(s)$
	$ende(r,s_1) + entfernung(s_1,s_2) \leq$ $start(r,s_2) +$ $bigM(1 - reihenfolge(s_1,s_2,r)) +$ $bigM(1 - rezeptAnStation(r,s_1)) +$ $bigM(1 - rezeptAnStation(r,s_2))$ $ende(r,s_2) + entfernung(s_2,s_1) \leq$ $start(r,s_1) +$ $bigM(1 - reihenfolge(s_2,s_1,r)) +$ $bigM(1 - rezeptAnStation(r,s_1)) +$ $bigM(1 - rezeptAnStation(r,s_2))$
$t_i - t_j + P_{i,k} - P_{j,l-1} \leq M(1-y)$	$ende(r_1,s) \leq start(r_2,s) +$ $bigM(1 - reihenfolge(s,r_1,r_2)) +$ $bigM(2 - rezeptAnStation(r_1,s) -$ $rezeptAnStation(r_2,s))$
$t_j - t_i + P_{j,l} - P_{i,k-1} \leq M(1-y^*)$	$ende(r_2,s) \leq start(r_1,s) +$ $bigM(1 - reihenfolge(s,r_2,r_1)) +$ $bigM(2 - rezeptAnStation(r_1,s) -$ $rezeptAnStation(r_2,s))$
$M = 2\sum_{k=1}^{n} P_{k,n_k}$	$bigM = 10000$
$y + y^* = 1$	$rezeptReihenfolge(s,r_1,r_2) +$ $rezeptReihenfolge(s,r_2,r_1) = 1$
$t_i \geq 0$	$start(r,s) \geq 0$ $dauer(r,s) \geq 0$ $ende(r,s) \geq 0$

Tabelle 3.1: Die Tabelle zeigt den Vergleich der in beiden Modellen verwendeten Parameter und Variablen

4 Bediensoftware

4.1 Programm

Die Bediensoftware ermöglicht eine benutzerfreundliche Eingabe der in Kapitel 3.2 vorgestellten Modellparameter. Es können Projekte angelegt werden, die durch die Modellierungssoftware GAMS (siehe [4]) gelöst werden. Dazu setzt die Bedienersoftware die eingegebenen Parameter in eine gdx-Datei (GAMS Data Exchange) um. Diese kann von GAMS aus geladen werden. Die im Modell vorhandenen Parameter werden mit den in der gdx-Datei befindlichen Werten initialisiert. Sofern eine optimale Lösung gefunden wurde, kann diese mithilfe der Bedienersoftware (siehe Abbildung 4.1) grafisch angezeigt werden. Somit ist eine benutzerfreundliche Auswertung der Ergebnisse möglich.

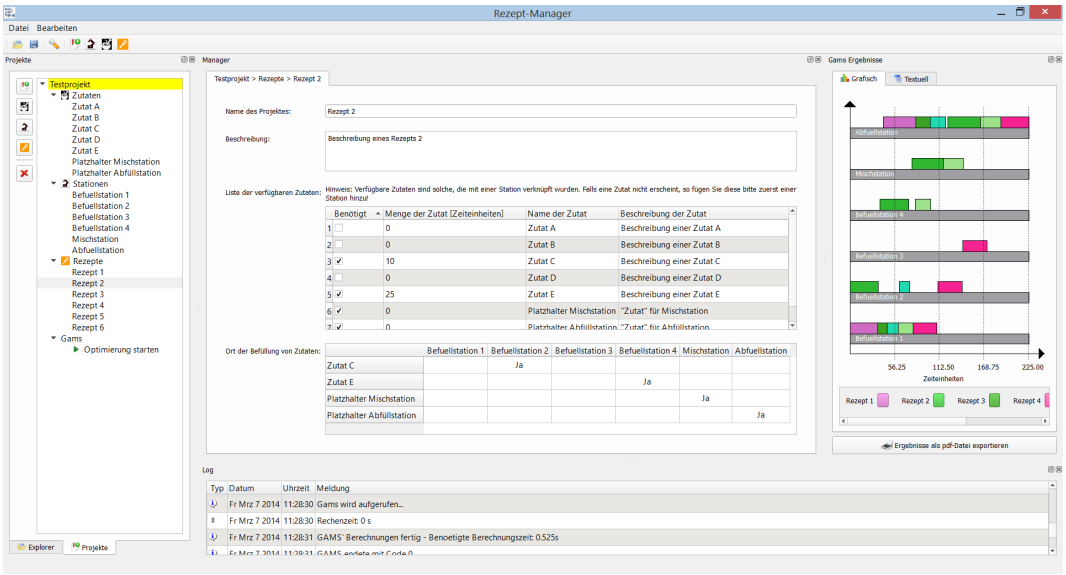


Abbildung 4.1: Übersicht über das Bedienprogramm

Grundlage für jedes Optimierungsproblem ist die Erstellung eines Projektes, in welches anschließend Zutaten, Stationen und abschließend Rezepte erstellt werden können.

4.1.1 Zutaten

Um ein Projekt vollständig erstellen zu können, werden unter anderem Zutaten benötigt, die im Anschluss auf die Stationen verteilt werden müssen. Jeder Zutat

kann ein eindeutiger Name und eine Beschreibung zugewiesen werden. Die Angabe einer verfügbaren Gesamtmenge für eine Zutat ist nicht relevant, da in der Realität für die zu simulierenden Perioden kein Mangel an Zutaten vorliegt.

4.1.2 Stationen

Stationen repräsentieren Maschinen, die während des gesamten Fertigungsprozesses durchlaufen werden können. Dazu zählen in dem hier betrachteten Beispiel aus der Farbmittelindustrie Befüllungs-, Misch- und Abfüllstationen. Stationen können bzw. müssen mit Zutaten belegt werden, sofern es sich um Befüllungsstationen handeln. Die Befüllzeit von Zutaten innerhalb eines Rezeptes auf einer Station ist proportional zu der benötigen Zutatenmenge und wird ausschließlich in Zeiteinheiten angegeben. Zusätzlich besteht die Möglichkeit der Einstellung einer fixen Stationswartezeit. Diese ermöglicht eine Stationsbelegung unabhängig von der verwendeten Menge an Zutaten auf der Station. Die komplette Dauer eines Rezeptes auf einer Station ergibt sich durch Addition der gesamten Befüllzeit der Zutaten zur fixen Stationswartezeit. Im Falle einer Misch- bzw. Abfüllstation zwingt die Wartezeit das jeweilige Rezept zum Verbleib auf einer Station. Das ist notwendig, da es nicht möglich ist einer Misch- bzw. Abfüllstation Zutaten zuzuweisen. Somit würde sich ohne die Stationswartezeit eine Dauer von Null Zeiteinheiten ergeben. Das Mischen und Abfüllen ist in der Regel unabhängig von der im Behälter befindlichen Zutatenmenge. Jedoch dauert der Misch- bzw. Abfüllvorgang eine endliche Zeit, die durch die fixe Stationszeit beschrieben werden kann.

4.1.3 Rezepte

Rezepte bestehen aus verschiedenen Zutaten mit unterschiedlich benötigten Mengen. Die Zutaten können über die Benutzeroberfläche ausgewählt werden. Dabei kann es vorkommen, dass eine ausgewählte Zutat auf mehreren Stationen vorhanden ist. Die erweist sich als sinnvoll, sofern Zutaten existieren, die Hauptbestandteil eines jeden Rezeptes sind. Durch die Belegung der Stationen mit Zutaten entsteht ein *Fahrplan* des Farbbehälters, der das Rezept bearbeitet. Zusätzlich existieren verschiedene Behälter zur Fertigung eines Rezeptes. Es stehen Behälter mit und ohne integrierter Mischfunktion zur Verfügung. Soll das Rezept in einem Behälter ohne Mischfunktion produziert werden, fährt der Behälter vor der Abfüllstation die Mischstation an.

Sind in dem Rezept Zutaten vorhanden, die auf mehreren Stationen vorhanden sind, muss eine Vorentscheidung getroffen werden, die entscheidet welche Zutat von welcher Station entnommen wird. Dabei existieren verschiedene Möglichkeiten der Zuweisung zu den Stationen.

Mögliche Ansätze stellen mengenmäßige sortierte Zuordnungen dar. Hierbei könnte die Zutat mit der größten Menge zuerst befüllt (auf einer nach bestimmten Kriterien festzulegenden Startstation) werden und im Anschluss daran die Zutat mit der zweitgrößten Menge usw. hinzugefügt werden. Dabei müsste nach jeder Befüllung geprüft werden, welche die am schnellsten zu erreichende Station ist, auf der die nächste Zutat vorhanden ist. Dabei kann es passieren, dass das Rezept mehrmals auf einer Station bearbeitet wird. Das im Rahmen dieser Arbeit entwickelte Modell lässt eine Mehrfachbelegung einer Station nicht zu. Diese Art der Zuordnung scheidet somit aus.

Die hier verwendete Methode weist die Befüllung den Stationen derart zu, dass dies auf Basis einer Minimierung von Stationswechseln durchgeführt wird. Die Station, auf der sich die maximale Anzahl an Zutaten befüllen lassen, wird zuerst angefahren. Anschließend wird der Prozess mit allen noch nicht angefahrenen Stationen für die noch nicht hinzugefügten Zutaten erneut wiederholt, bis das alle Zutaten hinzufügt wurden. Ein mehrmaliges Anfahren einer Station wird ausgeschlossen. Eine mengenorientierte Stationszuordnung ist hierbei nicht möglich. Lediglich die Anzahl der zu befüllenden Zutaten nimmt mit jedem Stationswechsel stetig ab. Dabei lässt sich keine Aussage über die zu befüllende Menge treffen.

4.2 Gams

4.2.1 Grundlagen

GAMS steht für 'General Algebraic Modeling System' [4] und ist eine algebraische Modellierungssprache, die zum Lösen von mathematischen Optimierungsproblemen genutzt werden kann. Im klassischen Operations Research-Bereich kommen überwiegend lineare und ganzzahlige Optimierungsprobleme zum Einsatz. Gams bietet zu dem die Möglichkeit neben linearen und ganzzahligen Probleme auch quadratische und nichtlineare Optimierungen zu lösen. Dazu stehen verschiedene Lösungsalgorithmen zur Verfügung. Im Rahmen dieser Arbeit wurde *CPlex* [1] verwendet. Laut der Gams-Dokumentation [3] nutzt *CPlex* für gemischt-ganzzahlige Optimierungsprobleme einen *Branch-And-Cut*-Algorithmus [2], welcher das gesamte Problem in eine Vielzahl kleinerer Unterprobleme teilt, diese löst und abschließend daraus eine Gesamtlösung bildet.

4.2.2 Integration

Die Verbindung zwischen der Software und Gams wird über den Austausch von Informationen hergestellt. Dazu werden so genannte *Gams Data Exchange*-Dateien

(GDX-Dateien) verwendet. Diese dienen sowohl dem Zuführen der Daten zu Gams als auch dem Speichern der Ergebnisse. Über die Programmierschnittstelle von Gams (engl. Application Programming Interface, API) wird es dem Programm ermöglicht, GDX-Dateien zu verarbeiten.

Zu Beginn der Optimierungsausführung werden die relevanten Daten aus der internen Datenbank gesammelt und per GDX-Datei exportiert. Darin enthalten sind die für das Modell relevanten Parameter (Entfernungen zwischen Stationen, ...), sowie die Mengen der Definition der verschiedenen Variablentypen (Vorhandene Stationen, ...). In der Modelldatei werden mittels *$LOAD*-Befehl diese Eingabedaten eingelesen und den jeweiligen Parametern zugewiesen.

In der Modelldatei (siehe A.1.2) bewirkt der folgende Befehl, dass Gams die Werte der Variablen *start*, *ende*, *dauer* und *tmax* in die Datei *results* schreibt. Die Daten dieser Datei werden wiederum eingelesen, aufbereitet und übersichtlich dargestellt.

$$Execute\_Unload\ 'results', start, ende, dauer, tmin;$$

5 Analyse eines Beispielprojektes

In diesem Kapitel sollen anhand eines selbst gewählten Beispielprojektes die Ergebnisse eines Simulationsergebnisses untersucht werden. Weiterhin soll durch Einführung einer weiteren Station die mindestens benötigte Gesamtzeit weiter minimiert werden.

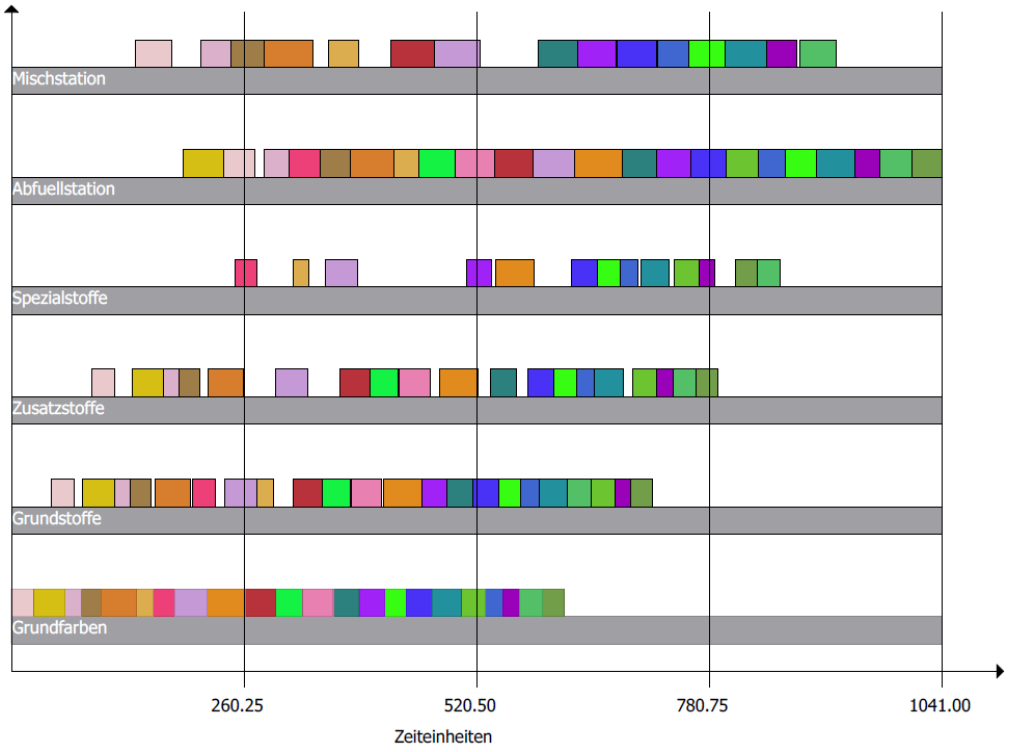
5.1 Projektbeschreibung

Das Beispielprojekt beinhaltet dreizehn Zutaten, die zur Farbenherstellung benutzt werden. Es existieren sechs Stationen, auf die die Zutaten eingeteilt wurden. Die Zutaten wurden entsprechend nach Grundfarben, Grundstoffe, Zusatzstoffen, Spezialstoffen, Mischstationen und Abfüllstationen aufgeteilt. Grundlage für die Rezepte war eine Tabelle von Echo Online [6]. Es wurden insgesamt zweiundzwanzig verschiedenen Farben erstellt. Dabei verfügt über die Hälfte der Rezepte über einen Behälter mit integrierter Mischfunktion. Im Anschluss wurde das Modell simuliert.

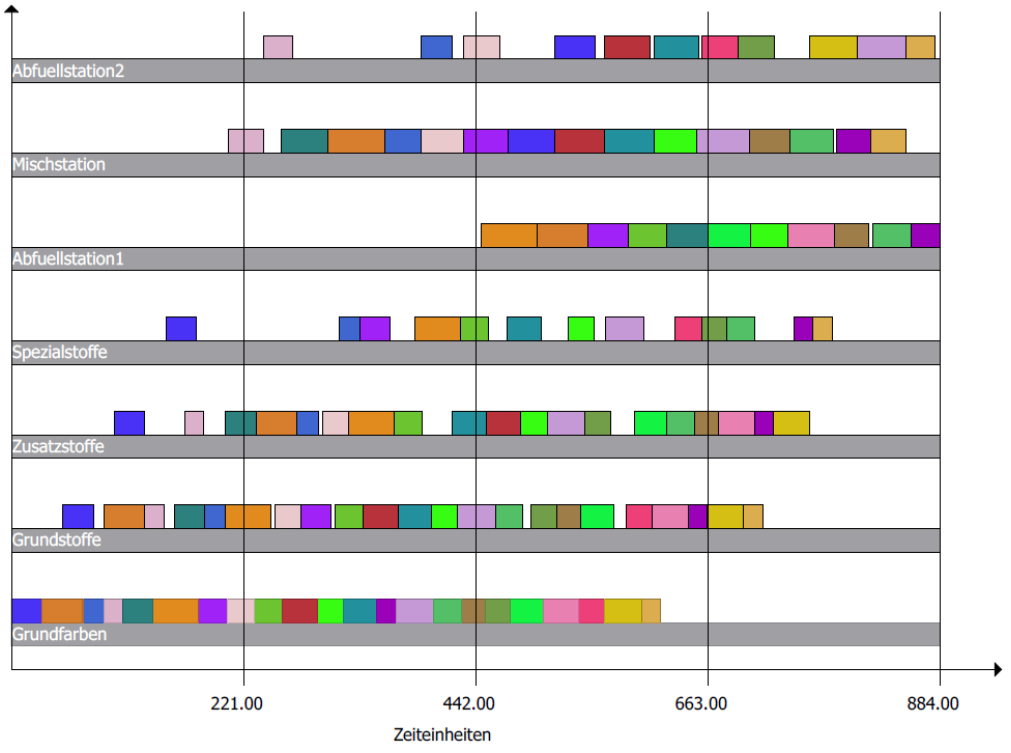
5.2 Ergebnisanalyse

Abbildung 5.1 (oben) zeigt den Fertigungsplan nach der Simulation. Die Abbildung zeigt, dass ein Engpass auf der Abfüllstation zu erkennen ist. Dies erweist sich als nachteilig, da die Station permanent beansprucht wird und somit für keine Wartungsarbeiten etc. zur Verfügung steht. Ein möglicher Ansatz ist die Entlastung der Station durch Erstellen einer weiteren Abfüllstation. Das Ergebnis nach der Simulation mit einer weiteren Abfüllstation ist in Abbildung 5.1 (unten) dargestellt. Einerseits reduziert sich die gesamte Produktionszeit um 15 %, andererseits wird damit die Auslastung der ersten Abfüllstation verringert. Erkennbar sind Zeitschlitze, die Wartungsarbeiten etc. ermöglichen.

Durch die Einführung einer weiteren Station entstehen zusätzliche Kosten. Als weiterer Planungsschritt muss eine Rentabilitätsrechnung durchgeführt werden. Dabei werden die anfallenden Kosten für die Erstellung einer weiteren Station gegen die Einsparung hinsichtlich der kürzeren Bearbeitungszeit im Hinblick auf die in der Zukunft zu erwartende Produktions- und Absatzmenge gegengerechnet. Die daraus resultierende Amortisationszeit entscheidet über die Einführung einer neuen Station, worauf an dieser Stelle nicht weiter eingegangen wird.



(a) vorher: eine Abfüllstation



(b) nachher: zwei Abfüllstationen

Abbildung 5.1: Stationsorientierter (maschinenorientierter) Belegungsplan für ein Beispielprojekt bestehend aus 22 Rezepten

5.3 Optimierungsparameter

Gams bietet die Möglichkeit die Simulation zu parametrieren. Dabei lassen sich beispielsweise die maximale Simulationszeit einstellen. Eine weitere Möglichkeit ist das Einstellen des relativen Optimalitätskriteriums. Die Angabe gibt die relative Abweichung zum relaxierten Optimum im Falle eines linearen Programms, bei dem das Ganzzahligkeitskriterium nicht erfüllt sein muss, an. Abbildung 5.2 zeigt die Auswirkung des Optimalitätskriteriums und der Gesamtproduktionszeit im Hinblick auf die benötigte Simulationszeit.

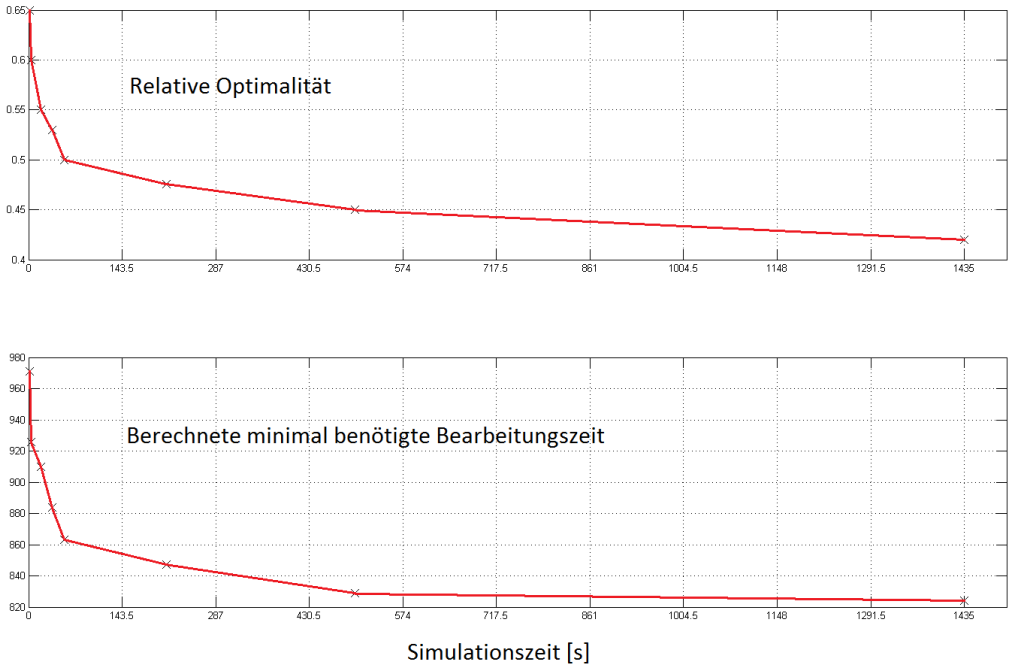


Abbildung 5.2: Gesamtbearbeitungszeit und relatives Optimalitätskriterium in Abhängigkeit von der Simulationszeit

Es ist zuerkennen, dass die Änderung der Gesamtbearbeitungszeit exponentiell mit der benötigten Simulationszeit abnimmt. Somit ist der benötigte zeitliche Mehraufwand für ein besseres Optimalitätskriterium im Hinblick auf die eventuell verbesserte Gesamtproduktionszeit nicht immer lohnenswert. Hierbei soll lediglich gezeigt werden, dass die Einstellung einer gewünschten relativen Optimalität erhebliche Auswirkungen auf die Simulationszeit haben kann.

6 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde ein Modell für ein Optimierungsproblem aus der Farbmittelindustrie entwickelt. Das entwickelte Modell erlaubt die Erstellung eines Maschinenbelegungsplans unter Einbeziehung zuvor eingestellter Parameter. Die programmierte Software ermöglicht eine benutzerfreundliche Erstellung kompletter Projekte bestehend aus Zutaten, Rezepten und Stationen. Zusätzlich können problemspezifische Einstellungen parametriert werden. Zur Auswertung der Ergebnisse einer Modellsimulation steht eine grafische Anzeige in der Bediensoftware zur Verfügung.

In Ansätzen ist es gelungen ein möglichst realistisches Modell, welches viele Umwelteinflüsse einbezieht, zu modellieren. Jedoch existieren einige Einschränkungen. Diese betreffen die Reihenfolge bei der Befüllung innerhalb eines Rezeptes. Ein mehrmaliges Durchlaufen einer Station innerhalb der Fertigung eines Rezeptes ist mit dem entwickelten Modell nicht möglich. Weitere Einschränkungen sind die fehlende Kollisionsvermeidung von Behältern bei einem Stationswechsel. In der Realität ist eine Kollisionsvermeidung unumgänglich, da die Behälter auf Transportbändern befördert werden, was stets eine Überprüfung bzw. Beeinflussung der Position von Behältern erfordert.

Weiterhin wurde das entwickelte Modell mit dem Modell aus der Literatur verglichen. Dabei wurde gezeigt, dass beide Modelle ähnliche Ansätze für die in Schedulingproblemen enthalten Grundprobleme, wie die Parallelität auf Maschinen, verfolgen. Jedoch handelt es sich bei dem Modell um ein allgemeines Modell, welches problemspezifische Eigenschaften nicht berücksichtigt.

Um das Modell in der Zukunft praxisnah einsetzen zu können, wäre die Berücksichtigung weiterer Aspekte ggf. von Interesse. Eine Produktionsanlage verfügt über Förderbänder, welche von allen Behältern benutzt werden, um einen Stationswechsel durchführen zu können. Die Implementierung einer Positionsüberwachung und Kollisionsvermeidung ist eine Möglichkeit, um den Ablauf in der Praxis abzubilden. Über die Förderbänder gelangen die Behälter nach der Befüllung zu Misch- und Abfüllstationen. Aktuell müssen den Rezepten im Bedienprogramm ggf. Misch- und Abfüllstationen zugewiesen werden. Sofern mehrere Misch- bzw. Abfüllstationen existieren, sollte das Modell ”selbstständig” entscheiden können, welche der Stationen benutzt werden sollte, um zu einem optimalen Ergebnis zu gelangen. Die Entscheidung, ob ein Rezept in einem Behälter mit integrierter Mischfunktion produziert wird, verbleibt derzeit beim Anwender. In Zukunft sollte die Anzahl der zu Verfügung stehenden Behälter, sowohl mit als auch ohne Mischfunktion, parametrierbar

sein. Die Verteilung der Rezepte auf die Behälter sollte durch die Modellierung gelöst werden. Um mögliche Verzögerungen im Ablauf verhindern zu können, wäre die Einführung einer maximalen Bearbeitungszeit für Rezepte eine sinnvolle Erweiterung. Sofern das Rezept Zutaten benötigt, die auf mehreren Stationen vorhanden sind, muss eine Möglichkeit geschaffen werden, die Zuweisung über Restriktionen optimal zu lösen.

Zusammenfassend wurde ein Modell entwickelt, welches unter den genannten Einschränkungen Fertigungspläne generieren kann. Mit Hilfe dessen lassen sich Auslastungen und Engpässe auf Stationen erkennen und mögliche Gegenmaßnahmen eruieren. Zusätzlich wurden einige Ansätze vorgestellt, die die Qualität eines Ergebnisses positiv beeinflussen können.

Literaturverzeichnis

[1] Corporation, IBM: *CPLEX Optimizer*. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html. Version: 2014

[2] George Nemhauser, Laurence W.: *Integer and Combinatorial Optimization*. Wiley Interscience, 1988

[3] GmbH, GAMS S.: *CPLEX 12*. http://www.gams.com/dd/docs/solvers/cplex.pdf. Version: 2014

[4] GmbH, GAMS S.: *General Algebraic Modeling System*. http://www.gams.com/. Version: 2014

[5] Lübbecke, Prof. Dr. M.: *Mathematisches Optimierungsproblem*. http://wirtschaftslexikon.gabler.de/Archiv/133136/mathematisches-optimierungsproblem-v6.html. Version: 2014

[6] online, Echo: *Farbmischtabelle HKS*. http://www.echo-online.de/storage/med/home/echodruck/733_farbmischtabelle_HKS_A4.pdf. Version: 2014

[7] Schuster, Christoph J.: *No-wait Job-Shop-Scheduling: Komplexität und Local Search*, Universität Duisburg-Essen, Diss., 2003

[8] Stobitzer, Christian: *Big-M-Methode bei linearen Optimierungsproblemen*. http://www.orklaert.de/big-m-methode.php. Version: 2014

Eidesstattliche Erklärung

Ich, Richard Spiegelberg, Matrikel-Nr. 27 46 25, versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema

Ganzzahlige Optimierung für ein Zeitablaufsteuerungsproblem aus der Farbmittelindustrie

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Aachen, den 20. März 2014

Richard Spiegelberg

A Anhang

A.1 Gams-Modell

A.1.1 Variablen und Parameter

A.1.2 Code für das Programm GAMS

Tabelle A.1: Die Tabelle zeigt eine Beschreibung der im Modell verwendeten Parameter und Variablen

Variable	**Beschreibung**	**Wertebereich**
r, r_1, r_2	Index für ein Rezept	$\in Rezepte$
s, s_1, s_2	Index für eine Station	$\in Stationen$
z	Index für eine Zutat	$\in Zutaten$
$start(r, s)$	Startzeitpunkt vom Rezept r an Station s	$\in \mathbb{Z}$
$dauer(r, s)$	Dauer vom Rezept r an Station s	$\in \mathbb{Z}$
$ende(r, s)$	Endzeitpunkt vom Rezept r an Station s	$\in \mathbb{Z}$
$rezeptReihenfolge(s, r_1, r_2)$	Wird Rezept r_1 vor Rezept r_2 an Station s bearbeitet	{0,1}
t_{min}	minimal benötigte Zeit um alle Rezepte zu fertigen	$\in \mathbb{Z}$
Parameter	**Beschreibung**	**Wertebereich**
$fixeStationswartezeit(s)$	zusätzliche Wartezeit für jedes Rezept auf der Station s	$\in \mathbb{Z}$
$menge(r, z, s)$	Menge der Zutat z in Zeiteinheiten der Zutaten von Rezept r an Station s	$\in \mathbb{Z}$
$entfernungenStationen(s_1, s_2)$	Entfernung in Zeiteinheiten von Station s_1 nach s_2	{0,1}
$reihenfolgeStationen(s_1, s_2, r)$	Rezept r auf Station s_1 und dann auf Station s_2 bearbeiten?	$\in \mathbb{Z}$
$rezeptAnStation(rezeptr, s)$	Muss Rezept r an Station s gefertigt werden?	{0,1}

```
$Title Bachelorarbeit
Sets
      rezept(*)      Rezeptvariable
      r1(rezept)     Rezeptvariable
      r2(rezept)     Rezeptvariable
      station(*)     Stationsvariable
      s1(station)    Stationsvariable
      s2(station)    Stationsvariable
      zutat(*)       Zutatenvariable
;
Scalar
      bigM / 10000 /
;
Parameter
      fixeStationswartezeit(station)                 Wartezeit auf der Station unabhängig vom Rezept
      menge(rezept,zutat,station)                    Menge der Zutat innerhalb des Rezeptes auf der Station
      entfernungenStationen(station,station)         Entfernungen zwischen den Stationen
      reihenfolgeStationen(station,station,rezept)   Wird Station vor anderen Station innerhalb eines Rezeptes bearbeitet
      rezeptAnStation(rezept,station)                Muss das Rezept an der Station bearbeitet werden
;
Variables
      start(rezept,station)                          Startzeitpunkt für Rezept auf Station
      dauer(rezept,station)                          Dauer für Rezept auf Station
      ende(rezept,station)                           Endzeitpunkt für Rezept auf Station
      tmin                                           Länge der Bearbeitung aller Rezepte
      rezeptReihenfolge(station,rezept,rezept)       Wird Rezept vor anderem Rezept auf Station bearbeitet
;
Positive Variables      start(rezept,station),dauer(rezept,station),ende(rezept,station);
Binary Variables        rezeptReihenfolge(station,rezept,rezept);

Equations
      dauerEq(rezept,station)                        Bearbeitungsdauer eines Rezeptes auf einer Maschine
      rezeptDauerEq(rezept,station)                  Rezeptdauer
      rezeptParallelEq(rezept,station,station)       stellt sicher das jedes Rezept nur auf einer Maschine parallel bearbeitet wird
      rezeptParallelEq2(rezept,station,station)      stellt sicher das jedes Rezept nur auf einer Maschine parallel bearbeitet wird
      stationParallelEq1(station,rezept,rezept)      stellt sicher das auf einer Maschine parallel bearbeitet wird
```

```
    stationParallelEq2(station,rezept,rezept)      stellt sicher das auf einer Maschine parallel bearbeitet wird
    stationParallelEq3(station,rezept,rezept)      stellt sicher das auf einer Maschine parallel bearbeitet wird
    stationParallelEq4(station,rezept,rezept)      stellt sicher das auf einer Maschine parallel bearbeitet wird
    tminEq1(rezept,station)                        mainimale Bearbeitungszeit der Station
    tminEq2(rezept,station)
;

$GDXIN input.gdx
$LOAD  station
$LOAD  s2=station
$LOAD  s1=station
$LOAD  rezept
$LOAD  r2=rezept
$LOAD  r1=rezept
$LOAD  zutat
$LOAD  fixeStationswartezeit
$LOAD  menge
$LOAD  entfernungenStationen
$LOAD  reihenfolgeStationen
$LOAD  rezeptAnStation
$GDXIN

dauerEq(rezept,station) .. dauer(rezept,station) =e= sum(zutat, menge(rezept,zutat,station))+ fixeStationswartezeit(station);

rezeptDauerEq(rezept,station) .. start(rezept,station) + dauer(rezept,station) =l= ende(rezept,station)
                                    + bigM*(1-rezeptAnStation(rezept,station));


rezeptParallelEq(rezept,s1,s2) $(not sameas(s1,s2)) .. ende(rezept,s1) + entfernungenStationen(s1,s2) =l= start(rezept,s2)
                                                       + bigM*(1-reihenfolgeStationen(s1,s2,rezept))
                                                       + bigM*(2-rezeptAnStation(rezept,s1)-rezeptAnStation(rezept,s2));

rezeptParallelEq2(rezept,s1,s2) $(not sameas(s1,s2)) .. ende(rezept,s2) + entfernungenStationen(s2,s1) =l= start(rezept,s1)
                                                         + bigM*reihenfolgeStationen(s1,s2,rezept)
                                                         + bigM*(2-rezeptAnStation(rezept,s1)-rezeptAnStation(rezept,s2));

stationParallelEq1(station,r1,r2) $(not sameas(r1,r2)) .. ende(r1,station) =l= start(r2,station)
                                                          + bigM*(1-rezeptReihenfolge(station,r1,r2))
                                                          + bigM*(2-rezeptAnStation(r1,station)-rezeptAnStation(r2,station));

stationParallelEq2(station,r1,r2) $(not sameas(r1,r2)) .. ende(r2,station) =l= start(r1,station)
                                                          + bigM*rezeptReihenfolge(station,r1,r2)
                                                          + bigM*(2-rezeptAnStation(r1,station)-rezeptAnStation(r2,station));

stationParallelEq3(station,r1,r2) $(not sameas(r1,r2)) .. rezeptReihenfolge(station,r1,r2) + rezeptReihenfolge(station,r2,r1) =e= 1;

stationParallelEq4(station,r1,r2) $(sameas(r1,r2)) .. rezeptReihenfolge(station,r1,r2) + rezeptReihenfolge(station,r2,r1) =e= 0;
```

```
tminEq1(rezept,station) .. ende(rezept,station) =l= tmin;
tminEq2(rezept,station) .. start(rezept,station) =l= tmin;

Model Bachelorarbeit / all / ;
Solve Bachelorarbeit using mip minimizing tmin;

Display "Dauer eines Rezeptes auf einer Station:";
Display dauer.l ;

Display "Startpunkt vom Rezept auf Maschine:";
Display start.l ;

Display "Startet Rezept i vor Rezept j auf Maschine:";
Display rezeptReihenfolge.l ;

Display "Endpunkte vom Rezept auf Maschine:";
Display ende.l ;

Display "Maximale Bearbeitungszeit:";
Display tmin.l ;

Execute_Unload 'results', start, ende, dauer, tmin;
```