Simulation von Scheduling Algorithmen

Verbesserungspotentiale in der Feinplanung

Masterarbeit

vorgelegt von:	Thomas Winkler
Studienrichtung:	Industrielogistik
Matrikelnummer:	0435089
Vorgelegt am:	Lehrstuhl für Informationstechnologie
Vorgelegt bei:	Prof. Dr. Peter Auer

Zusammenfassung

Die Verbesserung des Schedulings von Aufträgen in produzierenden Unternehmen birgt oft enorme Potentiale. Produktions- Auftrags- und Maschinendaten sind meist im ERP-System des jeweiligen Unternehmens vorhanden, es fehlen jedoch oft die notwendigen Werkzeuge sowie das Know-how, auf Basis dieser Daten die Reihenfolgeplanung zu optimieren
Im Rahmen dieser Arbeit wurde ein Simulationsprogramm entwickelt, mit dem es möglich ist, Fertigungen zu simulieren und unterschiedliche Algorithmen bzw. Heuristiken zur Auftragsreihenfolgeplanung zu testen. Dieses Simulationsprogramm wurde eingesetzt, um einen Teilbereich der Pressenfertigung der Böhler Schmiedetechnik GmbH und Co KG zu simulieren.
Die Auftragsreihenfolgeplanung wurde dabei analysiert, neue Algorithmen entwickelt und diese in der Simulation, unter Auswahl repräsentativer Daten, evaluiert. Weiters konnte der differenzierte Einsatz von Prioritätsregeln, die aus der Produktionssteuerung bekannt sind, mit Hilfe der Simulation getestet und deren Einfluss auf eine Auswahl an Kennzahlen bewertet werden.

Abstract

Improved scheduling in production companies sometimes bears enormous potential. Production-, order-, and machine data are usually available in the ERP-System of the company, but often tools and know-how for optimizing the schedule based on this data are not available. In this master thesis a simulation program was developed to simulate production processes and to test various algorithms and heuristics of scheduling. This simulation program was used to simulate some sections of Bohler-Forging's press-shop.
Bohler's scheduling has been analyzed and new algorithms have been developed and evaluated by representative data. Furthermore, specific priority rules which are known from production steering, have been tested and their influence on key logistic performance indicators has been evaluated.

Inhaltsverzeichnis

Abbildungsverzeichnis **IV**

Tabellenverzeichnis **V**

1 Einführung **1**
1.1 Anstoß für die Arbeit . . . 1
1.2 Ziel der Arbeit . . . 2
1.3 Verwendete Technologien . . . 2
1.4 Vorstellung des Unternehmens . . . 3

2 Problemstellung **5**
2.1 Beschreibung des Produktionsprozesses . . . 6
2.2 Scheduling der Presse . . . 7
2.3 Scheduling der Presse vorgelagerter Aggregate . . . 11
2.4 Scheduling der Presse nachgelagerter Aggregate . . . 12
2.5 Auswahl representativer Daten . . . 14

3 Lösungsansätze **15**
3.1 Simulationsprogramm . . . 15
3.1.1 Allgemeines über Simulationsprogramme . . . 16
3.1.2 Events . . . 17
3.1.3 Eventhandler . . . 18
3.1.4 Maschine . . . 20
3.1.5 ProductQueue . . . 21
3.1.6 Einlastungsstrategie - Simulationssteuerung . . . 24
3.2 Scheduling Algorithmen . . . 25
3.2.1 Allgemeines über Scheduling Algorithmen . . . 26
3.2.2 Notation von Scheduling Problemen . . . 26
3.2.2.1 Maschinencharakteristik . . . 27

3.2.2.2 Auftragscharakteristik 29
3.2.2.3 Optimierungskriterien 31
3.3 Beispiele für einfache Lösungsalgorithmen 32
3.3.1 $1 \mid p_j = 1 \mid \sum \omega_i U_i$ 32
3.3.2 $1 \mid \mid \sum U_j$ 33

4 Gewählte Lösungsansätze 35
4.1 Scheduling Aggregate vor und nach der Presse 35
4.1.1 Scheduling Aggregate vor der Presse 35
4.1.2 Scheduling der Aggregate nach der Presse 36
4.1.3 Modell des Produktionsprogramms und der Fertigung . . 36
4.1.4 Verwendete Heuristiken 38
4.1.4.1 Größte Anzahl Folgevorgänge(GAFV) 41
4.1.4.2 Größte Restbearbeitungszeit (GRBZ) 41
4.1.4.3 Frühester Fertigstellungstermin (FFT) 42
4.1.4.4 Kürzeste Schlupfzeit (KSZ) 42
4.1.4.5 Größter Auftragswert (GAW) 42
4.1.4.6 First In - First Out (FIFO) 43
4.1.4.7 Simulationsergebnisse 43
4.2 Lösungsansätze Pressenscheduling 45
4.2.1 Algorithmus Böhler1 45
4.2.2 Algorithmus Böhler2 48
4.2.3 Algorithmus Heuristik1 49
4.2.4 Problembeschreibung als Traveling Salesman Problem . . 51
4.2.4.1 Allgemeines über TSP 52
4.2.4.2 Vergleich TSP - Pressenscheduling 53
4.2.5 Heuristik Greedy 58
4.2.6 Genetische Algorithmen 63
4.2.6.1 Chromosomen und Gene 63
4.2.6.2 Fitness 64
4.2.6.3 Ausgangspopulation 64
4.2.6.4 Crossover ψ 65
4.2.6.5 Elternauswahl 69
4.2.6.6 Mutation ϕ 71
4.2.6.7 Der Algorithmus 72

4.2.6.8 Implementierte Algorithmen 73
4.2.7 Vergleich der Ergebnisse 73

5 Diskussion der Ergebnisse **75**

6 Ausblick **79**

Literaturverzeichnis **80**

Eidesstattliche Erklärung **83**

Abbildungsverzeichnis

1 Darstellung der 3 wesentlichen Fertigungsbereiche 6
2 Spindelpresse SP31 mit 315 MN Presskraft 7
3 Reihung BCA, Hitzewartezeit 00:20:00 10
4 Reihung CBA, Hitzewartezeit 00:00:00 11

5 Klassendiagramm Events 18
6 Klassendiagramm Eventhandler 19
7 Sequenzdiagramm Simulationsschritt 19
8 Klassendiagramm Maschine 21
9 Klassendiagramm ProductQueue 22
10 Sequenzdiagramm ProductQueue 23
11 Sequenzdiagramm Einlastungsstrategie 25
12 Job Shop Fertigung 29
13 Flow Shop Fertigung 29

14 Pressenproblem als Partition Problem 56
15 Überprüfungsalgorithmus Pressenproblem 57
16 Nachbarschaft Vorgang A 59
17 Beispiel für ungünstige Vorgangsauswahl 62
18 Beispiel für Simple Crossover 66
19 Beispiel für Order Crossover 67
20 Beispiel für Cycle Crossover 68
21 Beispiel für Slide Crossover 69
22 Roulette Wheel für Beispielpopulation 70
23 Beispiel einer Mutation 71

Tabellenverzeichnis

1 Beispielaufträge Pressparameter 9
2 Beispielaufträge Ofenparameter 9
3 Beispielaufträge Arbeitspläne und Stückzahlen 13
4 Maschinenbelegung Auftragsreihenfolge ABC 13
5 Maschinenbelegung Auftragsreihenfolge CBA 14

6 Mögliche Werte für Parameter α_1 28
7 Mögliche Werte für Parameter β_{1-6} 30

8 Verwendete Prioritätsregeln 41
9 Simulationsergenisse Prioritätsregeln 44
10 Übersichtstabelle Arbeitsgänge 46
11 Endgültige Auftragsreihenfolge Algo1 47
12 Endgültige Auftragsreihenfolge Algo2 49
13 Endgültige Auftragsreihenfolge Heuristik1 51
14 Beispiel Presswartezeiten Vorgänge 54
15 Beispiel Presswartezeiten Zeiten 55
16 Endgültige Auftragsreihenfolge Greedy Heuristik 61
17 Simulationsergenisse Pressenproblem 74

1 Einführung

1.1 Anstoß für die Arbeit

ERP-Systeme sind in verschiedenen Ausprägungen in den meisten produzierenden Unternehmen ab einer gewissen Größe vorhanden. Auch die Böhler Schmiedetechnik GmbH & Co. KG (BSTG) verwendet ein solches System, um die vorhandenen Ressourcen möglichst effizient für den betrieblichen Ablauf einzusetzen. Im Laufe meiner mittlerweile mehrjährigen Tätigkeit in der Auftragsplanung in diesem Unternehmen, kam die Erkenntnis, dass dieses System zwar die Grobplanung[1] und Vefügbarkeitsprüfung[2] sehr gut unterstützt, im Bereich der Feinplanung bzw. Auftragsreihenfolgeplanung jedoch nicht alle Möglichkeiten ausgeschöpft werden.
Obwohl man sich der Relevanz und der Möglichkeiten zur Optimierung und Kosteneinsparung sehr wohl bewusst ist, fehlt es in diesem Bereich an Know-How und Ressourcen, um mittels gezielter Reihenfolgeplanung die Effizienz in der Fertigung zu erhöhen. Es gibt zwar ein konkretes Projekt, das zum Ziel hat, spezielle im Unternehmen entwickelte Reihungsalgorithmen in das ERP-System zu integrieren, jedoch liegt das Hauptaugenmerk nicht auf den Algorithmen, sondern auf der Implementierung einer lauffähigen Lösung. Bei intensiverer Beschäftigung mit den zur Implementierung vorgesehenen Algorithmen, lag die Vermutung nahe, dass diese noch verbessert werden könnten, womit weitere Effizienzsteigerungen in der Fertigung zu erwarten wären.
Aufgrund eines bereits in der Lehrveranstaltung Software - Engineering in Zusammenarbeit mit 4 Studienkollegen entwickelten Simulationsprogramms

[1] Planungsabteilung, welche sicherstellt, dass Aufträge terminlich und kapazitiv ordnungsgemäß verplant werden. Die Grobplanung hat die Befugnis, freie Maschinenkapazitäten mit Kundenaufträgen, die vom Vertrieb angelegt werden, zu füllen.
[2] Die Verfügbarkeitsprüfung stellt sicher, dass zum Produktionstermin eines Produktes alle notwendigen Produktionsfaktoren vorhanden sind.

entstand die Idee, im Rahmen dieser Diplomarbeit dieses Programm weiterzuentwickeln, bzw. auf die Bedürfnisse der BSTG anzupassen. Die Entscheidung ein solches Simulationsprogramm selbst zu erstellen und keine Standardsoftware zu verwenden, fiel aus dem Grund heraus, dass der Aufwand für die Auswahl geeigneter Standardsoftware und deren Anpassung an die Problemstellung vorab nicht abschätzbar war und nicht denselben Reiz hatte, wie eine Lösung selbständig zu implementieren.
Weiters stand im Vordergrund, die bereits bekannte Programmiersprache Java™ zu verwenden und keine neue Sprache, wie es zur Erweiterung und Anpassung von Standard-Simulationssoftware notwendig gewesen wäre, erlernen zu müssen.

1.2 Ziel der Arbeit

Ziel der Arbeit ist es, mit Hilfe eines selbst implementierten Simulationsprogramms, bestehende Scheduling Algorithmen an realen Daten zu testen und weiterzuentwickeln, bzw. neue Algorithmen zu entwerfen. Die dabei gewonnenen Erkenntnisse sollen dazu verwendet werden, die Feinplanung innerhalb der BSTG zu verbessern und den Produktionsablauf durch gezielte Reihenfolgeplanung effizienter gestalten zu können. Unterschieden wird im Rahmen der Arbeit das Scheduling des Hauptaggregats, vom Scheduling der restlichen Fertigung, welche alle Aggregate außer dem Hauptaggregat beinhaltet. Das Hauptaggregat in der BSTG ist die Spindelpresse, auf welche in Kapitel 2 detaillierter eingegangen wird.

1.3 Verwendete Technologien

Das Simulationsprogramm ist in der Programmiersprache Java™ in der Version 1.6 geschrieben und implementiert einen diskreten Event-basierten Simulationsansatz. Diese Simulation stellt die Umgebung zur Evaluierung der unterschiedlichen Algorithmen und Heuristiken zur Verfügung. Weiters wurden zusätzliche Bibliotheken verwendet, um nicht alle Funktionen selbst implementieren zu müssen. Konkret handelt es sich dabei um folgende Bibliotheken:

- jXLS 0.9.8 (`http://jxls.sourceforge.net`) zum Lesen und Schreiben von Exceldokumenten
- JFreeChart 1.0.13 (`http://jfree.org/jfreechart`) zur Darstellung unterschiedlicher Diagramme
- iText 2.1.7 (`http://www.lowagie.com/iText/`) zur Erstellung von PDF Dokumenten direkt aus Java™

Die Algorithmen zum Scheduling wurden getrennt von der Simulationssoftware ebenfalls in Java™ entwickelt, wobei unterschiedliche Heuristiken und evolutionäre Algorithmen zum Einsatz kamen. Obwohl es sich bei der Problemstellung um kein klassisches Traveling Salesman Problem[3] handelt, stammen die Ideen zur Lösung des Problems aus diesem Bereich und wurden entsprechend adaptiert.
Weiters zum Einsatz kommen Prioritätsregeln, die aus der Produktionslogistik bekannt sind und ebenfalls für den Einsatz in der Simulation implementiert wurden[4]. Neben den verwendeten Heuristiken wurde auch ein Optimierungsansatz implementiert, der darauf basiert, alle möglichen Auftragsreihenfolgen zu berechnen und jene auszuwählen, die die geringste Verzögerung am Hauptaggregat ergibt. Diese optimale Reihenfolge wird dazu verwendet, die Güte der Lösungen von Heuristiken und genetischen Algorithmen bei einer kleinen Anzahl von Aufträgen zu bewerten.

1.4 Vorstellung des Unternehmens

Die Böhler Schmiedetechnik GmbH und Co. KG (BSTG) ist ein Tochterunternehmen der Voest Alpine AG, einem österreichischem Stahlhersteller, mit Hauptsitz in Linz. Das Unternehmen ist spezialisiert auf die Herstellung von Gesenkschmiedestücken für die Luftfahrt, die Energietechnik, sowie Spezialschmiedestücke für den Schiffsbau. In Summe werden derzeit ca. 750 verschiedene Gesenkschmiedeteile hergestellt. Die Losgrößen reichen von 5 bis 100 Stk. Am Standort in Kapfenberg sind derzeit ca. 650 Mitarbeiter beschäftig, der Jahresumsatz betrug im Geschäftsjahr 2008 ca. 190 Mio. EUR.

[3] genaue Erläuterung dazu siehe Kapitel 4.2.4
[4] näheres dazu in Kapitel 4.1.4

Die Produktion erfolgt nach dem Prinzip der Werkstattfertigung, was bedeutet, dass die einzelnen Fertigungsbereiche nach der dort durchgeführten Tätigkeit strukturiert sind, unabhängig davon, für welche Produkte oder an welcher Stelle des Produktionsprozesses diese Tätigkeit benötigt wird. Der wesentliche Vorteil dieser Produktionsform ist das hohe Maß an Flexibilität, das heißt schnell auf sich ändernde Kundenwünsche eingehen und sehr viele unterschiedliche Produkte anbieten zu können. Die Problematiken einer solchen Produktionsform liegen in den langen Durchlaufzeiten, der schwierigen Produktionsplanung, sich scheinbar spontan ändernder Engpässe und der immer wieder auftretenden, oft kapitalintensiven, Zwischenlagerbildung.
Die Produktionsplanung erfolgt in zwei Detaillierungsebenen. Die Grobplanung führt die Verfügbarkeitsprüfung durch und gibt Aufträge zu Produktion frei. Im Rahmen der Verfügbarkeitsprüfung wird das Vorhandensein aller Produktionsfaktoren, insbesondere des Rohmaterials, der Werkzeuge (Gesenke) und Kapazität auf den Hauptproduktionsaggregaten überprüft. In der Regel werden immer so viele Aufträge freigegeben, sodass ein Arbeitsvorrat von ca. 7-10 Arbeitstagen vorliegt.
Die Feinplanung erstellt auf Arbeitsplatzebene Schichtprogramme, welche die Auftragsreihenfolge vorgeben. Dabei werden je nach Arbeitsplatz unterschiedliche Kriterien zur Reihung herangezogen, einerseits um Rüst- und Kapitalbindungskosten zu sparen, andererseits um die erforderliche Termintreue zu gewährleisten.

2 Problemstellung

Die Auftragsreihenfolgeplanung ist ein wesentlicher Bestandteil eines jeden produzierenden Unternehmens mit weitreichende Konsequenzen für alle Unternehmensbereiche, vor allem aber für die Fertigung. Auch in der BSTG wird dem Scheduling von Fertigungsaufträgen, das von den sogenannten Feinplanern vor Ort in der Produktion vorgenommen wird, wesentliche Bedeutung beigemessen. Die Problematik besteht im konkreten Fall darin, eine sowohl für das zu planende Aggregat, als auch eine für die gesamte Fertigung optimale Auftragsreihenfolge zu finden, um einerseits unter idealen Voraussetzungen produzieren zu können und andererseits mit Zielgrößen versehene Logistikkennzahlen wie Termintreue, WIP [5] o.ä. einzuhalten.
Im Fall der BSTG kann diese Problematik in drei Subprobleme unterteilt werden. Der im Rahmen dieser Arbeit betrachtete Fertigungsbereich lässt sich grob in drei Bereiche aufteilen. Der wichtigste Bereich ist jener mit der Spindelpresse SP31 und den beiden Drehherdöfen SWP1 und SWP2, der in der Regel den Engpass der Fertigung darstellt, somit den Durchsatz der gesamten Fertigung bestimmt und auf dessen Planung das größte Augenmerk gelegt wird. In weiteren Verlauf der Arbeit wird dieser Bereich als der Pressenbereich bezeichnet.
Alle Maschinen und Arbeitsstationen, die in den Arbeitsplänen vor dem Pressenbereich anzutreffen sind, sind ebenfalls einem eigenen Bereich zuzuordnen. Dieser wird im weiteren Verlauf als Sägebereich bezeichnet. Besteht nun für den Pressenbereich eine konkrete Reihenfolgeplanung, muss sichergestellt werden, dass die Produkte rechtzeitig vor Ort sind, damit die geplante Reihenfolge im Pressenbereich tatsächlich eingehalten werden kann. Dies sicher zu stellen ist die Herausforderung beim Scheduling des Sägebereichs.
Maschinen und Arbeitsstationen, die in den Arbeitsplänen nach dem Pressen-

[5] Work in Progress, Größe die den Bestand an angearbeiteten Material innerhalb der Fertigung darstellt. Einheit meist Euro, Stück, oder Stunden

bereich angeführt sind, werden im weiteren Verlauf dem Endfertigungsbereich zugeordnet, dessen Aufgabe es ist, die Produkte nach Passieren des Pressenbereiches rechtzeitig zum geplanten Liefertermin fertig zu stellen.
Eine schematische Darstellung dieser drei Fertigungsbereiche mit den zugehörigen Aggregaten und Arbeitsstationen ist in Abbildung 1 ersichtlich.

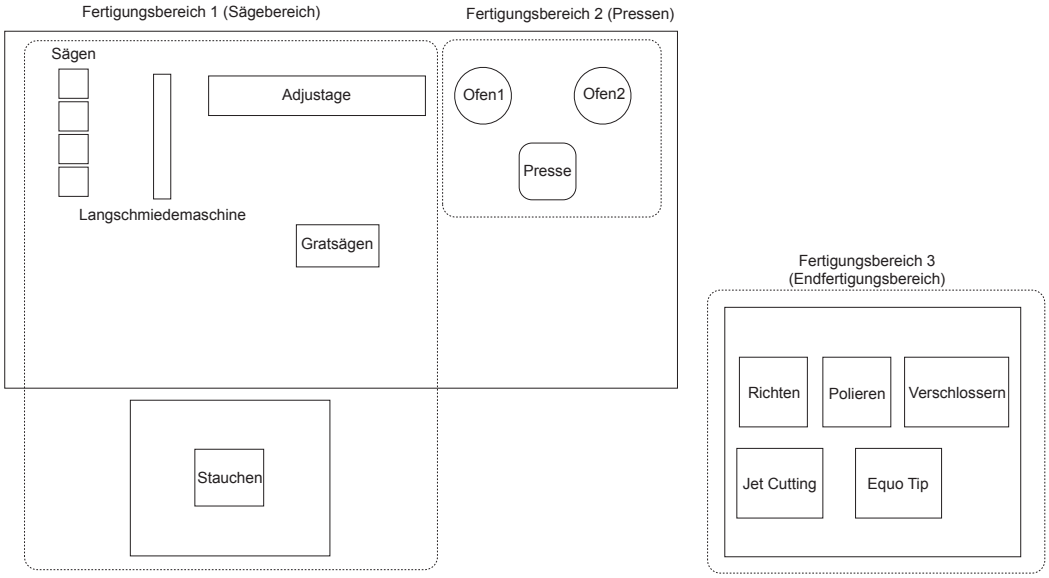


Abbildung 1: Darstellung der 3 wesentlichen Fertigungsbereiche

2.1 Beschreibung des Produktionsprozesses

Der Produktionsprozess weist trotz der in der Herstellung sehr unterschiedlichen Produkte einige Gemeinsamkeiten auf. Begonnen wird immer mit dem Sägen des Vormaterials auf das sogenannte Stöckelgewicht. Vormaterialien sind 3-5 m lange Stangen unterschiedlicher Titan- und Nickellegierungen bzw. Turbinen- und Luftfahrtstählen mit Durchmessern von 50 - 300mm. Nach dem Sägen können unterschiedliche Arbeitsschritte wie Schleifen, Freiformschmieden, Beizen etc. erfolgen, die unter dem Begriff Pressvorbereitung zusammengefasst werden. Das Pressen selbst erfolgt auf einer Spindelpresse mit einer Presskraft von 315 MN, die in Abbildung 2 zu sehen ist.

Abbildung 2: Spindelpresse SP31 mit 315 MN Presskraft

Geschmiedet wird im Gesenkschmiedeverfahren, welches sich vom Freiformschmieden darin unterscheidet, dass das Schmiedestück völlig, oder zu einem wesentlichen Teil von den gegeneinander bewegten Formwerkzeugen, den Gesenken, umschlossen wird. Die in das Gesenk eingebrachte Gravur bestimmt die Form des fertigen Schmiedestücks. Vor dem Schmieden muss das Pressgut in den der Presse vorgelagerten Öfen auf Umformtemperatur gebracht werden. Nach dem Schmieden folgen diverse Schleif-, Wärmebehandlungs- und/oder Prüfvorgänge, die unter dem Begriff Endfertigung zusammenzufassen sind.

2.2 Scheduling der Presse

Aufgrund unterschiedlicher Umformtemperaturen der produzierten Schmiedestücke und damit verbundener Durchwärm[6]- und Haltezeiten[7] in den Öfen, kommt es an der Spindelpresse zu Wartezeiten. Diese Wartezeiten entstehen

[6] Jene Zeit, die benötigt wird, um das gesamte Schmiedestück auf Prozesstemperatur zu bringen.
[7] Jene Zeit, die ein Schmiedestück im Ofen verbringen muss, um die gewünschten metallurgischen Effekte zu erzielen.

genau dann, wenn die Presse bereits frei ist, die Schmiedestücke aber noch im Ofen gehalten werden müssen, da die erforderlicher Haltezeit noch nicht erreicht ist.
Die Herausforderung ist, Algorithmen zu finden um diese Wartezeiten durch möglichst optimale Reihung von Fertigungsaufträgen zu minimieren. Folgende Randbedingungen müssen dabei beachtet werden.

- Es existieren zwei Öfen, zwischen denen gewechselt werden kann.
- Aufträge können nicht geteilt werden, es muss immer das gesamte Los produziert werden.[8]
- In welchem Ofen ein Produkt gewärmt werden muss, ist im Arbeitsplan festgelegt.
- Rüst-, Bearbeitungs- und Wärmezeiten müssen eingehalten werden.
- Das Einlegen von Pressgut in den Ofen darf bereits bei niedrigerer Temperatur als der erforderlichen Prozesstemperatur erfolgen. Nachdem auf die erforderliche Temperatur aufgeheizt wurde, muss das Pressgut für die in der Haltezeit definierten Zeitspanne im Ofen verbleiben.
- Das Einlegen von Pressgut erfolgt immer in der Taktzeit des dem Wärmevorgang folgenden Pressvorgangs, sodass eine konstante Versorgung der Presse mit Pressgut gewährleistet ist.
- Sollte es aufgrund unterschiedlicher Taktzeiten aufeinanderfolgender Aufträge nicht möglich sein, die erforderliche Taktzeit einzuhalten, kann dies vernachlässigt werden, da die geplante Wärmdauer des Pressguts nur geringfügig unter- bzw. überschritten wird, was metallurgisch nicht relevant ist.
- Ist die aktuelle Ofentemperatur höher als die vorgegebene Prozesstemperatur muss mit dem Einlegen in den Ofen gewartet werden, bis der Ofen auf die erforderliche Temperatur abgekühlt ist.

[8] Theoretisch wäre eine Teilung möglich, jedoch ist dies aufgrund der dadurch entstehenden hohen Rüstkosten betriebswirtschaftlich nicht sinnvoll.

- Wenn auf einen Pressvorgang direkt ein Wärmvorgang folgt, muss dieser unmittelbar nach dem Pressen eingeplant werden. Dies ist bei Fertigung mit mehreren Hitzen der Fall.[9]
- Bei der Wärmezeit sind die minimale[10] und die maximale Wärmezeit[11] einzuhalten.

Um die Problematik zu verdeutlichen, folgendes Beispiel: Die in Tabelle 1 und Tabelle 2 gelisteten Aufträge sollen in eine Reihenfolge gebracht werden, sodass an der Presse möglichst geringe Wartezeiten entstehen. Der Ofengradient für die beiden Öfen vor der Presse beträgt dabei 60°C/h. Eine weitere Annahme sei, dass zu Beginn der Produktion die Ofentemperatur der erforderlichen Temperatur des ersten Auftrags entspreche. Weiters ist zu erwähnen, dass Auftrag C mit mehreren Hitzen gefertigt wird, wodurch es erforderlich ist, C_1 und C_2 immer hintereinander zu reihen. Dies ist auch in Abbildung 3 und 4 ersichtlich.

Auftrag	Rüstzeit[min]	Stückzeit[sek]	Anzahl[Stk.]	Hitzen
A	29	66	20	1
B	29	80	30	1
C	29	84	50	2

Tabelle 1: Beispielaufträge Pressparameter

Auftrag	Temp.[°C]	Ofen	min.Wärmezeit[min]	Haltezeit[min]
A	960	2	60	45
B	1038	1	60	45
C	940	2	60	45

Tabelle 2: Beispielaufträge Ofenparameter

[9] Unter Fertigung mit mehreren Hitzen versteht man das mehrmalige Wärmen und darauffolgendes Schmieden mit demselben Gesenk.
[10] minimale Wärmezeit = Durchwärmzeit + Haltezeit
[11] maximale Wärmezeit = Durchwärmzeit + Haltezeit + Toleranz(ca. 10%)

Um die Auswirkung der Reihenfolge der Fertigungsaufträge darzustellen ist der Produktionsablauf mit den entstehenden Wartezeiten an der Presse für 2 der 6 möglichen Reihungen in Abbildung 3 und Abbildung 4 abgebildet.
Deutlich zu sehen ist, dass die Auftragsreihenfolge BCA zu einer wesentlich längeren Hitzewartezeit führt, da es beim Übergang des zweiten Vorgangs von Auftrag C (C_2) zu Auftrag A (A_1) eine Temperaturdifferenz von 20°C zu überwinden gilt. Das Schmiedegut kann trotzdem vorab eingelegt werden und muss nach Erreichen der Prozesstemperatur von 960°C (nach 20 min) nur noch die im Wert Haltezeit angegebene Dauer (45 min) im Ofen verbleiben.
Die Reihenfolge CBA sieht hingegen vor, dass während der notwendigen Temperaturänderung in Ofen 2 auf Ofen 1 gewechselt wird und somit keine weitere Wartezeit entsteht.

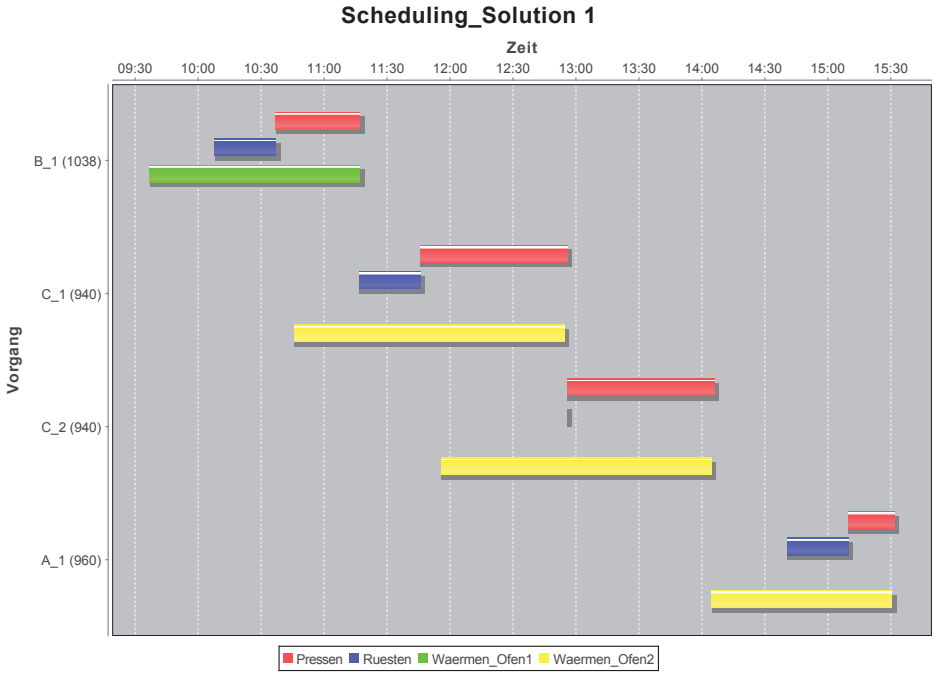


Abbildung 3: Reihung BCA, Hitzewartezeit 00:20:00

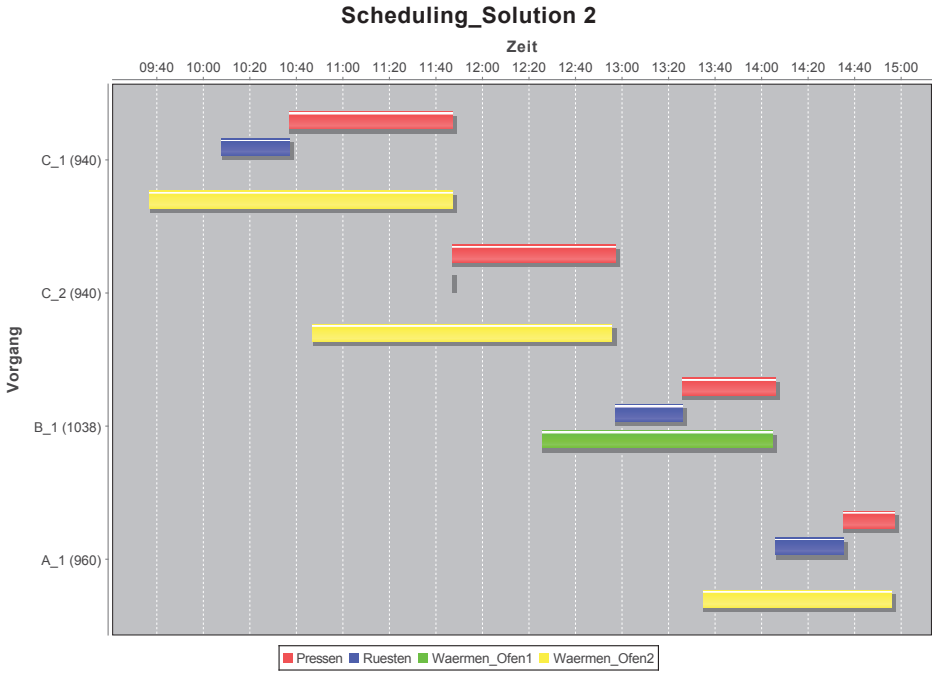


Abbildung 4: Reihung CBA, Hitzewartezeit 00:00:00

Als erste Idee würde sich anbieten, alle möglichen Varianten, die sich aus der Permutation der Aufträge ergeben zu berechnen und die beste Reihenfolge zu wählen. Ziel ist es jedoch einen Arbeitsvorrat von bis zu 14 Tagen zu reihen, was in der Regel 100 Aufträgen und mehr entspricht. Dies würde 100! unterschiedliche Varianten ergeben, was aufgrund deutlich zu langer Rechenzeit nicht ausgewertet werden kann.
Aus dieser Problematik heraus ergibt sich die Notwendigkeit Algorithmen zu entwickeln, die in annehmbarer Laufzeit möglichst gute Auftragsreihungen erzielen.

2.3 Scheduling der Presse vorgelagerter Aggregate

Beim Scheduling der Pressen wird davon ausgegangen, dass die der Presse vorgelagerten Aggregate das Pressgut in der gewünschten Reihenfolge fertigstellen. Die durch das Pressenscheduling berechneten Termine, zu denen das

Pressgut vor den Öfen angelangt sein muss, sind somit unbedingt einzuhalten. Die Verfügbarkeitsprüfung der Maschinenkapazitäten[12] hat dabei im Rahmen der Grobplanung bereits stattgefunden, wodurch davon auszugehen ist, dass es auch tatsächlich möglich ist, alle Termine einzuhalten[13]
Es ist jedoch möglich, dass die vom Pressenscheduling vorgegebene Reihenfolge ungünstig für den Sägebereich ist und zu längeren Wartezeiten oder vermehrten Rüstzeit führt. Die Herausforderung liegt darin, Algorithmen bzw. Heuristiken zur Reihenfolgeplanung einzusetzen, die das Pressenprogramm, unter optimaler Auslastung der Maschinen im Sägebereich, nicht gefährden.
Weiters ist zu beachten, dass aufgrund von Maschinenausfällen Kapazitätsengpässe an den Maschinen im Sägebereich möglich sind und der Terminplan der Pressen somit nicht mehr eingehalten werden kann. Treten solche Probleme auf, müssen die Algorithmen möglichst intelligent darauf reagieren.

2.4 Scheduling der Presse nachgelagerter Aggregate

Haben Produkte den Pressenbereich passiert, liegt das Hauptaugenmerk darauf, diese rechtzeitig zum Liefertermin fertig zu stellen. Wie die bereits in Kapitel 2.3 beschriebene möglicherweise ungünstige Auftragsreihenfolge im Sägebereich, kann die Reihenfolge des Pressenschedulings auch im Endfertigungsbereich problematisch sein.
Eine eigene Reihenfolgeplanung ist daher notwendig, um die Fertigstellung der Produkte unter optimalen Maschineneinsatz und unter Einhaltung vorgegebener logistischer Zielgrößen sicherzustellen. Herauszufinden gilt es, welche Algorithmen, bzw. Heuristiken unter welchen Bedingungen die besten Ergebnisse liefern. Wie auch im Sägebereich kann es auch im Endfertigungsbereich zu Kapazitätsausfällen kommen, worauf ebenfalls möglichst zielgerichtet reagiert werden muss.

[12] Im Rahmen der Verfügbarkeitsprüfung wird überprüft, ob die durch den zu produzierenden Produktmix entstehenden Kapazitätsbedarfe, durch die bestehenden Maschinenkapazitäten abgedeckt werden können.
[13] Dass dies nicht immer der Fall ist wird in Kapitel 2.4 dargelegt.

Um die Problematik zu konkretisieren folgendes Beispiel: Es gibt 3 Aufträge (A,B,C), die nach dem Pressen in den Endfertigungsbereich gelangen. Der Endfertigungsbereich besteht aus 3 Maschinen (M1, M2, M3). Die zugehörigen Arbeitspläne und Stückzahlen sind in Tabelle 3 ersichtlich.

Auftrag	M1[min]	M2[min]	M3[min]	Anzahl[Stk.]
A	-	10	5	20
B	3	15	-	30
C	5	-	7	50

Tabelle 3: Beispielaufträge Arbeitspläne und Stückzahlen

Vergleicht man nun die Gesamtdauer der Auftragsreihenfolge ABC und CBA ergeben sich deutlich unterschiedliche Werte. Ersichtlich ist dies in Tabelle 4 und Tabelle 5. Die Zahlen in den Klammern geben dabei die Wartezeit vor den Maschinen M1, M2, M3 an, die dadurch entsteht, dass der vorherige Auftrag noch nicht fertig bearbeitet ist. Dabei ist zu erwähnen, dass immer das gesamte Produktionslos fertig produziert werden muss, bevor einzelne Produkte zur nächsten Maschine weitertransportiert werden dürfen. Diese Einschränkung resultiert daraus, dass es ansonsten zur Vermischung unterschiedlicher Aufträge kommen könnte. Diese Einschränkung wird zwar in der Praxis nicht immer eingehalten, wird aufgrund notwendiger Komplexitätsreduktionen im Rahmen dieser Arbeit aber angenommen.

Auftrag	M1[min]	M2[min]	M3[min]	Σ[min]
A	-	200	100	300
B	90	450(110)	-	650
C	250(90)	-	350	690

Tabelle 4: Maschinenbelegung Auftragsreihenfolge ABC

Auftrag	M1[min]	M2[min]	M3[min]	Σ[Stk.]
C	250	-	350(50)	650
B	90(250)	450	-	790
A	-	200	100	300

Tabelle 5: Maschinenbelegung Auftragsreihenfolge CBA

Somit wird deutlich, welchen Einfluss die Reihenfolgeplanung auf die Liefertermine hat, und dass trotz einer positiven Verfügbarkeitsprüfung nicht unbedingt davon auszugehen ist, dass Termine ohne weiteres eingehalten werden können.

2.5 Auswahl representativer Daten

Um nicht nur theoretische, sondern auch für die BSTG relevante Ergebnisse zu erzielen, ist es notwendig, die entwickelten Algorithmen und Heuristiken auf Basis realer Daten anzuwenden. Die Fülle an Daten, die dazu jedoch notwendig ist um dies zu realisieren, ist im Rahmen dieser Arbeit nicht bearbeitbar, da die notwendige Datenqualität teilweise nicht vorhanden ist und keine Schnittstelle zum ERP-System existiert, um die erforderlichen Daten komfortabel zu extrahieren.
Um auch das im Rahmen der Arbeit simulierte System nicht unnötig komplex werden zu lassen, wird nur ein Teil der Fertigung, unter Berücksichtigung möglichst repräsentativer Daten, abgebildet. Die Auswahl dieser Daten betrifft vor allem Arbeitspläne, von denen in der Realität mehrere tausend bestehen und alle gekürzt, bzw. vereinfacht werden müssten, da nicht alle Maschinen, die für die Durchführung der in den Arbeitsplänen enthaltenen Vorgänge notwendig wären, im simulierten System abgebildet sein werden. Sinn macht es daher nur, in der Simulation mehrere dutzend Arbeitspläne anzulegen, sodass möglichst reale Produktionsprogramme von mehreren Wochen dargestellt werden können.

3 Lösungsansätze

Die in Kapitel 2 beschriebenen Problemstellungen des Schedulings der Presse, sowie der restlichen Aggregate der Fertigung, wird umso komplexer, umso mehr Maschinen betrachtet werden. Eine isolierte Betrachtung der einzelnen Maschinen und der Berechnung der Auswirkungen unterschiedlicher Auftragsreihenfolgen ist zwar möglich, jedoch fehlen dabei Aspekte wie gegenseitige Abhängigkeiten und stochastische Einflüsse. Gerade aber die Auswirkungen der Auftragsreihenfolge von beispielsweise Maschine A auf Maschine B und zufällige Maschinenausfälle, sind essentiell für die Evaluierung unterschiedlicher Auftragsreihenfolgen. Eine Betrachtung ist daher nur im Maschinenverbund sinnvoll, wodurch der Einsatz eines Simulationsprogramms, in Verbindung mit unterschiedlichen Scheduling Algorithmen, nahezu unumgänglich wird.

3.1 Simulationsprogramm

Ziele des Einsatzes eines Simulationsprogramms sind mehrere Aspekte. Es soll verstanden werden, wie der komplexe Maschinenverbund funktioniert, speziell welche Gründe zu immer wieder auftretenden wechselnden Engpässen führen, weiters soll das bestehende Feinplanungssystem durch den Einsatz von Scheduling Algorithmen verbessert werden, was jedoch nur möglich ist, wenn diese vorab in der Simulation getestet werden können. Ebenfalls besteht die Möglichkeit, alternative Kapazitätskonfigurationen zu testen, womit neben einer Verbesserung der Personalplanung, auch Investitionsentscheidungen getroffen bzw. abgesichert werden können. [Chung 2004, Seite 1-3f]

3.1.1 Allgemeines über Simulationsprogramme

Prinzipiell gibt es sehr viele Möglichkeiten Simulationsmodelle zu klassifizieren, wobei folgende drei Dimensionen häufig verwendet werden [Kelton u. a. 2004, Seite 7].

- **statisch vs. dynamisch**
  Bei statischen Simulationen spielt die Zeit keine Rolle, wohingegen dynamische Simulationen den Zeitaspekt berücksichtigen. Ein Beispiel für eine statische Simulation wäre ein Versuch, das Traveling Salesman Problem durch das zufällige Erzeugen mehrerer Routen zu lösen. Dynamisch hingegen wäre die Simulation des Materialbestandes eines Lager im Laufe eines Jahres.

- **kontinuierlich vs. diskret**
  In einem kontinuierlichen Model kann sich der Systemstatus kontinuierlich über die Zeit verändern, wohingegen bei einem diskreten Modell sich der Systemstatus nur zu bestimmten Zeitpunkten ändern kann. Die Simulation wird somit schrittweise durchgeführt. Kommen in einer Simulation sowohl kontinuierliche als auch diskrete Systemstatusänderung vor, spricht man von hybrider Simulation. Kontinuierlich wäre die Simulation des Füllstandes eines Benzintanks im Laufe einer Autofahrt, wohingegen die Simulation einer Maschine, die beispielsweise alle 3 Minuten ein Werkstück fertig stellt, diskret wäre.

- **deterministisch vs. stochastisch**
  Modelle die keinerlei Zufallseinflüsse enthalten, sind deterministisch, Modelle mit Zufallseinflüssen stochastisch. Stochastizität würde vorliegen, wenn die bereits genannte Maschine zufällig ausfallen würde, wohingegen fixe Wartungsintervalle deterministisch wären.

Die Vorteile einer Simulation liegen ganz klar darin, dass Experimente in sehr kurzer Zeit durchgeführt werden können, Systeme mit weniger analytischem Aufwand zu analysieren sind und die Funktionsweise schnell und relativ einfach demonstriert werden kann. Die Nachteile hingegen sind, dass meist Vereinfachungen notwendig sind, um die Realität in einem Simulationsmodell abbilden zu können, und die Ergebnisse daher Abweichungen zur tatsächlichen Situation

aufweisen können. Aus diesem Grund ist auch die Modellbildung am Anfang jeder Simulation von großer Bedeutung, da die Qualität des Modells wesentlichen Einfluss auf die Qualität des Endergebnisses hat.
Weiters ist es nicht damit getan, ein Problem zu simulieren um auch die Lösung eines Problems zu finden - eine Simulation liefert nicht die Lösung eines Problems, sondern erleichtert die Analyse [Chung 2004, Seite 1-6].
Bei dem im Rahmen dieser Arbeit entwickelten Simulationsprogramm handelt es sich um ein dynamisch - diskret - stochastisches Simulationsmodell. Der Aufbau und die Funktionsweise des Programms mit der Beschreibung der wichtigsten Klassen wird in den nächsten Kapiteln erläutert.

3.1.2 Events

Der zeitliche Ablauf der Simulation wird mittels Events realisiert. Ein Event enthält als Attribute die Zeit und eine Entität der Simulation, an der das Event auftreten wird. Im Falle der im Rahmen dieser Arbeit implementierten Simulation treten Events immer an Maschinen auf und definieren die Änderungen, welche die jeweilige Maschine zum Zeitpunkt des Auftretens des Events erfährt. Weiters besteht noch die Möglichkeit, Zusatzinformationen zu speichern, die bei der Ausführung des Events benötigt werden. Die abstrakte Oberklasse `Event` implementiert die Attribute `time` und `ort`. Sämtliche Events erben von dieser Klasse und implementieren eventuell weitere benötigte Attribute. Das Klassendiagramm dazu ist in Abbildung 5 zu sehen.

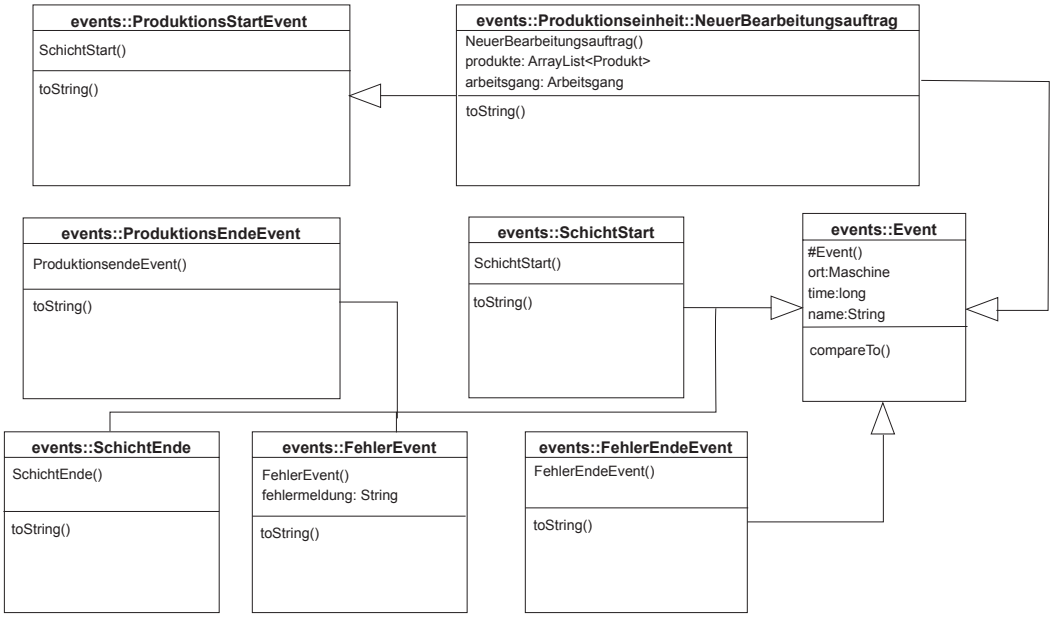


Abbildung 5: Klassendiagramm Events

Die dargestellten Events sind nur ein kleiner Teil aller in der Simulation verwendeten Events, jedoch sind diese wesentlich, da von diesen Events wiederum andere Events erben.

3.1.3 Eventhandler

Der Eventhandler ist die zentrale Klasse in der Simulation. Im Eventhandler sind alle Events aufsteigend nach Zeit in einer PriorityQueue gespeichert. Eine PriorityQueue speichert Objekte und sortiert diese in einer festlegbaren Reihenfolge. Es ist nicht sichergestellt, dass alle Elemente in der Priority Queue in der richtigen Reihenfolge sind, sonder nur, dass das kleinste Element an oberster Stelle steht.
Jedes mal wenn ein neues Event erzeugt wird, wird dieses an den Eventhandler geschickt und in die Eventliste einsortiert. Um nun einen Schritt in der Simulation weiterzuspringen, wird das erste Event der Liste an die im Attribut `ort` gespeicherte Maschine gesendet, die dann wiederum bestimmte Aktionen setzt und eventuell weitere Events generiert. Dass Klassendiagramm des Eventhandlers mit der zugehörigen Eventliste, ist in Abbildung 6 zu sehen.

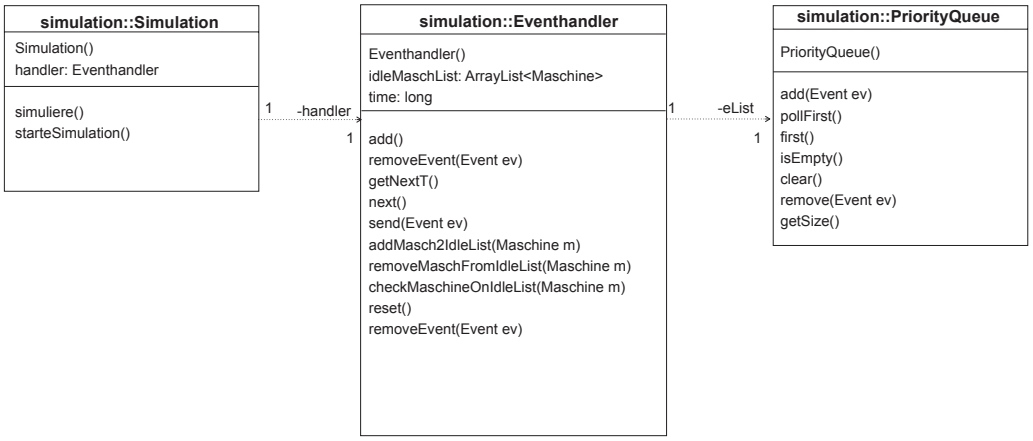


Abbildung 6: Klassendiagramm Eventhandler

Die Klasse Simulation wurde in der Darstellung um die meisten Attribute und Methoden reduziert, da nur die Assoziation zwischen `Simulation`, `Eventhandler` und `PriorityQueue` dargestellt werden sollte.
Wie der Programmablauf bei einem Simulationsschritt im Detail abläuft, ist im Sequenzdiagramm in Abbildung 7 zu sehen.

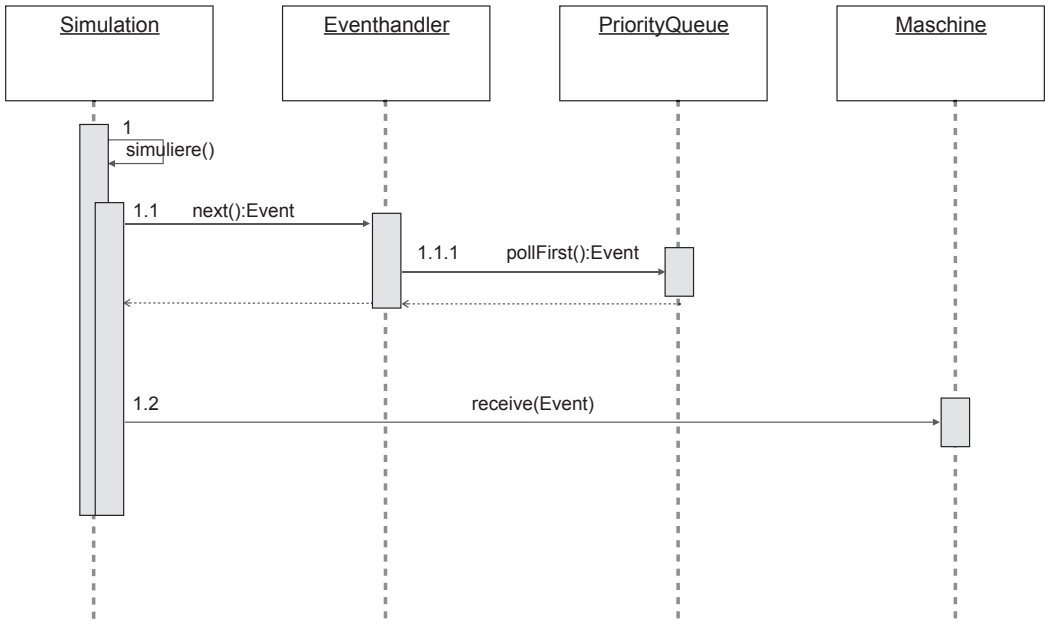


Abbildung 7: Sequenzdiagramm Simulationsschritt

Die Methode `simuliere()` in der Klasse `Simulation` ruft die Methode `next()` des `Eventhandlers` auf, die wiederum die Methode `pollFirst()` der Klasse `PriorityQueue` aufruft. Das erste Event wird damit aus der `PriorityQueue` „geworfen“ und an die `Simulation` zurückgegeben. Noch in der Methode `simuliere()` wird die Methode `receive(Event)` der Klasse `Maschine` aufgerufen und das Event damit an die „Zieldestination“ geschickt, wo es anschließend verarbeitet wird.

3.1.4 Maschine

Wie bereits erwähnt, werden Maschinen mittels Events gesteuert. Beim Erhalt eines Events wird geprüft um welchen Eventtyp es sich handelt, und dann an die jeweilige Eventverarbeitungsmethode weitergegeben. Sämtliche Grundfunktionalitäten, die von allen Maschinen benötigt werden, sind in der abstrakten Oberklasse `Maschine` implementiert, Zusatzfunktionen in den jeweiligen abgeleiteten Klassen. Diese Funktionen sind beispielsweise Bestandsbuchungen, Fehlerhandling, Maschinenverfügbarkeit und die Berechnung von Maschinenkennzahlen. Das Klassendiagramm der Maschine inklusive der davon abgeleiteten Klassen und der verwendeten Klasse `ProductQueue` ist in Abbildung 8 zu sehen. Zu sehen sind auch die Assoziationen zwischen der Klasse `Lager` und den anderen abgeleiteten Klassen, sowie die Assoziationen mit der Klasse `ProductQueue`.

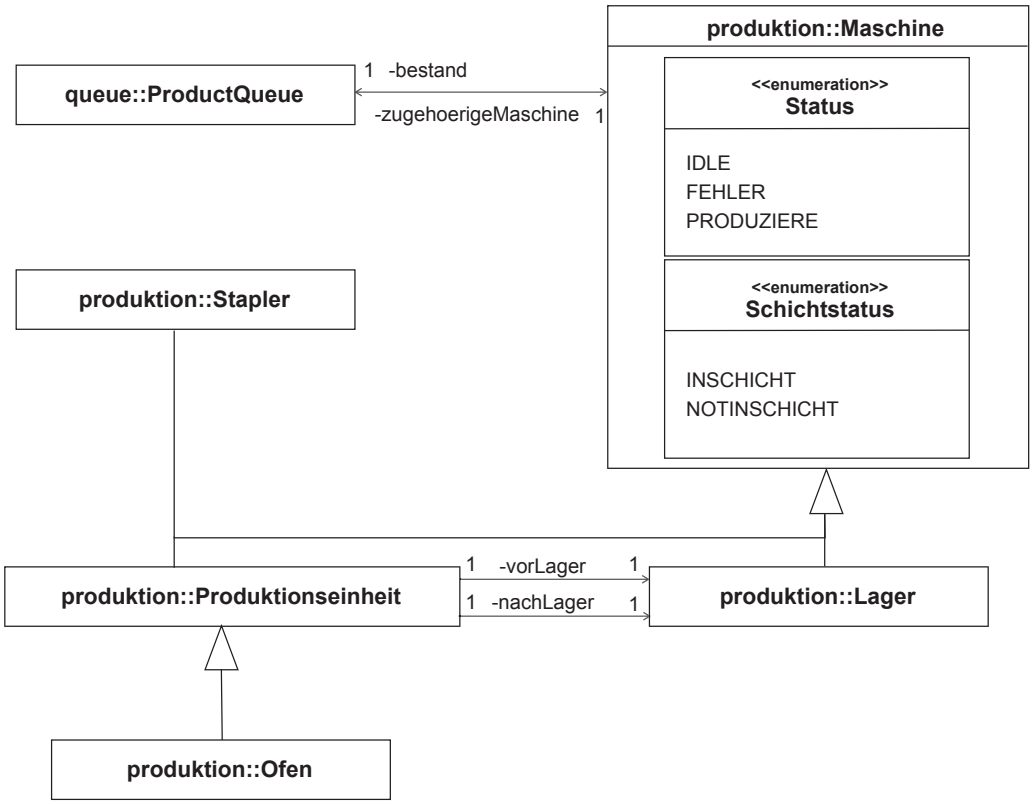


Abbildung 8: Klassendiagramm Maschine

Im Diagramm wurde auf die Darstellung der Attribute und Methoden bewusst verzichtet, da nur Vererbungshierarchie und Assoziationen dargestellt werden sollten.

3.1.5 ProductQueue

Unter einer ProductQueue wird im Rahmen dieser Arbeit eine Liste von Produkten verstanden, die dazu verwendet wird, Warteschlangen abzubilden. Wie in Abbildung 8 ersichtlich ist, wird der Bestand an Werkstücken, die sich in der Maschine befinden, durch eine ProductQueue implementiert. Somit ist es möglich, sowohl Materialbestand bzw. ein Warteschlangensystem in einer Maschine, sowie sämtlich Arten von Lagern zu modellieren. Ein Beispiel dafür ist die Klasse `Produktionseinheit`, welche ein Attribut vom Typ `ProductQueue` als Bestand verwendet und die Attribute `Vorlager` und `Nachlager` beinhaltet,

die wiederum ProductQueues als Bestand verwenden.
Weiters werden in den ProductQueues jeweils Mittelwert, Standardabweichung, Maximum und Minimum der Verweildauer und Produktanzahl berechnet, wodurch ein Fertigungssystem sehr gut analysiert werden kann. Immer wenn ein Produkt der ProductQueue hinzugefügt, oder der ProductQueue entnommen wird, werden die Kennzahlen neu berechnet,wodurch bereits während der Simulation wechselnde Engpässe identifiziert werden können. Die Implementierung ist in Abbildung 9 und Abbildung 10 dargestellt. Getter- und Setter-Methoden sind bewusst nicht im Klassendiagramm abgebildet.

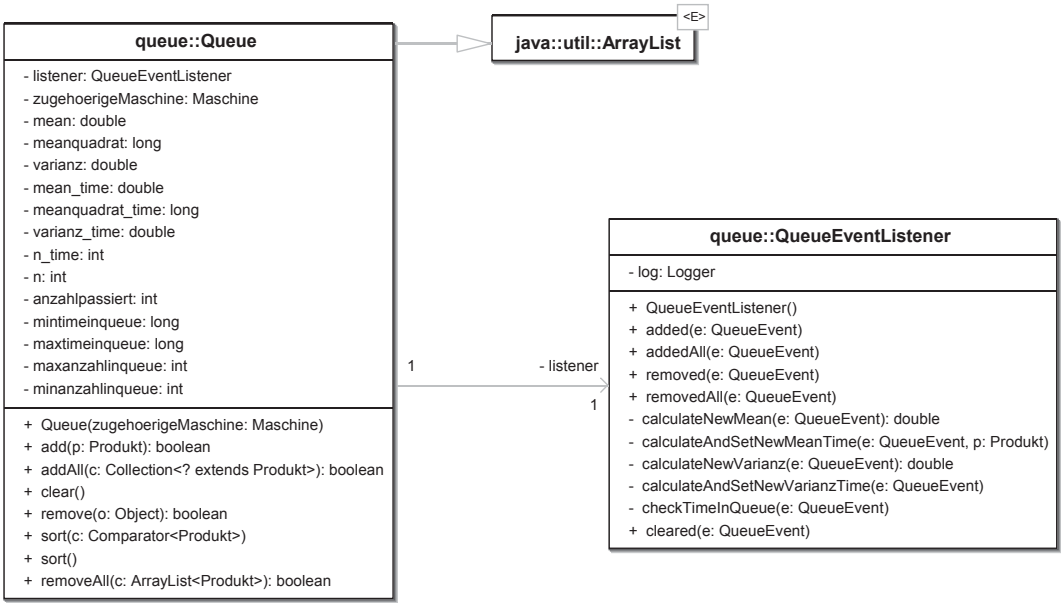


Abbildung 9: Klassendiagramm ProductQueue

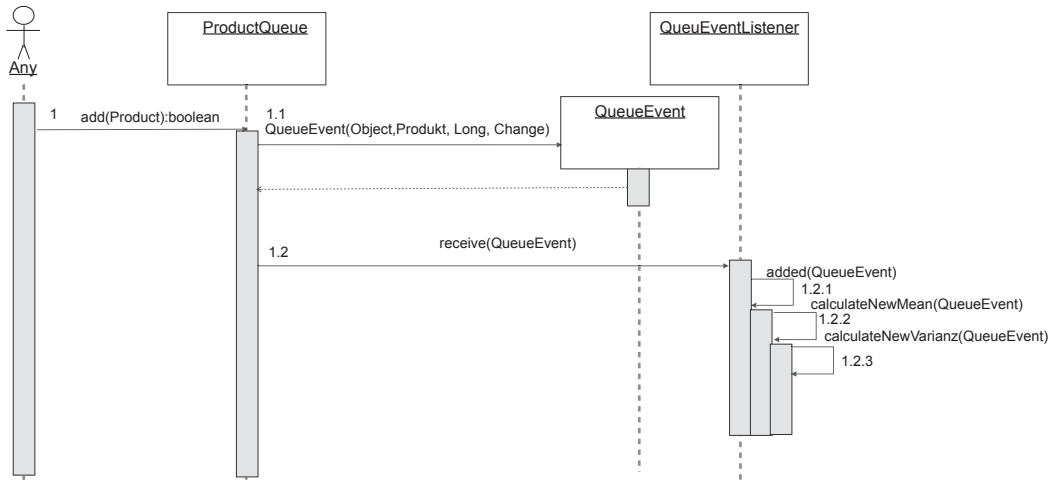


Abbildung 10: Sequenzdiagramm ProductQueue

Im Sequenzdiagramm in Abbildung 10 ist zu sehen, dass bei jedem Aufruf der Methode `add(Produkt p)`, eine neue Instanz der Klasse QueueEvent erzeugt wird. Dieses QueueEvent wird mittels der Methode `receive(QueueEvent e)` an den `QueueEventListener` übergeben, wo im Anschluss überprüft wird, was sich in der `ProductQueue` geändert hat und in diesem Fall daraufhin die Methode `added(QueueEvent e)` aufgerufen wird, welche Mittelwert und Varianz der Queue-Länge neu berechnet.
Die Entscheidung den Umweg über ein QueueEvent zu gehen scheint auf den ersten Weg unnötig, da diese Art von Events nicht mit den Events zur Simulationssteuerung in Verbindung stehen. Der Vorteil liegt darin, dass die Klasse `ProductQueue` nichts mit der Neuberechnung der Kennzahlen zu tun hat, sondern lediglich ein QueueEvent an die Methode `receive(QueueEvent e)` des QueueEventListeners sendet und dieser differenziert, was sich geändert hat und aufgrund dessen Neuberechnungen durchführt. Somit sind Aktion und Reaktion streng voneinander getrennt und beide Klassen können einfach verändert oder ersetzt werden, ohne sich gegenseitig zu beeinflussen. Weiters besteht der große Vorteil darin, dass es bei der Implementierung einer grafischen Oberfläche klar definierte Systemänderungen gibt, auf die z.B. mit einer Aktualisierung der Oberfläche reagiert werden kann.

3.1.6 Einlastungsstrategie - Simulationssteuerung

Die Steuerung der Simulation erfolgt über eine sogenannte Einlastungsstrategie, welche den aktuellen Status der Simulation analysiert und daraufhin Events erzeugt und an den Eventhandler sendet. Die Analyse erfolgt nach folgendem Prinzip:

- Überprüfung aller Maschinen, ob in deren Nachlager Produkte enthalten sind, die weitertransportiert werden können. Ein Produkt darf nur weitertransportiert werden, wenn alle Produkte des Produktionloses im Nachlager vorhanden sind. Essentiell dabei ist sicherzustellen, dass ein Produkt, sobald es aufgrund eines Maschinenausfalls zerstört wird, auch aus dem Produktionslos entfernt wird.
- Auswahl der Maschinen, zu denen die Produkte transportiert werden sollen. Stehen mehrere Maschine zur Auswahl, wird jene Maschine mit der kürzesten Warteschlange gewählt. Die Warteschlangenlänge ist dabei über die Summe aller Bearbeitungszeiten der in der Warteschlange befindlichen Produkte definiert.
- Zuteilung der zu transportierenden Produkte zu freien Staplern. Da die Möglichkeit besteht, dass mehr Produkte transportiert werden sollen, als Stapler zur Verfügung stehen, erfolgt eine Reihung mittels festlegbarer Transportpriorität. Die Produkte des wichtigsten Auftrages erhalten somit den freien Stapler, der am nächsten ist usw. Sollte die Anzahl der Produkte des wichtigsten Auftrages die Kapazität des Staplers überschreiten, fährt der Stapler mehrmals.
- Überprüfung aller Maschinen, ob in deren Vorlagern Produkte enthalten sind, die bearbeitet werden können. Die Reihung, welche Produkte als nächste produziert werden sollen, erfolgt über einen individuell in jeder Maschine einstellbaren Reihungsalgorithmus. Ein Produkt darf nur bearbeitet werden, wenn alle Produkte des Produktionsloses im Vorlager enthalten sind. Übersteigt die Produktanzahl des Produktionsloses die Kapazität der Maschine, wird ein Folgeauftrag mit den restlichen Produkten generiert.

Der Programmablauf dazu, ist im Sequenzdiagramm in Abbildung 11 zu sehen.

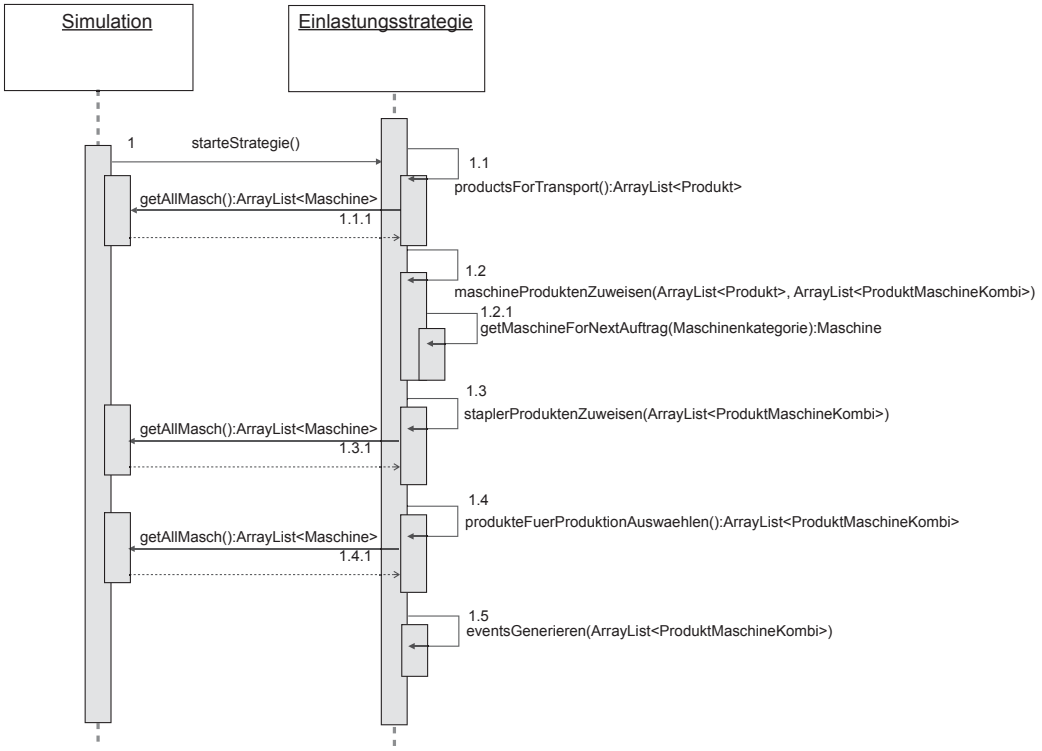


Abbildung 11: Sequenzdiagramm Einlastungsstrategie

3.2 Scheduling Algorithmen

Wie bereits in Kapitel 3.1.1 erwähnt, helfen Simulationsprogramme dabei, ein Problem besser darzustellen und besser zu analysieren, jedoch lösen sie es nicht. Im Fall der Reihenfolgeplanung von unterschiedlichen Maschinen mit dem Hauptaugenmerk auf ein Engpassaggregat, handelt es sich um ein sehr komplexes Problem, welches in den folgenden Kapiteln eingehend betrachtet und differenziert wird.

3.2.1 Allgemeines über Scheduling Algorithmen

Unter dem Begriff „Scheduling“ versteht man die Erstellung eines Plans, der im Grunde beinhaltet, welches Produkt wann auf welcher Maschine produziert werden soll. Im Rahmen dieser Arbeit wird das detaillierte Scheduling von Maschinen bzw. Arbeitsstationen betrachtet, welches genau festlegt, wann ein Arbeitsgang begonnen werden und auch wieder fertiggestellt werden muss. Diese Art des Scheduling wird im weiteren Verlauf auch Feinplanung oder Auftragsreihenfolgeplanung genannt.
Ein Scheduling Problem ist grundsätzlich wie folgt defniert [Brucker 2006]:
Man nehme an, dass m Maschinen $M_j (j = 1, ..., m)$ n Aufträge ausführen sollen. Ein „Schedule“ oder auch „Zeitplan“ ist eine Zuteilung eines Auftrags zu einem Zeitintervall einer bestimmten Maschine. Ein solcher Zeitplan kann als Gannt Chart dargestellt werden, wie dies bereits in Abbidlung 3 und 4 der Fall war.
Ein Auftrag (Job) J_i besteht auf einer Anzahl von n Vorgängen (Operation) $O_{i,1}, ..., O_{i,n}$. Jedem Vorgang O_{ij} ist eine gewisse Arbeitsleistung (processing requirements) p_i zugewiesen. Gemeint ist damit, dass daraus, je nach Leistung der Maschine auf der bearbeitet wird, eine Bearbeitungszeit berechnet werden kann. Ein Freigabezeitpunkt r_i kann angegeben sein, an dem der erste Vorgang von J_i fertig für die Bearbeitung ist. Jedem Vorgang O_i ist eine Menge von Maschinen $\mu_{ij} \subseteq \{M_1, ..., M_m\}$ zugewiesen, an denen O_i bearbeitet werden kann. Alle Maschinen der Menge μ_{ij} können von unterschiedlichem oder auch vom gleichen Typ sein.
Weiters gibt es noch eine Kostenfunktion $f_i(t)$, welche die Kosten für die Fertistellung von J_i zum Zeitpunkt t angibt. Ein Ferstigstellungsdatum (due date) d_i und ein Gewichtungsfaktor (weight) w_i, können bei der Definition der Kostenfunktion f_i verwendet werden.

3.2.2 Notation von Scheduling Problemen

Dieses Kapitel ist in großen Teilen [Graham u. a. 1976] und [Brucker 2006] entnommen, da diese beiden Werke eine sehr kompakte und übersichtliche Einführung in die Definition und Notation von Scheduling Problemen bietet. In

diesem Kapitel soll vor allem ein Überblick über die Notation gegeben werden, eine genaue Kenntnis dieser ist aber für das weitere Verständnis dieser Arbeit nicht notwendig.
Es gibt sehr viele unterschiedliche Arten von Scheduling Problemen. In der Literatur werden diese mittels einer drei-Feld $\alpha|\beta|\gamma$ Klassifizierung unterschieden, wobei α die Maschinenpark-, β die Auftrags- und γ die Optimierungskriterien spezifiziert.

3.2.2.1 Maschinencharakteristik

Die Maschinenparkkriterien werden mittels eines Strings $\alpha = \alpha_1\alpha_2$ von zwei Parametern festgelegt. Der Parameter α_1 gibt die Beschaffenheit des Maschinenparks an, der Parameter α_2 gibt an, wieviele Maschinen zur Verfügung stehen. Wird der Parameter α_2 mit *null* angegeben, so ist die Anzahl der Maschinen variabel.
Eine Übersicht über die möglichen Werte, die α_1 annehmen kann und eine kurze Erklärung dazu, ist in Tabelle 6 angegeben.

α_1	**Beschreibung**
null	jeder Auftrag muss auf einer bestimmten Maschine gefertigt werden
P	idente parallele Maschinen (Maschinen haben gleiche Bearbeitungsgeschwindigkeit)
Q	gleiche parallele Maschinen (Maschinen haben unterschiedliche Bearbeitungsgeschwindigkeit)
R	verschiedene parallele Maschinen
$PMPM$	Mehrzweckmaschinen mit identer Bearbeitungszeit
$QMPM$	Mehrzweckmaschinen mid auftragsabhängiger Bearbeitungszeit
G	General Shop Problem
F	Flow Shop Problem
J	Job Shop Problem
O	Open Shop Problem
X	Mixed Shop Problem

Tabelle 6: Mögliche Werte für Parameter α_1

Job shop, flow shop, open shop und mixed shop Probleme sind alles spezielle Fälle des general shop Problems und sind wie folgt zu unterschieden [Zetschke 2003].
Bei einem flow shop Problem gibt es n Aufträge und m Maschinen. Jeder Auftrag $(J_1, ..., J_n)$ besteht aus m Arbeitsgängen $(O_1, ..., O_m)$, welche die Maschinen jeweils in der gleichen Reihenfolge durchlaufen müssen. $O_{i1} \rightarrow O_{i2} \rightarrow ... \rightarrow O_{ik}$ für $i = 1, ...n$
Ein job shop Problem ist dadurch charakterisiert, dass es n Aufträge $(J_1, ...J_n)$ gibt, die jeweils aus m Arbeitsgängen bestehen $(O_1, ..., O_m)$, wovon jeder in der in J_i festgelegten Reihenfolge, auf einer der m Maschinen $(M_1, ..., m)$ bearbeitet werden muss, wobei die Reihenfolge jedoch von Auftrag zu Auftrag variieren kann. Das open shop problem ist das allgemeinste Problem. Es gibt wiederum n Aufträge $(J_1, ..., J_n)$ mit jeweils m Vorgängen $(O_1, ..., O_m)$, die jeweils auf m Maschinen $(M_1, ..., M_m)$ produziert werden müssen, jedoch gibt es keinerlei Einschränkungen in der Maschinen oder Vorgangsreihenfolge.

Um die Unterschiede deutlich zu machen, sind job shop und flow shop Fertigungen in den beiden folgenden Abbildungen schematisch dargestellt.

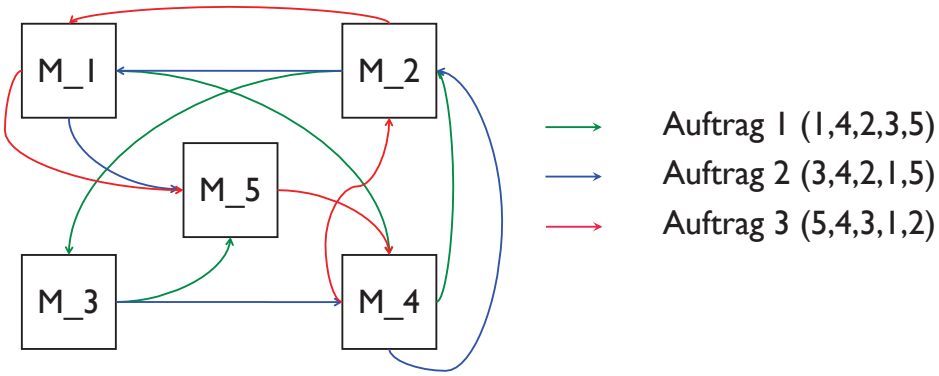


Abbildung 12: Job Shop Fertigung

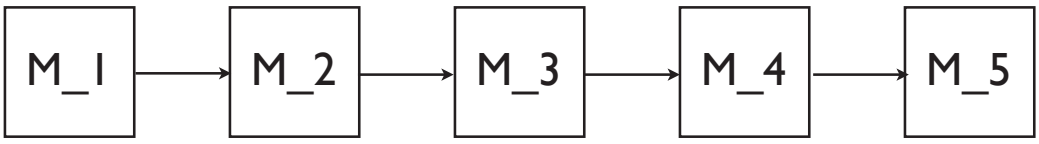


Abbildung 13: Flow Shop Fertigung

Ein mixed shop Problem ist nichts anderes als eine Mischung aus open shop und job shop.

3.2.2.2 Auftragscharakteristik

Die Charakteristik eines Auftrags wird mit den Parametern $\beta_1, ..., \beta_6$ spezifiziert. Tabelle 7 gibt eine Übersicht über die einzelnen Parameter und ihre Bedeutung.

β_1	**Splitten von Aufträgen**
pmtn	Aufträge zu splitten oder zu unterbrechen ist erlaubt
null	Aufträge zu splitten oder zu unterbrechen ist nicht erlaubt
β_2	**Reihefolgebeziehungen (RB) zwischen den Aufträgen**
prec	RB enstpricht azyklisch gerichteten Graphen
intree	RB entspricht einem Intree
outtree	RB entspricht einem Outtree
tree	RB enspricht entweder *intree* oder *outtree*
chains	RB enhält mehrere Chains
null	Es gibt keine Reihenfolgebeziehungen
β_3	**Freigabedatum**
Date	Es gibt ein Freigabedatum *Date*
β_4	**Einschränkungen in Bearbeitungszeit oder Vorgangsanzahl**
z.B. $p_i = 1$	Einheitliche Bearbeitungszeit für alle Aufträge/Vorgänge
null	Keine weiteren Einschränkungen
β_5	**Deadline**
d_i	Aufträge müssen vor d_i fertiggestellt sein
null	Keine Deadline vorhanden
β_5	**Aufträge können zu Batches zusammengefasst werden**
$s - batch$	Länge des Batch ist gleich lang wie dies Summe der Bearbeitungszeiten aller Aufträge innerhalb des Batch
$p - batch$	Länge des Batch ist gleich lang wie das Maximum der Bearbeitungszeiten aller Aufträge innerhalb des Batch
null	Batching ist nicht erlaubt

Tabelle 7: Mögliche Werte für Parameter β_{1-6}

3.2.2.3 Optimierungskriterien

Wurden in Kapitel 3.2.2.1 und 3.2.2.2 bisher nur die Notation für die Rahmenbedingungen definiert, werden nun die unterschiedlichen Zielfunktionen dargestellt. Es gibt im Wesentlichen zwei Arten von Kostenfunktionen:

$$f_{max}(C) := max\{f_i(C_i) \mid i = 1, ..., n\},$$

und

$$\sum f_i(C) := \sum_{i=1}^{n} f_i(C_i).$$

Dabei ist C_i die tatsächliche Fertigstellungszeit von Auftrag J_i und die dazugehörige Kostenfunktion ist $f_i(C_i)$. Die erste Kostenfunktion ist nichts anderes als das Maximum aller einzelnen Kostenfunktionen, was bedeutet, dass der Auftrag mit der längsten tatsächlichen Fertigstellungszeit die Kosten vorgibt und diese bei einer Optimierung zu minimieren wäre. Die zweite Kostenfunktion ist die Summe aller einzelnen Kostenfunktionen, womit bei einer Optimierung alle tatsächlichen Fertigstellungszeiten zu minimieren sind und ein einzelner Ausreißer sich weniger dramatisch auswirkt.
Weitere mögliche Zielfunktionen basieren auf dem geplanten Fertigstellungstermin d_i, welcher für jeden Auftrag J_i bekannt sein muss. Daraus können für jeden Auftrag folgende Zielfunktionen definiert werden:

$L_i := C_i - d_i$	Verspätung (Lateness)
$E_i := max\{0, d_i - C_i\}$	Frühezeitigkeit (Earlyness)
$T_i := max\{0, C_i - d_i\}$	Langsamkeit (Tardyness)
$D_i :=\mid C_i - d_i \mid$	absolute Abweichung
$S_i := (C_i - d_i)^2$	quadratische Abweichung
$U_i := \begin{cases} 0 \text{ if } C_i \leq d_i \\ 1 \text{ otherwise} \end{cases}$	unit penalty

Es bleibt noch zu erwähnen, dass mit jeder dieser Funktionen $G_i \in \{L_i, E_i, T_i, D_i, S_i, U_i\}$ vier unterschiedeliche Optimierungskriterien γ definiert werden können,

$$\gamma = \max G_i,\ \max w_i G_i,\ \sum G_i,\ \sum w_i G_i.$$

3.3 Beispiele für einfache Lösungsalgorithmen

In diesem Kapitel werden einige Lösungs- bzw. Optimierungsverfahren für eine Auswahl an bekannten Problemen dargestellt, die jedoch nicht auf die in Kapitel 2 beschriebene Problemstellung angewandt werden können, jedoch mit der Notation der Problemstellung und der Beschreibung der Lösungsalgorithmen vertraut machen sollen.

3.3.1 $1 \mid p_j = 1 \mid \sum \omega_i U_i$

Es steht eine Maschine zur Verfügung und eine gewissen Anzahl von Aufträgen ist zu produzieren, jedoch ist es innerhalb der zur Verfügung stehenden Zeit nicht möglich, alle Aufträge fertigzustellen. Eine Möglichkeit wäre, eine

Auftragsreihenfolge zu finden, welche die Summe der gewichteten verspäteten Aufträge minimiert. Eine sinnvolle Gewichtung wäre beispielsweise, die Verspätung des Auftrags eines Stammkunden stärker zu gewichten, als die eines normalen Auftrags. Um den nun vorgestellten Algorithmus verwenden zu können, muss angenommen werden, dass alle Aufträge eine einheitliche Bearbeitungszeit vorweisen ($\beta_5 \rightarrow p_j = 1$).
Mit Hilfe des folgenden Algorithmus kann eine optimale Auftragsreihenfolge generiert werden.[Brucker 2006, Seite 85]

Algorithm 1 $1 \mid p_j = 1 \mid \sum \omega_i U_i$

```
1: t:=1; S:=null;
2: for i := 1 to n do
3:    if d_i ≥ t then
4:       Add i zu S; t := t + 1
5:    else if there exists a job k ∈ S with w_k ≤ w_i then
6:       Delete job k form S where k ist the largest index such that the value
         w_k is minimal;
7:    end if
8: end for
```

Der Algorithmus macht nicht anderes, als sämtliche Aufträge der Reihe nach in die Auftragsreihenfolge S zu geben. Sollten dann beispielsweise ein Auftrag i nicht mehr hineinpassen, wird jener Auftrag k mit dem geringsten Gewicht aus S entfernt und stattdessen Auftrag i hinzugefügt.

3.3.2 $1 \mid \mid \sum U_j$

Wiederum handelt es sich um ein Problem mit einer Maschine, jedoch sind diesmal die Bearbeitungszeiten variabel, sowie gilt es, die Anzahl an zu späten Aufträgen zu minimieren. Das heißt, dass die maximale Anzahl an Aufträgen innerhalb der zur Verfügung stehenden Zeit produziert werden soll. Das optimale Ergebnis sieht so aus, dass die Aufträge, die innerhalb der zur Verfügung

stehenden Zeitspannen gefertigt werden können, aufsteigend nach Fertigstellungstermin sortiert sind, gefolgt von den restlichen Aufträgen in beliebiger Reihenfolge.
Mit folgendem Algorithmus kann eine solche Auftragsreihenfolge generiert werden. [Brucker 2006, Seite 86]

Algorithm 2 $1 \mid\mid \sum U_j$

1: Enumerate jobs such that $d_1 \leq d_2 \leq ... \leq d_3$
2: $S := null; t := 0;$
3: **for** $i := 1$ to n **do**
4: $\quad S := S \cup \{i\};$
5: $\quad t := t + p_i;$
6: $\quad$ **if** $t \geq d_i$ **then**
7: $\quad\quad$ Find job j in S with largest p_j value;
8: $\quad\quad S := S \setminus \{j\}$
9: $\quad\quad t := t_i + p_j$
10: $\quad$ **end if**
11: **end for**

Der Algorithmus macht nichts anderes, als die im Vorfeld nach Auftragsdatum aufsteigend sortierten Aufträge der Reihe nach S hinzuzufügen. Führt jedoch das Hinzufügen von Auftrag j dazu, dass dieser Auftrag innerhalb der zur Verfügung stehenden Zeitspanne nicht fertig gestellt werden kann, wird jener Auftrag aus S entfernt, der die größte Bearbeitungszeit aufweist.

4 Gewählte Lösungsansätze

In diesem Kapitel wird beschrieben, welche Lösungsansätze bzw. welche Algorithmen und Heuristiken zur Bearbeitung der Problemstellung gewählt wurden. Dabei wird die bereits in Kapitel 2 beschriebene Dreiteilung der Fertigung angewandt und somit drei eigenständige Scheduling Probleme betrachtet.

4.1 Scheduling Aggregate vor und nach der Presse

4.1.1 Scheduling Aggregate vor der Presse

Wie bereits in Kapitel 2.3 beschrieben, ist es die Aufgabe dieser Aggregate, die von der Presse vorgegebenen Termine für das Pressgut unbedingt einzuhalten. Es stehen dabei m Maschinen $(M_1, ..., M_m)$ zur Verfügung und es sind n Aufträge $(J_1, ..., J_n)$ zu bearbeiten. Die Arbeitsgänge J_{ij} (Arbeitsgang J des Auftrags i mit Vorgangsnummer j) müssen in der vorgegebenen Reihenfolge bearbeitet werden.
Konnte bisher angenommen werden, dass es sich dabei um ein jobshop Problem handelt, wird bei genauerer Betrachtung klar, dass die in Kapitel 3.2.2.1 beschriebenen Bedingungen nicht ganz erfüllt werden. So ist es möglich, dass $j \leq m$ und somit ein Auftrag nicht auf jeder Maschine bearbeitet werden muss. Weiters kommt noch dazu, dass es teilweise pro Maschinentyp mehrere Maschinen gibt, wodurch es zu einem resource constraint project scheduling problem kommt, die nur sehr schwer zu lösen sind [Graham u. a. 1976, Seite 311].
Mögliche Notationen von resource contraint project scheduling problems sind in [Brucker u. a. 1999] beschrieben, diese in vollem Umfang zu beschreiben ist

jedoch zu komplex für den Rahmen dieser Arbeit. Sie basieren jedoch auf der $\alpha \mid \beta \mid \gamma$ Notation und sind um einige Parameter erweitert. Es handelt sich im konkreten Fall um ein $PSm, 1, y_i \mid\mid \sum U_i$ Problem, was bedeutet, dass es sich im ein project scheduling Problem mit m Maschinen insgesamt handelt, wobei jeder Vorgang genau eine Maschine (1) benötigt und von diesem Maschinentyp y_i Maschinen zur Verfügung stehen. Reihenfolgevorgaben, in denen die Aufträge bearbeitet werden müssen gibt es keine, jedoch müssen die Vorgangsreihenfolgen $O_i, 1 \rightarrow O_i, 2, ..., \rightarrow O_i, n$ innerhalb der Aufträge eingehalten werden. Zu optimieren ist die Anzahl der verspäteten Aufträge. Eine Gewichtung wird dabei nicht vorgenommen, da jede Verspätung zur Folge hat, dass das Scheduling an der Presse nicht eingehalten werden kann und dieses somit neu berechnet werden muss.

4.1.2 Scheduling der Aggregate nach der Presse

Wie auch bereits in Kapitel 2.4 beschrieben, ist es die Aufgabe dieses Fertigungsbereiches die Produkte zum gewünschten Liefertermin fertigzustellen. Aufgrund beschränkter Ressourcen ist dies jedoch nicht immer möglich, wodurch die gewichteten zu späten Aufträge minimiert werden sollen. Ansonsten gelten die gleichen Rahmenbedingungen wie für die Aggregate vor der Presse. Die Problemstellung kann auch wieder in der $\alpha \mid \beta \mid \gamma$ Notation mit $PSm, 1, y_i \mid\mid \sum \omega_i U_i$ dargestellt werden.

4.1.3 Modell des Produktionsprogramms und der Fertigung

Um die in den folgenden Kapiteln beschriebenen Heuristiken anwenden und evaluieren zu können, ist es neben dem Modell der Fertigung essentiell, eine möglichst representative Menge von Aufträgen auszuwählen, mit denen der Produktionsablauf simuliert werden kann. Da es sich in der Realität um immer wieder kehrende Aufträge handelt, wurde auf Basis der Einlastung eines Jahres eine ABC Analyse durchgeführt und jene Produkte ausgwählt, die am

meisten Maschinenkapazität beanspruchen. Weiters ist noch zu berücksichtigen, dass es in der Realität auch eine zweite Presse gibt, die aufgrund der deutlich geringeren Einlastung in der Simulation nicht vorkommt, jedoch auch diese über dieselben Aggregate wie die erste Presse mit Pressgut versorgt wird. Um trotzdem ein möglichst representatives Modell zu haben, wurde wie folgt vorgegangen:

- **Auswertung der ABC Analyse**
  Gesamtanzahl an unterschiedlichen Produkten $\sim$ 300. Anzahl an Produkten mit denen 70 % der gesamten Pressenkapazität abgedeckt werden können: 100. Somit ist es notwendig, 100 Arbeitspläne und die dazugehörigen Aufträge in der Simulation anzulegen.
- **Vereinfachung der Arbeitspläne**
  Die Arbeitspläne, die in der Simulation angelegt werden sollen, sind direkt aus dem ERP-System der BSTG entnommen und bedürfen einiger Überarbeitung bzw. Ergänzungen. Aufgrund z.B. fehlender Bearbeitungs- bzw. Rüstzeiten, sowie fehlender Maschinenzuweisung unrelevante Arbeitsgänge werden aus den Arbeitsplänen entfernt. Da es teilweise für dieselben Maschinen unterschiedliche Arbeitsplatzbezeichnungen gibt, müssen diese zu einer einheitlichen Bezeichnung zusammengeführt werden.
- **Maschinenauswahl** Auf Basis der bereinigten Arbeitspläne werden die in der Simulation anzulegenden Maschinen ausgewählt. Die Entfernungen, bzw. Transportzeiten zwischen den Maschinen werden entweder durch Messungen oder durch Expertenbefragung ermittelt. Die Maschinenkapazitäten werden aus dem ERP System ermittelt und um jene Kapazität verringert, welche die Produkte für die zweite Presse benötigen. Maschinenkennzahlen wie MTTF[14], MTTR[15] etc., werden, falls vorhanden dem ERP-System entnommen, ansonsten wird die Maschine ohne der Möglichkeit auszufallen modelliert.
- **Anpassung der Aufträge** Aufgrund der Vereinfachung der Arbeitspläne, werden nicht alle Maschinen, die in der realen Fertigung vohanden

[14]Mean Time to Fail
[15]Mean Time to Repair

sind, in der Simulation abgebildet. Aufgrunddessen müssen die Start- und Endtermine, die aus dem ERP-System übernommen werden, angepasst werden. Zuerst werden die Aufträge nach dem ursprünglichen Starttermin gereiht. Anschließend werden für jeden Tag drei Aufträge ausgewählt und mit dem neuen Startdatum versehen (eine Einlastung von 3 Aufträgen pro Tag ist realistisch und hat sich auch nach Tests mit den Realdaten als sinnvoll erwiesen). Da nun nicht alle Arbeitsgänge der realen Fertigung in das Simulationsmodell übernommen werden, muss ein neuer, realistischer Endtermin errechnet werden. Dazu wurden alle Aufträge eines Jahres auf ihr Verhältnis von Durchlaufzeit zu pyhsikalischer Durchlaufzeit [16] analysiert und der Mittelwert aller Aufträge und somit der Flussgrad der Fertigung bestimmt. Auf Basis der gegebenen Daten liegt dieser Wert bei 5. Um nun einen neuen Endtermin zu errechnen, wurden alle für die Produktion eine Auftrages notwendigen Bearbeitungs-, Rüst- und Transportzeiten summiert, mit 5 multipliziert und zum Starttermin hinzuaddiert. Die Simulation hat gezeigt, dass die auf diesem Weg errechneten Fertigstellungstermine durchaus realistisch sind.

4.1.4 Verwendete Heuristiken

Im Rahmen dieser Arbeit ist es nicht gelungen, in der Literatur exakte Lösungsverfahren zu finden, die sich mit resource constraint scheduling Problemen beschäftigen. Größenteils werden Heuristiken oder evolutionäre Algorithmen verwendet um möglichst gute Lösungen zu finden. Algorithmen zur Bestimmung einer exakten Lösungen existieren laut [Brucker u. a. 1999] unterschiedlichen Branch and Bound Algorithmen, jedoch nur zur Lösung des $PS \mid prec \mid C_max$ Problems. Laut der Notation von [Brucker u. a. 1999] handelt es sich dabei um ein allgemeines Project Scheduling Problem (PS) mit Reihenfolgevorhaben bei den einzelnen Vorgängen ($prec$) wobei es gilt, den maximalen Fertigstellungstermin zu minimieren (C_{max}).
Da auf die beiden Fertigungsbereiche nicht das Hauptaugenmerk der Arbeit

[16] Die physikalische Durchlaufzeit ist jene Zeit, die ein Auftrag theoretisch benötigt, wenn er niemals liegt oder wartet.

liegt, wurde auf die Implementierung aufwendiger Algorithmen verzichtet, sondern sogenannte Prioritätsregeln verwendet. Prioritätsregeln haben den wesentlichen Vorteil, dass sie robust, intuitiv, leicht zu implementieren und sehr schnell sind. Weiters ist es einfach möglich, ein System um weitere, selbst definierte Prioritätsregeln zu erweitern [Kolisch 1996].
Prioritätsregeln können in unterschiedeliche Gruppen unterteilt werden [Klein 2000, Seite 181].

- **Network based rules**
  Diese Prioritätsregeln basieren auf Informationen innerhalb des Arbeitsplans, und ignorieren die Maschinen betreffende Informationen. Ein Beispiel dafür sind Prioritätsregeln, die sich nach der Vorgangsdauer oder der Anzahl von Folgevorgängen orientieren.

- **Critical path based rules**
  Diese Regeln basieren auf einer Vorwärts-, bzw. Rückwärtsterminierung und reihen dann nach den frühesten oder spätesten Start-, bzw. Endzeiten. Diese Art von Prioritätsregeln kommt vor allem beim Scheduling von Projekten vor und hat weniger Relevanz für die im Rahmen dieser Arbeit bearbeiteten Problematik, da bei Werkstattfertigungen Vorwärts- bzw. Rückwärtsterminierungen problematisch sind, da die entstehenden Warteschlangen vor bestimmten Maschinen vorab nicht eingeschätzt werden können und es daher auch keine zuverlässigen Transferzeiten zwischen der Bearbeitung auf M_1 und M_2 gibt.

- **Resource based rules**
  Diese Regeln basieren darauf, wieviel Ressourcen ein Vorgang beansprucht und reihen die Vorgänge dementsprechend.

- **Composite rules**
  Diese Regeln basieren darauf mehrere Informationen zu kombinieren. Oft wird dabei die gewichtete Summe der nach unterschiedlichen Informationen berechneten Prioritäten gebildet und darauf basierend eine Reihung vorgenommen.

Prioritätsregeln können aufgrund unterschiedlicher Kriterien differenziert werden. Kolisch verwendet in [Kolisch 1996] eine $\alpha \mid \beta \mid \chi \mid \delta$ Notation. α gibt dabei an, ob es sich um eine network, time, oder resource $\alpha = \{N, T, R\}$

basierende Regel handelt. β gibt an, ob es sich um eine statische oder dynamische $\beta = \{S, D\}$ Prioritätsregel handelt. Statisch bedeutet in diesem Kontext, dass die Priorität, die einem Auftrag an einer Arbeitsstation zugewiesen wird unverändert bleibt, während bei einer dynamischen Prioritätsregel eine Neuberechnung beim Freiwerden der Arbeitsstation erfolgen muss. [Dill 2001]
Regeln, die eine kleine Menge an Daten benötigen die direkt an der Arbeitsstation eruiert werden können werden lokal genannt, Regeln die mehr Daten benötigen global $\chi = \{L, G\}$. Weiters gibt es noch die Möglichkeit zu differenzieren, ob der Maximalwert oder der Minimalwert der Zielfunktion gewählt wird. Dies wird mit $\delta = \{B, -\}$ B gibt dabei an, dass ein Minimalwert verwendet wird (lower bound), $-$ gibt an, dass der Maximalwert ausgewählt wird. Der Minimalwert der Zielfuntkion bekommt beispielsweise bei der Frühester Fertigstellungstermin (FFT) - Regel die höchste Priorität. [Kolisch 1996]
Kolisch analysiert in [Kolisch 1996] mehrere Arbeiten, unter anderem jene von von Davis und Patterson [Davis und Patterson 1975], Alvarez-Valdes und Tamarit [Alvares-Valdes und Tamarit 1989] und wählt auf diese Weise die besten 4 klassischen Prioritätsregeln aus, die auch im weiteren Verlauf dieser Arbeit analysiert und evaluiert werden.
Tabelle 8 zeigt eine Übersicht über die in der Simulation verwendeten Prioritätsregeln, und deren Klassifizierung. Die mit * markierten Prioritätsregeln, sind jene, die Kolisch als die besten 4 klassischen Prioritätsregeln identifiziert hat. Es gäbe noch viele weitere Prioritätsregeln, deren nähere Betrachtung sehr interessant wäre, vor allem auch deren Verknüpfung und Gewichtung würde sehr viele Möglichkeiten bieten, jedoch ist dies im Rahmen dieser Arbeit nicht möglich.

Prioritätsregel	**Klassifizierung**
Größte Anzahl Folgevorgänge *	$N \mid S \mid G \mid -$
Größte Restbearbeitungszeit*	$N.T \mid S \mid G \mid -$
Frühester Fertigstellungstermin*	$N.T \mid S \mid G \mid B$
Kürzeste Schlupfzeit*	$N.T \mid S \mid G \mid B$
First In - First Out	$N \mid S \mid L \mid -$
Größter Auftragswert	$R \mid S \mid G \mid -$

Tabelle 8: Verwendete Prioritätsregeln

Eine Bewertung der einzelnen Prioritätsregeln und eine Klassifizierung zur Einsetzbarkeit bei einer bestimmten Problemstellung ist zu diesem Zeitpunkt noch unseriös. Erst die Simulation kann zeigen, welche Regel am besten geeignet ist, um die gewünschten Zielgrößen zu erreichen. In den folgenden Kapiteln wird auf die einzelnen verwendeten Prioritätsregeln näher eingegangen, wobei vor allem das Ergebnis der Simulation detaillierter betrachtet wird.
Es ist noch hinzuzufügen, dass immer nur gesamte Aufträge und keine einzelnen Produkte priorisiert werden, da die Splittung eines Loses nicht vorgesehen und auch in der Realität nicht erlaubt ist.

4.1.4.1 Größte Anzahl Folgevorgänge(GAFV)

Die Prioritätsregel betrachtet alle Arbeitspläne, der in der Warteschlange befindlichen Aufträge. Der aktuelle Arbeitsgang wird ausgewählt und die Anzahl der noch folgenden Vorgänge ermittelt. Jener Auftrag mit der größten Anzahl erhält die höchste Priorität. Die Idee dabei ist, jene Aufträge, die noch sehr viele Bearbeitungsschritte vor sich haben zu bevorzugen, da sich diese ansonsten verspäten.
Das Simulationsergebnis ist in Tabelle 9 zu sehen.

4.1.4.2 Größte Restbearbeitungszeit (GRBZ)

Die Prioritätsregel betrachtet jeden Arbeitsplan, der in der Warteschlange befindlichen Aufträge und summiert die Bearbeitungszeiten der noch verbleiben-

den Arbeitsgänge. Rüstzeiten werden dabei nicht berücksichtigt. Der auf diese Weise ermittelte Wert repräsentiert die Netto - Restbearbeitungszeit für jedes Produkt des Auftrags. Um die gesamte Restbearbeitungszeit eines Auftrags zu ermitteln, muss dieser Wert noch mit der Produktanzahl des Auftrags multipliziert werden. Jener Auftrag mit der größten Restbearbeitungszeit erhält die höchste Priorität
Die Idee dieser Prioritätsregel liegt darin, nicht nur die Anzahl der Folgevorgänge zu betrachten, sondern auch deren Bearbeitungszeit und Losgröße mit zu berücksichtigen. Somit werden jene Aufträge bevorzugt, die am meisten Maschinenkapazität in Anspruch nehmen.
Das Simulationsergebnis ist in Tabelle 9 zu sehen.

4.1.4.3 Frühester Fertigstellungstermin (FFT)

Die Prioritätsregeln betrachtet jeden Auftrag einer Warteschlange und erteilt dem Auftrag mit dem frühesten Fertigstellungstermin die höchste Priorität. Somit wird versucht, die Anzahl an zu späten Aufträgen zu reduzieren.
Das Simulationsergebnis ist in Tabelle 9 zu sehen.

4.1.4.4 Kürzeste Schlupfzeit (KSZ)

Die Prioritätsregel ist im Grunde eine Kombination aus frühester Fertigstellungstermin und größte Restbearbeitungszeit. Es wird jeder Auftrag der Warteschlange betrachtet und vom Fertigstellungstermin die Restbearbeitungszeit, wie sie in Kapitel 4.1.4.2 beschrieben ist abgezogen. Die Differenz, die dabei entsteht ist als Schlupfzeit definiert. Der Auftrag mit der geringsten Schlupfzeit erhält die höchste Priorität. Einfach gesagt werden bei dieser Prioritätsregel jene Aufträge bevorzugt, bei denen es am „knappesten“ wird.
Das Simulationsergebnis ist in Tabelle 9 zu sehen.

4.1.4.5 Größter Auftragswert (GAW)

Die Prioritätsregel betrachtet jeden Auftrag in der Warteschlange und priorisiert jenen am höchsten, der den höchsten Auftragswert aufweist. Die Idee

dahinter ist, in möglichst kurzer Zeit möglichst viel Umsatz zu produzieren. Das Simulationsergebnis ist in Tabelle 9 zu sehen.

4.1.4.6 First In - First Out (FIFO)

Die Prioritätsregel vergibt eigentlich keine Prioritäten, sondern sorgt dafür, dass die Aufträge in der Reihenfolge, in der sie eine Warteschlange betreten, diese auch wieder verlassen. Diese Prioritätsregel wird dazu implementiert, um zu sehen, ob es überhaupt Vorteile bringt, besondere Prioritätsregeln zu implementieren. Das Simulationsergebnis ist in Tabelle 9 zu sehen.

4.1.4.7 Simulationsergebnisse

In Tabelle 9 sind die Simulationsergebnisse für die einzelnen Prioritätsregeln zu sehen. Gemessen wurde dabei vor der Presse, da dies einen wichtigen Punkt in der Simulation darstellt, und nach dem letzten Aggregat der Simulation. Diese zwei Messpunkte wurden desshalb verwendet, um eine Differenzierung in sinnvolle Regeln für den ersten Fertigungsbereich und sinnvolle Regeln für den dritten Fertigungsbereich vornehmen zu können.
Gemessen wurden für beide Fertigungsbereiche folgende Werte:

- Zeitdauer bis alle Aufträge das Ende der Simulation erreicht haben ($Time$)
- Anzahl der verspäteten Aufträge am Ende des Fertigungsbereiches (#)
- Wert der verspäteten Aufträge am Ende des Fertigungsbereiches ($)
- Summe der Verpätungen der einzelnen Aufträge am Ende des Fertigungsbereichs ($\sum Time$)

Die Ergebnisse der Simulation werden in Kapitel 5 diskutiert.

Prio. Regel	$Time$[h]	$\sum Time_{Press}$[h]	$\sum Time_{End}$[h]	$\#_{Press}$	$\#_{End}$	$\$_{Press}$[GE]	$\$_{End}$[GE]
GAFV	4874	152	43.829	4	99	506.992	9.760.879
GRBZ	4894	210	50.370	7	76	883.268	6.999.601
FFT	5224	417	28.617	6	77	800.913	7.241.687
KSZ	5224	417	28.617	6	77	800.913	7.241.687
GAW	4839	102	27.672	5	74	673.935	6.926.971
FIFO	4932	289	34.051	5	92	623.827	8.494.982

Tabelle 9: Simulationsergenisse Prioritätsregeln

4.2 Lösungsansätze Pressenscheduling

In diesem Kapitel werden sämtliche Algorithmen vorgestellt, die in der Simulation verwendet werden können um die Aufträge vor den Pressen zu reihen. Insgesamt wurden sechs Algorithmen implementiert, wovon sich fünf als sinnvoll erwiesen haben. 2 Algorithmen wurden durch die BSTG vorgegeben und sind bereits in deren ERP System implementiert. Neben der Beschreibung der einzelnen Algorithmen werden auch die Simulationsergebnisse, wodurch die Güte eines Algorithmus bestimmt werden kann, dargelegt.
Diese können unabhängig vom Rest der Fertigung gestestet werden, da die einzige Zielgröße, die es zu optimieren gilt, die Wartezeit der Presse ist. Als Basis zur Simulation werden die Aufträge eines Jahres herangezogen, wobei immer der Arbeitsvorrat von zwei Wochen zur Erstellung und Bewertung eines Pressenprogramms verwendet wird. Aufgrund der hohen Anzahl an Vorgängen innerhalb eines vollständigen Pressenprogramms, ist es nicht möglich, den jeweiligen Algorithmus mit der Optimallösung zu vergleichen. Aufgrunddessen werden aus den Pressenprogrammen jeweils zehn Vorgänge entnommen und mit dem jeweiligen Algorithmus gereiht. Das Ergebnis wird anschließen mit der optimalen Lösung verglichen, wodurch eine absolute Aussage zur Güte des Algorithmus getroffen werden kann. Die Rahmenbedingungen für das Scheduling der Pressen wurden bereits in Kapitel 2.2 beschrieben.
Die detaillierten Daten, die zum Testen der einzelnen Algorithmen verwendet wurden, befinden sich im Anhang.

4.2.1 Algorithmus Böhler1

Dieser Algorithmus ist der erste der beiden Algorithmen, die bereits durch die BSTG vorgegeben wurden und bereits im ERP-System implementiert sind. Die Beschreibung dieses Algorithmus in den Projektunterlagen lautet wie folgt: Alle Vorgänge werden dem entsprechendem Ofen zugeordnet und aufsteigend nach Ofentemperatur sortiert. Anschließend wird ein Startofen gewählt. Von diesem Ofen ausgehend wird damit begonnen, die Presse mit Arbeitsgängen zu beschicken. Entsteht aufgrund einer Temperaturänderung im Ofen eine Wartezeit an der Presse, wird auf den anderen Ofen gewechselt. Diese Vorgehensweise

wird mit beiden Öfen als Startöfen durchgeführt und die bessere Variante ausgewählt.
Da die Darstellung als Pseudo- oder Javacode unübersichtlich und schwer verständlich ist, wird der Algorithmus an einem Beispiel erläutert.
Tabelle 10 beinhaltet alle verfügbaren Arbeitsgänge, die in eine möglichst optimale Reihenfolge gebracht werden sollen, um die Wartezeit an der Presse zu minimieren. Die in Tabelle 10 gelisteten Arbeitsgänge werden als Referenzbeispiel für die in Kapitel 4.2.1, 4.2.2 und 4.2.3 beschriebenen Algorithmen verwendet.

Nr.	**Ofen**	**Produkt**	**Text**	**Stück**	**Rüst-zeit[min]**	**Bearb.-Zeit[min]**	**Ofen-temp[°C]**
1	2	8151	Kalibrieren	32	29	00:01:06	960
2	1	8332	Kalibrieren	6	29	00:01:20	1038
3	2	8254	Fertigpressen	52	29	00:02:48	940
4	1	8111	Fertigpressen	10	29	00:03:00	1080
5	2	8051	Vorpressen	36	29	00:03:00	940
6	1	8400	Vorpressen, Biegen	34	29	00:03:00	1030
7	2	8267	Fertigpressen	32	29	00:03:00	920
8	1	8111	Kalibrieren	10	29	00:01:00	1030
9	2	8144	Fertigpressen	23	29	00:03:30	940
10	1	8125	Vorpressen	60	29	00:03:00	1040

Tabelle 10: Übersichtstabelle Arbeitsgänge

Als erster Schritt wird nun die Ausgangstabelle in zwei Tabellen, jeweils eine für Ofen1 und Ofen2, aufgeteilt und aufsteigend nach Ofentemperatur sortiert.
Anschließend wird ein Ofen als Startofen gewählt und begonnen, die Presse mit Arbeitsgängen zu beschicken, so lange es keine Temperaturänderung gibt. Wird nun angenommen mit Ofen2 als Startofen begonnen, wird mit Vorgang Nr. 7 zu produzieren begonnen und, da der darauf folgende Vorgang mit Nr. 3 eine höhere Temperatur aufweist, auf Ofen1 gewechselt und Vorgang Nr. 6 als nächstes produziert. Nun wird von Ofen1 die Presse weiter beschickt, bis es wieder zu einer Temperaturänderung kommt. Die nächste Temperaturän-

derung wäre dann bei Vorgang Nr. 2, wo wieder auf Ofen2 gewechselt wird. Führt man dies zu Ende, kommt man zu der endgültigen Auftragsreihenfolge, die in Tabelle 11 ersichtlich ist.

Nr.	Ofen	Produkt	Text	Stück	Rüst-zeit[min]	Bearb.-Zeit[min]	Ofen-temp[°C]
7	2	8267	Fertigpressen	32	29	00:03:00	920
6	1	8400	Vorpressen, Biegen	34	29	00:03:00	1030
8	1	8111	Kalibrieren	10	29	00:01:00	1030
3	2	8254	Fertigpressen	52	29	00:02:48	940
5	2	8051	Vorpressen	36	29	00:03:00	940
9	2	8144	Fertigpressen	23	29	00:03:30	940
2	1	8332	Kalibrieren	6	29	00:01:20	1038
1	2	8151	Kalibrieren	32	29	00:01:06	960
10	1	8125	Vorpressen	60	29	00:03:00	1040
4	1	8111	Fertigpressen	10	29	00:03:00	1080

Tabelle 11: Endgültige Auftragsreihenfolge Algo1

Betrachtet man den Algorithmus detaillierter, kommt man zu dem Schluss, dass er zwei große Schwäche aufweist. Angenommen die beiden Listen für Ofen1 und Ofen2 beinhalten gleich viele Vorgänge, wobei bei Ofen1 nach jedem zweiten Vorgang eine Temperaturänderung stattfindet, bei Ofen2 jedoch nur nach jedem fünften Vorgang. In diesem Fall entsteht die Problematik, dass die Vorgänge auf Ofen2 zu schnell aufgebraucht werden, um die Pausen durch die Temperaturänderung bei Ofen1 zu überbrücken. Eine weitere Schwäche besteht darin, dass nicht differenziert wird, ob durch eine Temperaturänderung in den Öfen tatsächlich eine Pause an der Presse entsteht. So könnte der Fall auftreten, dass die Rüstzeit des Folgevorgangen, die Zeit für die Temperaturänderung und die notwendige Haltezeit im Ofen überschreitete, wodurch es nicht notwendig ist, auf den anderen Ofen zu wechseln.
Die Performance dieses Algorithmus ist daher sehr stark davon abhängig, wie die Verteilung der Vorgänge auf die beiden Öfen gestaltet ist und kann in ungünstigen Fällen sehr schlecht werden.

4.2.2 Algorithmus Böhler2

Dieser Algorithmus ist der zweite der beiden Algorithmen, die bereits durch die BSTG vorgegeben wurden. Die Beschreibung dieses Algorithmus in den Projektunterlagen lautet wie folgt:
Alle Vorgänge werden dem entsprechendem Ofen zugeordnet und der Ofentemperatur aufsteigend nach Ofentemperatur sortiert. Anschließend wird ein Startofen gewählt. Die Vorgangsliste des Startofens wird analysiert, und die größte entstehende Pause durch den ersten Vorgang des anderen Ofens ersetzt. Dann wird die zweitgrößte Pause durch den zweiten Vorgang des anderen Ofens ersetzt. Dies wird so lange durchgeführt, bis es entweder keine Pausen mehr zu ersetzen gibt, oder keine Vorgänge mehr zur Ersetzung von Pausen übrig sind. Sind keine Pausen mehr zu ersetzen, werden die übrigen Vorgänge an das Ende der Liste gesetzt. Wieder wird dieses Vorgehen mit beiden Öfen als Startöfen durchgeführt und das bessere Ergebnis ausgewählt.
Zum besseren Verständnis wird dieser Algorithmus wiederum am selben Beispiel wie bereits Algorithmus 1 erläutert, jedoch diesmal Ofen1 als Startofen ausgewählt. Begonnen wird wie bereits bei Algorithmus Böhler 1 mit einer Aufteilung der Vorgänge auf die beiden Öfen.
Als nächster Schritt wird Ofen1 als Startofen gewählt und die größte auftretende Temperaturänderung, zwischen Vorgang Nr.10 und Vorgang Nr.4 durch Vorgang Nr.7 von Ofen2 ersetzt. Dies wird so lange fortgeführt, bis die Abbruchbedingung erreicht ist. Die fertigen Reihungen nach diesem Algorithmus ist in Abbildung 12 zu sehen.

Nr.	Ofen	Produkt	Text	Stück	Rüst-zeit[min]	Bearb.-Zeit[min]	Ofen-temp[°C]
6	1	8400	Vorpressen, Biegen	34	29	00:03:00	1030
8	1	8111	Kalibrieren	10	29	00:01:00	1030
3	2	8254	Fertigpressen	52	29	00:02:48	940
2	1	8332	Kalibrieren	6	29	00:01:20	1038
5	2	8051	Vorpressen	36	29	00:03:00	940
10	1	8125	Vorpressen	60	29	00:03:00	1040
7	2	8267	Fertigpressen	32	29	00:03:00	920
4	1	8111	Fertigpressen	10	29	00:03:00	1080
9	2	8144	Fertigpressen	23	29	00:03:30	940
1	2	8151	Kalibrieren	32	29	00:01:06	960

Tabelle 12: Endgültige Auftragsreihenfolge Algo2

Wie bereits Algorithmus Böhler 1 weist auch dieser Algorithmus mindestens zwei große Schwächen auf. Bereits das vorangegangene Beispiel zeigt sehr gut, wo eine der Schwächen dieses Algorithmus liegt. Ersetzt man die Pausen der Größe nach mit den nach Temperatur aufsteigend sortierten Vorgängen des anderen Ofens, kann es zu sehr großen Temperatursprüngen am anderen Ofen kommen. Ersichtlich ist dies in der Vorgangsreihenfolge 5-10-7, wo Ofen2 von 940°C auf 920°C abgekühlt und im Anschluss wieder auf 940°C erhitzt werden soll.
Eine weitere Schwäche liegt wie bei Algorithmus Böhler1 auch darin, dass nicht überprüft wird, ob eine Temperaturänderung tatsächlich zu eine Wartezeit an der Presse führt.
Die Performance des Algorithmus ist sehr stark davon abhängig, in welcher Reihenfolge die Pausen des Startofens ersetzt werden und kann in ungünstigen Fällen sehr schlecht werden.

4.2.3 Algorithmus Heuristik1

Nachdem die Schwächen der beiden vorgegebenen Algorithmen anaylsiert wurden, versucht dieser Algorithmus diese Schwächen auszugleichen. Das Prinzip

der Sortierung der beiden Ofenlisten und der Pausenersetzung wird dabei beibehalten. Als erstes wird berechnet, wieviel Pausen am Startofen tatsächlich entstehen. Dies ist der Fall, wenn die Rüstzeit des Folgevorgangs kleiner ist, als die Aufwärm- bzw. Abkühlzeit plus der Haltezeit des Schmiedestücks. Im Anschluss wird ermittelt, wieviele Pausen durch Vorgänge des anderen Ofens ersetzt werden können, da möglicherweise nicht genügend Vorgänge zur Verfügung stehen. Existieren x Pausen, jedoch nur $y \leq x$ Vorgänge zur Ersetzung zur Verfügung, so werden die y größten Pausen ausgewählt. Diese werden im Anschluss durch die Vorgänge des anderen Ofens, jedoch in aufsteigender Reihenfolge ersetzt. Übrig bleibende Vorgänge werden an das Ende der Liste hinzugefügt. Dieses Vorgehen wird wiederum mit beiden Öfen als Startofen durchgeführt und das bessere Ergebnis ausgewählt.
Zum besseren Verständnis wird die Anwendung des Algorithmus am bereits bekannten Beispiel erläutert. Um Berechnen zu können, ob es zu einer tatsächlichen Pause bei einer Temperaturänderung kommt, wird ein Ofengradient von 60°C / h und eine Haltezeit von 20 [min] für jedes Schmiedestück angenommen. Die Grundliste mit den zur Verfügung stehenden Aufträgen ist dabei in Abbildung 10 ersichtlich. Als Startofen wird Ofen1 gewählt.
Es gibt eine tatsächliche Pause beim Übergang von 1040°C auf 1080°C. Der Übergang von 1030°C auf 1038°C dauert nur 8 Minuten und inklusive der Haltezeit von 20 Minuten entspricht das 28 Minuten, was geringer ist, als die 29 Minuten Rüstzeit des Folgevorgangs. Dasselbe gilt für den Übergang von 1038°C auf 1040°C. Da es nur zu einer tatsächlichen Pause kommt, besteht besteht auch kein Mangel an Ersetzungsvorgängen. Diese Pause wird durch den ersten Vorgang von Ofen2 ersetzt, un die restlichen Vorgänge von Ofen2 hinter die Vorgänge von Ofen1 gesetzt, wodurch das in Tabelle 13 ersichtliche Endergebnis zustande kommt.

Nr.	Ofen	Produkt	Text	Stück	Rüst-zeit[min]	Bearb.-Zeit[min]	Ofen-temp[°C]
6	1	8400	Vorpressen, Biegen	34	29	00:03:00	1030
8	1	8111	Kalibrieren	10	29	00:01:00	1030
2	1	8332	Kalibrieren	6	29	00:01:20	1038
10	1	8125	Vorpressen	60	29	00:03:00	1040
7	2	8267	Fertigpressen	32	29	00:03:00	920
4	1	8111	Fertigpressen	10	29	00:03:00	1080
3	2	8254	Fertigpressen	52	29	00:02:48	940
5	2	8051	Vorpressen	36	29	00:03:00	940
9	2	8144	Fertigpressen	23	29	00:03:30	940
1	2	8151	Kalibrieren	32	29	00:01:06	960

Tabelle 13: Endgültige Auftragsreihenfolge Heuristik1

Auch dieser Algorithmus weist Schwächen auf, obwohl er versucht, die Schwächen der beiden vorangegangen Algorithmen auszugleichen. Nimmt man an, dass es zwischen zwei Vorgängen zu einer Pause von 60 Minuten kommt, der Vorgang, der dies Pause füllen soll jedoch nur 40 min in Anspruch nimmt, so kommt es wiederum zu Wartezeit an der Presse. Eine Möglichkeit dies vorzubeugen wäre zu überprüfen, ob jeder Vorgang, der eine Pause ersetzen soll lange genug dauert um dies auch wirklich zu tun. Dadurch wiederum entsteht das Problem, dass nicht klar ist, welcher Vorgang stattdessen als Ersetzungsvorgang gewählt werden soll. Eine Möglichkeit wäre, die Ersetzungsvorgänge der Größe nach zu reihen und die längste Pause mit dem längsten Vorgang zu ersetzen, jedoch kommt es dann wieder zu Temperatursprüngen, die zu Pausen auf dem anderen Ofen führen können.

4.2.4 Problembeschreibung als Traveling Salesman Problem

Durch die intensive Beschäftigung mit dem Problem kam die Idee, die Problemstellung des Pressenschedulings als Traveling Salesman Problem zu betrachten.

Dass es sich dabei um kein echtes Traveling Salesman Problem handelt und worin die Unterschiede, bzw. die Gemeinsamkeiten liegen, wird in diesem Kapitel erläutert.

4.2.4.1 Allgemeines über TSP

Gegeben sei eine Menge von Städten und die Entfernung zwischen allen Städten. Die Aufgabenstellung, die Traveling Salesman Problem (TSP) genannt wird, liegt darin, den kürzesten Weg zu finden, alle Städte zu besuchen und wieder zur Ausgangsstadt zurückzukehren [Stefan und Heiner 2006, Seite 258f]. Die zu besuchenden Städte sind Knoten und die Entfernungen zwischen den Städten Kanten in einem Graph $G = (V, E)$.
Da dieser Graph vollständig ist, existiert immer mindestens eine Rundreise durch alle Städte, bei der keine Stadt mehrfach besucht wird und die Gesamtdauer der Reise minimal ist. Eine Rundreise wird auch Hamiltonkreis genannt, der per Definition ein geschlossener Pfad in einem Graphen darstellt, der jeden Knoten einmal besucht. Das TSP ist eine Spezialfall des Hamiltonkreis Problems, bei welchem nach einem kürzesten Hamiltonkreis in einem gerichteten Graphen mit Kantengewichten gesucht wird.
Weiters kann noch differenziert werden, ob es sich um ein symmetrisches oder asymmetrisches TSP handelt.
Bei einem symmetrischen TSP handelt es sich um einen ungerichteten, vollständigen Graphen, innerhalb dessen der kürzeste Hamiltonkreis gesucht werden soll. Bei einem **ungerichteten** Graphen ist die Kantenbewertung von Knoten A nach Knoten B gleich wie von Knoten B nach Knoten A ($cij = cji$).
Bei einem asymmetrischen TSP gilt es, den kürzesten Hamiltonkreis innerhalb eines vollständigen, **gerichteten** Graphen zu finden. In einem gerichteten Graphen gilt $cij \neq cji$.
Weiters gibt es noch den Fall, dass die Kantenlängen eines metrischen TSP die Dreiecksungleichung erfüllen ($c_{ij} \leq c_{ik} + c_{kj}$). Das bedeutet, dass eine direkte Verbindung zwischen zwei Knoten nie länger ist, als die Verbindung über einen dritten Knoten und sich Umwege somit nicht lohnen.
Das TSP Problem gehört der Komplexitätsklasse NP[17] an und ist auch NP vollständig. Ein Entscheidungsproblem gehört der Komplexitätsklasse NP an,

[17] nicht deterministisch polynomiell lösbar

wenn man in polynomieller Zeit überprüfen kann, ob die durch einen Algorithmus generiere Lösung auch tatsächlich eine gültige Lösung ist. Kann eine gültige Lösung in polynomieller Zeit gefunden werden, ist ein Problem der Komplexitätsklasse P[18] zuzuordnen.
Ein Entscheidungsproblem ist NP schwer, wenn alle Probleme aus NP auf dieses Problem polynomiell reduziert werden können. Würde man nun dieses Problem lösen können, so würden sich auch alle anderen Probleme aus NP in polynomieller Zeit lösen lassen. Dies ist auch der meist verwendete Ansatz um die NP Schwere eines Problems nachzuweisen.
Ein Entscheidungsproblem ist NP vollständig, wenn es sowohl der Komplexitätsklasse NP zugehörig als auch NP schwer ist.
Im Falle des TSP wäre das Entscheidungsproblem zu fragen, ob es eine Rundreise gibt, deren Kosten geringer sind als ein Wert c. Der Beweis der NP-Vollständigkeit wird meist durch die Reduktion Hamiltonkreis Problem auf das TSP vollzogen, da die NP- Vollständigkeit des Hamiltonkreisproblems bewiesen ist.

4.2.4.2 Vergleich TSP - Pressenscheduling

Vergleicht man die beiden Problemstellungen könnte man die einzelnen Pressvorgänge als Städte und die entstehenden Wartezeiten an der Presse beim Wechsel von einem Vorgang auf den nächsten, als Entfernung zwischen den Städten sehen. Das Problem läge darin, alle Städte in einer Reihenfolge zu bereisen, sodass die zurückgelegte Entfernung minimal wäre.
Es existiert aber ein wesentlicher Unterschiede zum TSP, wodurch eine detailliertere Betrachtung der Komplexität des Problems notwendig wird:
Die Entfernungen zwischen den einzelnen Städten sind nicht absolut, sondern immer davon abhängig, welcher Weg vorab zurückgelegt worden ist. Ein Beispiel, um diese Problematik zu verdeutlichen und zu begründen ist in Tabelle 14 und Tabelle 15 angeführt.
In dem Beispiel wird deutlich, dass der beim Übergang von Vorgang I zu Vorgang E (I-E) und von Vorgang E zu Vorgang J (E-J) unterschiedliche Wartezeiten an der Pressen entstehen, je nachdem welche Vorgänge zuvor gefertigt

[18] polynomiell lösbar

wurden und vor allem welche Wartezeiten dadurch zuvor entstanden sind.
Der Übergang A-B, B-F, F-G erzeugen keine Wartezeit, das sie die gleiche Temperatur benötigen, bzw. am anderen Ofen sind. Beim Übergang G-C muss die Ofentemperatur von 940°C auf 960°C erhöht werden, was 40 Minuten in Anspruch nimmt. Damit wird begonnen, sobald Vorgang B abgeschlossen ist. Da in der Zwischenzeit Vorgang F und G von Ofen2 gefertigt werden und dies 20 Minuten in Anspruch nimmt, bleibt eine Wartezeit von 20 Minuten beim Übergang G-C übrig. Beim Übergang C-H ist eine Temperaturänderung von 100°C in Ofen2 notwendig, was 200 Minuten dauert. Damit kann wiederum begonnen werden, nachdem Vorgang G abgeschlossen wurde. Da beim Übergang G-C 20 Minuten gewartet werden muss und Vorgang C 10 Minuten dauert, entspricht die tatsächliche Wartezeit beim Übergang C-H nur noch 170 Minuten. Führt man diese Berechnung weiter fort, kommt man zu dem Ergebnis, welches in Tabelle 15 angeführt ist.

Ofen1 (30°C/h)	**Bearb. Zeit[min]**	**Temp**	**Ofen2 (30°C/h)**	**Bearb. Zeit[min]**	**Temp**
A	10	940	F	10	1100
B	15	940	G	10	1100
C	10	960	H	20	1200
D	15	980	I	15	1240
E	10	1080	J	10	1250

Tabelle 14: Beispiel Presswartezeiten Vorgänge

Reihenfolge	Übergang	Wartezeit[min]
A-B-F-G-C-H-D-I-E-J (Algo Böhler1 St.Ofen 1)	A-B	0
	B-F	0
	F-G	0
	G-C	20
	C-H	170
	H-D	0
	D-I	65
	I-E	**120**
	E-J	**0**
F-G-A-B-H-I-E-J-C-D (Algo Beispiel)	F-G	0
	G-A	0
	A-B	0
	B-H	175
	H-I	80
	I-E	**0**
	E-J	**10**
	J-C	220
	C-D	40

Tabelle 15: Beispiel Presswartezeiten Zeiten

Es kann zwar für jedes Pressenproblem ein TSP generiert werden, jedoch ist dies umgekehrt nicht der Fall, womit eine polynomielle Reduktion des TSP auf das Pressenproblem nicht möglich ist und somit auch der Beweis der NP-Vollständigkeit nicht über ein TSP geführt werden kann.

Die Frage, die sich nun stellt ist die der Komplexität. Gehört das Problem des Pressenschedulings (PSP) ebenfalls zur Klasse der NP vollständigen Problemen, oder gibt es einen Algorithmus der dieses Problem in polynomieller Laufzeit löst.
Ein möglicher Beweisansatz wäre, ein Problem aus NP zu finden, welches sich in polynomieller Zeit auf das Pressenproblem reduzieren lässt. Betrachtet man das Pressenproblem nicht in Hinblick auf dessen Ähnlichkeiten mit dem TSP

sondern auf eine andere Art und Weise, können Gemeinsamkeiten zu einem anderem Problem aus NP, dem sogenannten Partition Problem gefunden werden.
Bei einem Partition Problem geht es darum eine gegebene Menge von positiven Zahlen auf zwei Mengen aufzuteilen, sodass die Differenz der Summen der Zahlen der beiden Mengen ident oder möglichst klein ist. Die mathematische Formulierung dazu lautet wie folgt:
Gegeben sei eine Menge A von N positiven Zahlen a_i. Gesucht wird eine Untermenge $A_1 \subset A$ so dass:

$$E := \left| \sum_{a_i \in A_1} a_i - \sum_{a_i \in A \setminus A_1} a_i \right|$$

minimal wird. Man nehme also an, die Vorgänge sind auf Ofen1 und Ofen2 so verteilt, dass auf Ofen2 zwei sehr große und gleich lange Pausen durch Temperaturänderungen enstehten, und auf Ofen1 sehr viele kurze Vorgänge vorhanden sind, die alle die gleiche Temperatur haben. Eine optimale Lösung des Pressenproblems wäre in diesem Fall die Vorgänge von Ofen1 so auf die beiden Pausen von Ofen2 aufzuteilen, dass die Summe der Bearbeitungs- und Rüstzeiten der Vorgänge gleich, oder deren Differenz minimal ist. Diese Problemstellung ist in Abbildung 14 noch einmal anschaulich dargestellt.

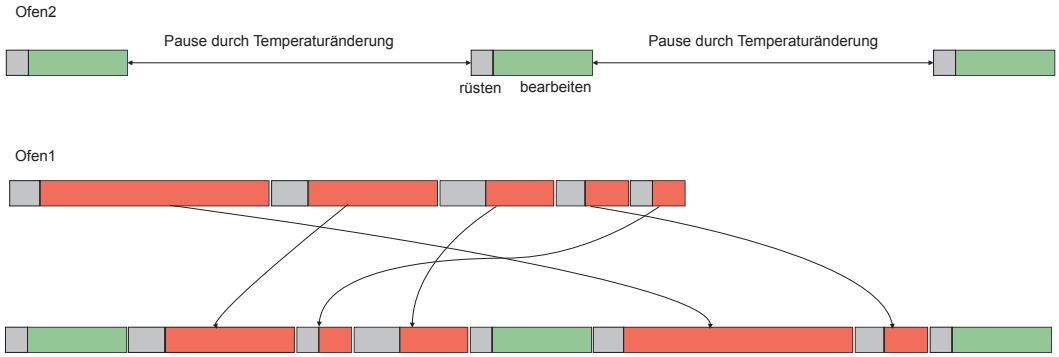


Abbildung 14: Pressenproblem als Partition Problem

Der erste Schritt des Beweises der NP Vollständigkeit des Pressenproblems besteht darin, die Zugehörigkeit zur Komplexitätsklasse NP zu zeigen. Dies geht

relativ einfach, da nur gezeigt werden muss, dass die Gültigkeit der Lösung in polynomieller Zeit überprüft werden kann.
Das Entscheidungsproblem welches es in diesem Fall zu lösen gilt lautet: Gibt es eine Auftragsreihenfolge auf beiden Öfen, wodurch die Wartezeit an der Presse kleiner ist als ein bestimmter Wert c. Da es sich bei den Auftragsreihenfolgen um einen vollständigen Graphen handelt, kann eine mögliche Lösung einfach geraten werden. Die Überprüfung der Lösung erfolgt mittels einer in der Simulation implementierten Methode, deren Funktionsweise im Sequenzdiagramm in Abbildung 15 ersichtlich ist und die in polynomieller Laufzeit die Gültigkeit und das Ergebnis einer möglichen Lösung ermittelt. In dieser Methode wird von einem Startvorgang und einer Startzeit aus überprüft, wann der nächste mögliche Zeitpunkt eines Pressvorganges wäre. Je nach Rüst- und Wärmezeiten ist dies entweder durch die Presse oder durch den Ofen bestimmt, und ausgehend von der ermittelten möglichen nächsten Presszeit, Rüst- und Wärmezeiten berechnet. Dies wird für jeden Vorgang einer Lösung durchgeführt.

Abbildung 15: Überprüfungsalgorithmus Pressenproblem

Der Nachweis der NP-Schwere kann erbracht werden, indem das Partition Problem auf des Pressenproblem reduziert wird. Die dabei verwendete Redukti-

onsfunktion f muss dabei in polynomieller Zeit deterministisch berechenbar sein. Diese Funktion f wird nun folgendermaßen definiert:

- Gegeben ist ein beliebiges Partition Problem mit einer Menge A positiver Zahlen a_i.
- Erzeuge für jede Zahl einen Vorgang auf Ofen1 mit der gleichen Temperatur, Rüstzeit $T_R = 1$ und Bearbeitungszeit $T_B = a_i - T_R$.
- Erzeuge auf Ofen2 drei Vorgänge mit unterschiedlicher Temperatur, sodass die dadurch entstehenden Pausen zwischen den Vorgängen gleich groß sind und genau so lange dauern wie $(\sum a_i/2)$. Wählen einen Vorgang als Startvorgang.

Somit wurde gezeigt, dass jedes Partition Problem in ein Pressenproblem umgewandelt werden kann. Falls es möglich wäre, Pressenprobleme in polynomieller Zeit zu lösen, könnten somit auch Partition Probleme in polynomieller Zeit gelöst werden, indem das Ergebnis des Pressenproblems auf das Partition Problem rückgeführt wird. Da jedoch das Partition Problem zu den NP-schweren Problemen gehört, ist damit auch das Pressenproblem NP schwer.
Aus den Beweisen, dass das Pressenproblem sowohl NP schwer ist, also auch der Komplexitätsklasse NP zugehörig ist, folgt, dass es auch NP vollständig ist. Dieser Beweis ist deswegen wichtig, da damit gezeigt werden kann, dass es nicht möglich ist, im Rahmen dieser Arbeit einen Algorithmus zu finden, der das Pressenproblem in polynomieller Laufzeit löst, bzw. soll die Schwierigkeit des Problems damit aufgezeigt werden.

4.2.5 Heuristik Greedy

Eine mögliche Heuristik, die oft zur Lösung von Traveling Salesman Problemen verwendet wird und dort auch häufig zu guten Lösungen führt, ist eine Nearest Neighbourhood Search, oder auch Greedy Heuristik. Aufgrund der Ähnlichkeit des Pressenproblems zum Traveling Salesman Problem (vollständiger gerichteter Graph), kann diese Heuristik auch zur Lösung des Pressenproblems verwendet werden.
Die Idee basiert darauf, ausgehend vom Startknoten den Knoten als nächstes

zu besuchen, zu dem die Kantenbewertung am geringsten ist und von diesem aus wieder den nächstgelegenen Knoten usw. Dieser Algorithmus ist sehr schnell und liefert oft sehr gute Ergebnisse.
Um diese Heuristik auch für das Pressenproblem zu verwenden sind einige zusätzliche Überlegungen notwendig.
Der Startpunkt ist beim Pressenproblem nicht vorgegeben und muss noch gewählt werden. Dabei empfiehlt es sich, einen möglichst guten Startpunkt zu wählen. Als sinnvoll hat sich dabei herausgestellt, einen Vorgang mit der niedrigsten Temperatur auf einem der beiden Öfen zu wählen.
Die Berechnung der Entfernung zu all den anderen möglichen Knoten zu denen gewechselt werden kann, erfolgt über eine eigene Methode, welche die entstehenden Hitzewartezeiten angibt. Gerade zu Begin des Problems gibt es sehr viele unterschiedliche Knoten, zu denen gewechselt werden kann, die dieselbe Entfernung, beispielsweise Null, haben. Betrachtet man das Beispiel in Tabelle 14 und 15 und man beginnt mit Vorgang A, so ergibt sich folgenden Bild.

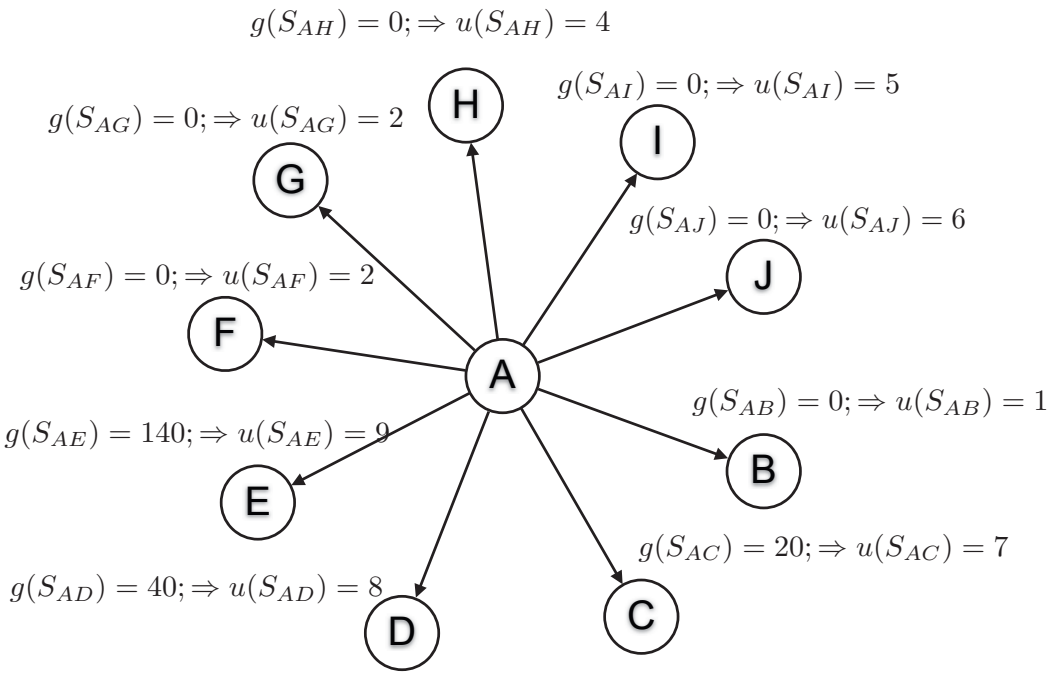


Abbildung 16: Nachbarschaft Vorgang A

Das Problem dabei ist, dass $g(S_{AX})X \in \{B, C, D, E, F, G, H, I, J\}$ oft das Ergebnis Null hat, vor allem dann wenn der Vorgang X dem anderen Ofen

zugehörig ist. Somit wird es notwendig, sich zwischen mehreren Vorgängen mit der gleichen Kantenbewertung zu entscheiden. Eine Möglichkeit wäre, einen Vorgang zufällig zu wählen, jedoch hat sicher herausgestellt, dass mit einer gewissen Systematik bessere Ergebniss erzielt werden.
Somit wurde folgende Bewertungsfunktion $u(S)$ definiert:

1. Bleibe wenn möglich auf gleichem Ofen
2. Wenn Wechsel zu anderem Ofen notwendig, wähle Vorgang mit geringster Temperatur
3. Wenn mehrere Vorgänge gleiche Temperatur haben, wähle zufällig aus diesen Vorgängen

In Abbildung 16 wurde diese Bewertungsfunktion bereits benutzt, wodurch ersichtlich ist, dass Vorgang B der nächste zu wählende Vorgang ist.
Die Erbegnis der Anwendung dieses Algorithmus ist in Tabelle 16 zu sehen. Die Grundliste mit den zur Verfügung stehenden Vorgängen ist dabei die gleiche wie bei den anderen vorgestellten Heuristiken und ist in Tabelle 10 abgebildet. Als Startofen wird Vorgang Nr. 7 in Ofen 2 gewählt. Weiters ist noch ein Wert für die Haltezeit erforderlich, dieser wird in diesem Fall für alle Teile mit einer Stunde angenommen, sowie ein Wert für die Durchwärmzeit der Schmiedestücke, die der Einfachheit halber mit 10 Minuten für alle Teile angenommen wird.
Auf Basis dieser Daten und der in Kapitel 2.2 beschriebenen Rahmenbedingungen, lassen sich $g(S)$ sowie $u(S)$ berechnen, womit die Voraussetzungen für die Durchführung des Algorithmus gegeben sind.

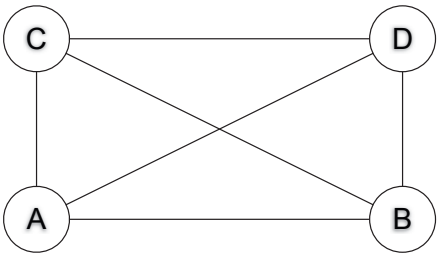
Nr.	Ofen	Produkt	Text	Stück	Rüst-zeit[min]	Bearb.-Zeit[min]	Ofen-temp[°C]
7	2	8267	Fertigpressen	32	29	00:03:00	920
8	1	8111	Kalibrieren	10	29	00:01:00	1030
6	1	8400	Biegen, Vor-pressen	34	29	00:03:00	1030
5	2	8051	Vorpressen	36	29	00:03:00	940
9	2	8144	Fertigpressen	23	29	00:03:30	940
3	2	8254	Fertigpressen	52	29	00:02:48	940
2	1	8332	Kalibrieren	6	29	00:01:20	1038
1	2	8151	Kalibrieren	32	29	00:01:06	960
10	1	8125	Vorpressen	60	29	00:03:00	1040
4	1	8111	Fertigpressen	10	29	00:03:00	1080

Tabelle 16: Endgültige Auftragsreihenfolge Greedy Heuristik

Sowohl im Ergebnis der Auftragsreihung als auch bei genauerer Betrachtung wird klar, dass dieser Algorithmus dem Algorithmus Böhler1 sehr ähnlich ist. Die Auftragsreihungen sind bei dem verwendeten Testbeispiel nämlich ident. Der Unterschied liegt im Detail darin, dass Algorithmus Böhler 1 bei einer Temperaturänderung immer auf den anderen Ofen wechselt und dort den Vorgang mit der niedrigsten Temperatur wählt, wohingegen die Greedy Heuristik immer zuerst überprüft, ob bei einer Temperaturänderung am aktuellen Ofen überhaupt eine Wartezeit an der Presse entsteht. Dies trifft zwar in den meisten Fällen zu, aber eben nicht immer. Ist die Rüstzeit für das nächste Produkt nämlich größer als die Zeit für die Temperaturänderung im Ofen und der Haltezeit entsteht keine Wartezeit und es ist nicht notwendig auf den anderen Ofen zu wechseln.
Weiters berücksichtigt dieser Algorithmus, dass eine Wechsel auf den anderen Ofen ebenfalls Wartezeiten verursachen kann. Dies ist beispielsweise der Fall, wenn von Ofen 1 auf Ofen 2 gewechselt wird, die Zeit für die Temperaturänderung und die Haltezeit des folgenden Vorgangs auf Ofen 2 jedoch länger sind, als die Rüst- und Bearbeitungszeit des vorangegangene Vorgangs auf Ofen 1. Ob und wie sich dies bei realen Daten auswirkt, ist in der Tabelle 9 zu sehen.

Die Schwächen des Algorithmus liegen darin, dass immer nur die Wartezeit zum nächsten Arbeitsgang berechnet werden kann und somit auch immer nur einen Vorgang in die Zukunft geblickt wird. Folgendes Beispiel in Abbildung 17 zeigt sehr detailliert, wann dieser Algorithmus kein optimales Ergebnis liefert.

Vorgang	**Temp.[°C]**	**Ofen**	**Rüst-zeit[min]**	**Bearb. Zeit[min]**	**Halte-zeit[min]**
A	920	1	20	50	20
B	940	1	60	70	30
C	1030	2	30	90	40
D	1060	2	30	40	30



$$\frac{g(S_{AB}) = 0; \Rightarrow u(S_{AB}) = 1}{g(S_{AC}) = 0; \Rightarrow u(S_{AC}) = 2} \quad \frac{g(S_{BC}) = 0; \Rightarrow u(S_{BC}) = 1}{g(S_{BD}) = 0; \Rightarrow u(S_{BD}) = 2} \quad \frac{g(S_{CD}) = 30; \Rightarrow u(S_{CD}) = 1}{}$$

$$\frac{g(S_{AB}) = 0; \Rightarrow u(S_{AB}) = 1}{g(S_{AC}) = 0; \Rightarrow u(S_{AC}) = 2} \quad \frac{g(S_{CB}) = 0; \Rightarrow u(S_{CB}) = 1}{g(S_{CD}) = 30; \Rightarrow u(S_{CD}) = 2} \quad \frac{g(S_{BD}) = 0; \Rightarrow u(S_{BD}) = 1}{}$$

Abbildung 17: Beispiel für ungünstige Vorgangsauswahl

Der Algorithmus würde in diesem Fall die Auftragsreihenfolge ABCD wählen, wodurch beim Übergang von Vorgang C auf Vorgang D eine Wartezeit von 30 Minuten an der Presse entstehen würde. Diese Problematik entsteht dadurch, dass dem Algorithmus vorgegeben ist, die Presse so lange wie möglich von demselben Ofen zu beschicken und erst dann auf den anderen Ofen zu wechseln. In diesem konkreten Fall führt das zu dem Problem, dass für die Ersetzung der Pause zwischen Vorgang C und D auf Ofen 1 kein Vorgang mehr zur

Verfügung steht und die Presse somit 30 Minuten auf die Teile aus dem Ofen warten müsste.

4.2.6 Genetische Algorithmen

Genetische Algorithmen sind Algorithmen, welche die Idee der natürlichen Evolution verwenden und womit für unterschiedlichen Probleme, vor allem für jene, für dies es keine effizienten Lösungsalgorithmen gibt, gute Lösungen gefunden werden können. Erfunden wurden sie von John Hollan und seinen Kollegen[Holland 1975] in den frühen 1970er Jahren. Die zum Verständnis notwendigen Begriffe werden in den folgenden Unterkapiteln genauer erläutert. Genetische Algorithmen werden in unterschiedlichen Anwendungen unterschiedlicher Wissenschaften verwendet, jedoch können drei grundlegende Problemklasse identifiziert werden:

- NUM: Numerical parameter estimation
- SEQ: Sequencing
- SUB: Subset selection

Das im Rahmen dieser Arbeit betrachtete Scheduling Problem gehört zu der Problemklasse der Sequencing Probleme und gehört, wie bereits in Kapitel 4.2.4 beschrieben, zur Klasse der NP-vollständigen Problemen.

4.2.6.1 Chromosomen und Gene

Die Struktur, die codiert, wie ein Lebewesen aufgebaut ist, ist in den Chromosomen eines Lebewesens enthalten. Jedes Lebewesen einer Population enthält ein oder mehrere solcher Chromosomen. Die Lösung eines Optimierungsproblems kann in einem String codiert werden und wird ebenfalls Chromosom genannt, wobei für die meisten kombinatorischen Optimierungsprobleme ein einzelnes Chromosom ausreicht, um eine mögliche Lösung darzustellen.
Die Symbole, die in dem Lösungsstring, also dem Chromosom abgebildet sind, werden Gene genannt.

Betrachtet man nun das Pressenscheduling Problem (PSP), so wird eine mögliche Pressenreihenfolge in einem Chromosom abgebildet, wobei die Gene des Chromosoms durch die einzelnen Vorgänge repräsentiert werden. Sollen beispielsweise zehn unterschiedliche Pressvorgänge $\in \{A, B, C, D, E, F, G, H, I, J\}$ so gereiht werden, dass eine möglichst geringe Wartezeit an der Pressen entsteht, ist ein Chromosom eine mögliche Lösung, die beispielsweise durch den String $ACEGIBDFHJ$ repräsentiert wird.

4.2.6.2 Fitness

Der Fitnesswert eine Individuums ist ein positiver Wert, der die Kosten einer Lösung angibt. Im Falle eines Minimierungsproblems gelten Lösungen mit dem geringeren Fitnesswert als fitter.
Im Falle des PSP ist der Fitnesswert die Wartezeit, die durch die im Chromosom festgelegte Auftragsreihenfolge entsteht. Die Funktion, welche den Fitnesswert bestimmt ist von den in Kapitel 2.2 beschriebenen Rahmenbedingungen abhängig. Es handelt sich dabei um eine Minimierungsproblem, da Lösungen mit geringerer Wartezeit, also auch mit geringerem Fitnesswert besser sind.

4.2.6.3 Ausgangspopulation

Genetische Algorithmen arbeiten immer mit einer Population von Individuen. Es existieren also immer eine bestimmte Anzahl von unterschiedlichen möglichen Lösungen. Die Qualität der letztendlich berechneten Lösung hängt sehr stark davon ab wie groß die Ausgangspolulation ist, und wie diese generiert wurde. Es gibt die Möglichkeit zufällige Lösungen zu generieren, oder auch bewusst Lösungen in die Ausgangspopulation zu setzten, die gezielt generiert wurden um bereits von Beginn an bessere oder schlechtere Individuen in der Poulation zu haben. Das Verfahren, bei dem gezielt Individuen in die Ausgangspolulation gesetzt werden, die nach bestimmten Heuristiken generiert worden sind, nennt man Seeding [Sadiq und Youssef 1999].
Im Rahmen dieser Arbeiten wurden sowohl die zufällige Erzeugung der Ausgangspolulation, als auch das Seeding angewandt. Als Erzeugungsheuristiken

wurden dabei in den Kapiteln 4.2.1, 4.2.2, 4.2.3 und 4.2.4 vorgestellten Verfahren verwendet.

4.2.6.4 Crossover ψ

Die Kreuzung (Crossover) ψ zweier Individuen ist ein wichtiger genetische Operator. Bei der Kreuzung werden zwei Individuen ausgewählt, die sogenannten Eltern und es werden zwei Nachkommen, die Kinder erzeugt. Die Art und Weise, wie die Kreuzung durchgeführt wird ist dafür verantwortlich, welche Eigenschaften von den Eltern geerbt werden.
In der Literatur werden unterschiedlichen Crossover Operatoren beschrieben, wobei immer darauf zu achten ist, dass der jeweilige Operator auch zum Problem passt [Sadiq und Youssef 1999]. So ist es beispielsweise nicht möglich, für das PSP eine simple crossover Operation zu verwenden.
Der simple crossover Operator beispielsweise wählt zufällig einen Gen aus (random cutting point), an dem das Chromosom abgeschnitten wird und teilt die Chromosomen der Eltern an dieser Stelle. Die beiden Kinder werden dann dadurch erzeugt, dass die linke Hälfte des Chromosoms des ersten Elternteils mit der rechten Hälfte des Chromosoms des zweiten Elternteils kombiniert wird und umgekehrt. Dieses Vorgehen ist in Abbildung 18 dargestellt.

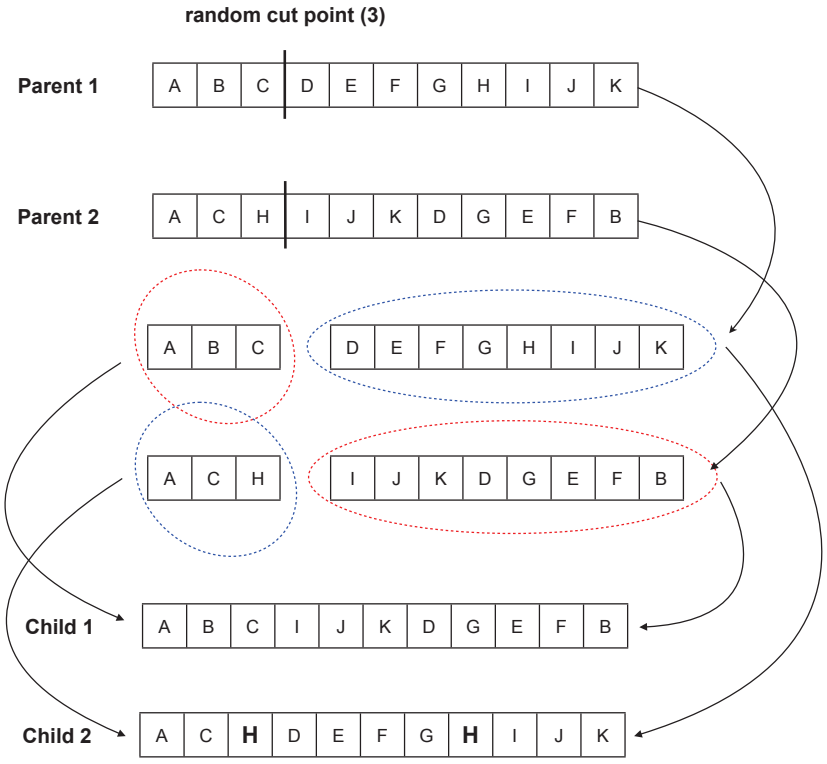


Abbildung 18: Beispiel für Simple Crossover

Wie bereits erwähnt, kann diese Art des Crossover kann nicht für das PSP angewendet werden, da wie in der Abbildung in Child 2 dargestellt ist, nicht garantiert ist, dass die Kinder wieder eine gültige Lösung des Problems sind (Vorgang H kommt im Chromosom doppelt vor).
Um dies zu verhindern, gibt es unterschiedliche Möglichkeiten. Eine mögliche davon ist die Anwendung des sogenannten Order Crossovers [Sadiq und Youssef 1999]. Dabei wird ein random cutting point ausgewählt und die linke Hälfte des ersten Elternteils (P_1) in das noch leere Chromosom des ersten Kindes kopiert. Als nächstes wird das Chromosom des zweiten Elternteils (P_2) von links ausgehend Gen für Gen durchsucht und die fehlenden Gene in der Reihenfolge wie sie in P_2 auftreten in das Chromosom des ersten Kindes kopiert. Für das zweite Kind wird gleich vorgegangen, nur wird mit P_2 begonnen. Abbildung 19 soll dieses Vorgehen verdeutlichen.

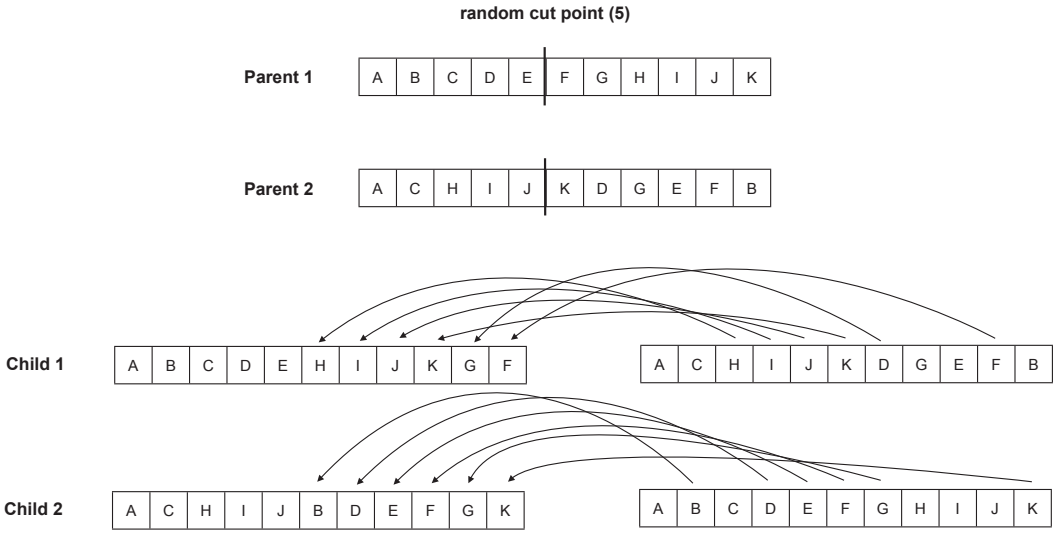


Abbildung 19: Beispiel für Order Crossover

Oliver et al schlagen vor, für die Lösung des Traveling Salesman Problem ein Cylce Crossover zu verwenden [Oliver u. a. 1987] und weisen sowohl theoretisch als auch empirisch nach, dass ein cycle crossover beim TSP zu einem besseren Ergebnis führt als ein Partitially Mapped Crossover, auf das in dieser Arbeit aber nicht weiter eingegangen wird. Aufgrund der Ähnlichkeit von TSP und PSP ist es sinnvoll, auch diese Art des Crossover zu implementieren. Beim Cycle Crossover wird eine zufälliges Gen aus dem ersten Elternchromosom gewählt und das passende Gegenstück im zweiten Elternchromosom gesucht (jenes, dass an derselben Stelle sitzt). Dieses Gen muss wiederum im ersten Elternteil gesucht werden. Dies muss so lange durchgeführt werden, bis ein Kreis entsteht.
Die Gene, die in diesem Kreis vorhanden sind, werden in beiden Kindern von den Eltern übernommen. Die restlichen Gene werden vom jeweils anderen Elternteil übernommen[Rosin u. a. 2006]. Um das eben beschriebene Vorgehen zu verdeutlichen ist dieses in Abbildung 20 noch einmal dargestellt.

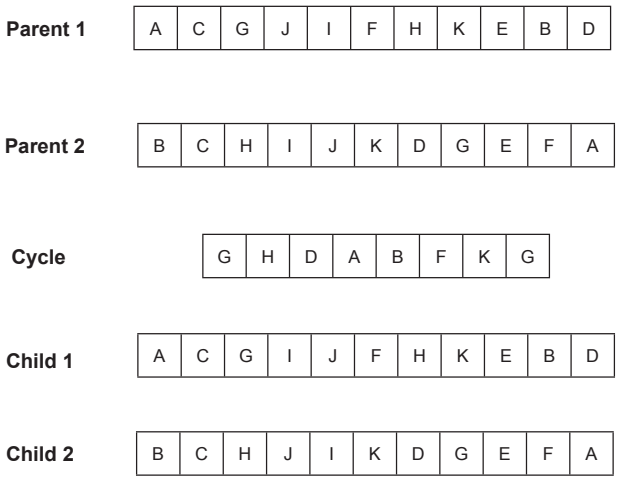


Abbildung 20: Beispiel für Cycle Crossover

Eine weitere Möglichkeit eine Crossover Operation durchzuführen, ohne jedoch wirklich zwei Individuen zu kreuzen, wäre eine Art Verschiebung einer Sequenz von Genen innerhalb eines Chromosoms. Der Vorteil dabei ist, dass die Reihenfolgeinformation innerhalb eines Chromosoms beibehalten wird und es nicht notwendig ist, sich Gedanken darüber zu machen, wie man aus zwei unterschiedlichen Individuen eine gültige Lösung konstruiert, ohne dabei die für das Pressen Scheduling Problem essentielle Reihenfolgeinformation zu verlieren.
Für die Kreuzungsopteration wählt man zufällig eine Sequenz von Genen innerhalb eines Chromosoms aus und verschiebt diese an eine zufällig gewählte Stelle innerhalb des Chromosom. Im Rahmen dieser Arbeit wird dieses Vorgehen Slide Crossover genannt. Abbildung 21 stellt diese Art des Crossover noch einmal im Detail dar.

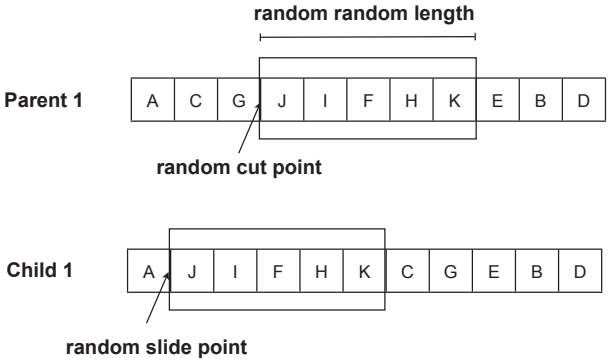


Abbildung 21: Beispiel für Slide Crossover

Es gibt noch viele weitere Crossover Operatoren, wobei jedoch im Rahmen dieser Arbeit darauf Wert gelegt wurde, die in der Literatur für Reihenfolgeprobleme empfohlenen Operatoren zu verwenden. Alle hier vorgestellten Arten des Crossover wurden auch in der Simulation implementiert und können somit miteinander verglichen werden. Die Ergebnisse der Simulation sind in Tabelle 17 zu sehen.

4.2.6.5 Elternauswahl

Die Auswahl der Individuen, welche sich fortpflanzen, ist ein sehr wichtiger Schritt, da sie die Population der neu erzeugten Generation wesentlich beeinflusst. Um die Idee der natürlichen Auslese beizubehalten, sollte die Auswahl der Eltern probabilistisch erfolgen. Das heißt, dass die Wahrscheinlichkeit, dass sich ein Individuum vermehrt umso größer ist, umso größer sein Fitnesswert ist.
Eine mögliche Variante, diese Art der Elternauswahl zu simulieren, wäre die Roulette Wheel Methode. In dieser Methode wird ein Rad konstruiert, wobei jedem Individuum einer Population ein Sektor dieses Rades zugewiesen wird, dessen Größe Abhängigkeit von der relativen Fitness des Individuums ist. Um ein Elternteil auszuwählen wird das Rad gedreht, und das dabei ausgewählte Individuum wird zur Fortpflanzung vorgemerkt. Somit haben auch Individuen mit einem schlechten Fitnesswert die Chance sich fortzupflanzen, wobei die

Wahrscheinlichkeit dementsprechend gering ist [Sadiq und Youssef 1999]. Abbildung 22 zeigt ein solches Roulette Wheel für die ebenfalls in der Abbildung ersichtlichen Beispielpopulation.

Individuum	**Vorg.Reihenfolge**	**Wartezeit[min]**	**relative Fitness**
S_1	[A B C D]	30	10,70%
S_2	[A B D C]	20	16,04%
S_3	[A C D B]	50	6,42%
S_4	[A C B D]	10	32,09%
S_5	[B A C D]	15	21,39%
S_6	[D A C B]	40	8,02%
S_7	[B C D A]	60	5,35%

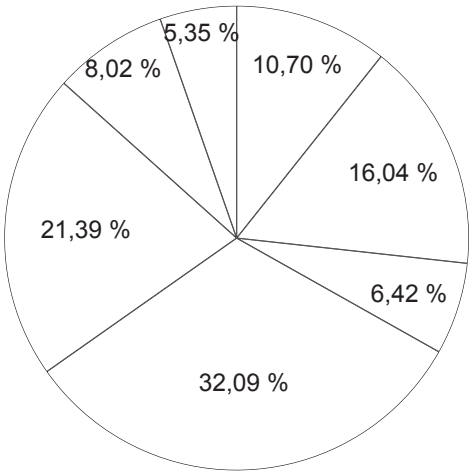


Abbildung 22: Roulette Wheel für Beispielpopulation

Die relative Fitness eines Individuums wird dabei berechnet, indem das Verhältnis der Kehrwerte einer Wartezeit zur Summe der Kehrwerte aller Wartezeiten ermittelt wird ($F_{rel_i} = \frac{1/tw_i}{\sum_j 1/tw_j}$) . Eine reine Auswahl basieren auf den Fitnesswerten der einzelnen Individuen könnte zu einer frühzeitigen Konvergenz führen, da es beispielsweise in einer Population mit wenigen sehr fitten und vielen schlechten Individuen dazu kommt, dass sich nur diese sehr guten Individuen vermehren und der Algorithmus somit möglicherweise in einem lokalen Optimum stecken bleibt [Sadiq und Youssef 1999].
Es gibt zwar unterschiedliche Verfahren dem bereits bei der Elternauswahl ent-

gegenzuwirken, jedoch wird im Rahmen dieser Arbeit darauf verzichtet, sich mit diesem Detail noch weiter zu beschäftigen.

4.2.6.6 Mutation ϕ

Mutation ist neben dem Crossover der zweite genetische Operator. Anders jedoch als das Crossover werden keine Nachkommen generiert, sondern zufällige Änderungen in den Chromosomen der Individuen der Population. Mutation ist deswegen so wichtig, da Crossover lediglich ein Vererbungsmechanismus ist, der Mutationsoperator aber neue Informationen in eine Population einbringt.
Weiters kann durch Mutation vermieden werden, dass der Algorithmus in lokalen Optima stecken bleibt. Meist wird die Mutation so realisiert, dass die Allele[19] von zufällig ausgewählten Genen eines Chromosoms verändert werden. Die Mutation wird in der Regel aber nur bei einer bestimmten Anzahl zufällig ausgewählter Individuen einer neuen Generation durchgeführt [Sadiq und Youssef 1999].
Im Rahmen dieser Arbeit ist die Mutation so implementiert, dass bei jedem Generationswechsel eine bestimmte, durch den Parameter Mutationsrate bestimmte Anzahl an Individuen der gesamten Population ausgewählt und der Mutation unterzogen werden. Bei der Mutation wird zufällig ein Gen eines Chromosoms ausgewählt und mit einem anderen, ebenfalls zufälligen ausgewählten Gen der Chromosoms vertauscht. In Abbildung 23 ist dieses Vorgehen noch einmal dargestellt.

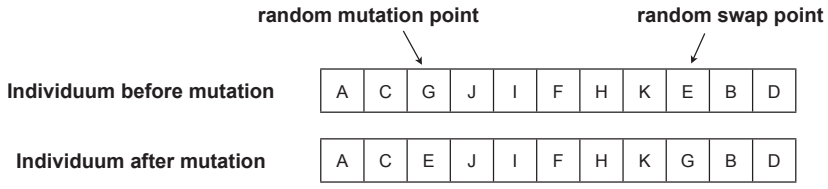


Abbildung 23: Beispiel einer Mutation

[19] Ein Allel ist jener Wert, den ein Gen annehmen kann.

4.2.6.7 Der Algorithmus

Um einen genetischen Algorithmus zu implementieren ist der wichtigste Schritt, die möglichen Lösungen eines kombinatorischen Optimierungsproblems als Chromosom, also einem String zu codieren. Mit diesem String muss es möglich sein, die genetischen Operatoren durchzuführen.
Ein vollständiger Durchlauf ist in Algorithmus 3 dargestellt. Die im Rahmen dieser Arbeit implementierten Algorithmen funktionieren alle nach diesem Schema, sie unterscheiden sich lediglich im Crossover.

Algorithm 3 Genetischer Algorithmus

M = Population size. (* *of possible solutions at any instance**)
N_g = Number of generations. (* *of iterations**)
N_o = Number of offsprings. (**to be generated by crossover**)
N_μ = Mutation probability.
$P \leftarrow \Xi(M)$ (**Construct initial population P**)
for $j = 1$ to M **do**
  Evaluate $\sigma(P[j])$ (**Evalute fitness of P**)
end for
for $i = 1$ to N_g **do**
  for $j = 1$ to N_o **do**
    $(x, y) \leftarrow \phi(P)$ (**Probability of parent selection ist proportional to fitness**)
    offspring[j] $\leftarrow \psi(x, y)$ (**Generate offsprings by crossover parents x and y**)
    Evaluate σ(offspring[j]) (**Evaluate fitness of each offspring**)
  end for
  for $j = 1$ to M **do**
    mutated[j] $\leftarrow \mu(y) Evaluate \sigma$ (mutated[j]) (**With probability P_μ apply mutation**)
  end for
  $P \leftarrow Select(P$,offsprings) (**Select best M solutions from parents and offsprings**)
end for
Return highest scoring configuration in P.

4.2.6.8 Implementierte Algorithmen

Im Rahmen der Arbeit wurden drei unterschiedliche genetische Algorithmen implementiert, vor allem um die unterschiedlichen, in Kapitel 4.2.6.4 vorgestellten Crossover Operatoren und deren Verwendbarkeit für diese Art von Problem zu evaluieren. Um immer gleiche Voraussetzungen zu schaffen, wurde die Ausgangspopulation auf 1000 Individuen, die Anzahl der Generation ebenfalls auf 1000 und die Mutationsrate auf 20 % festgelegt.
Eine Modifikation am in Kapitel 4.2.6.7 vorgestellten Algorithmus wurde noch vorgenommen, und zwar wird das beste jemals existierende Individuum gespeichert, sodass keine Verschlechterung gegenüber der Ausgangspopulation möglich ist.
In der Erzeugung der Ausgangspopulation wird das in Kapitel 4.2.6.3 beschriebene Seeding verwendet und vier Individuen nach den in Kapiteln 4.2.1, 4.2.2, 4.2.3 und 4.2.5 vorgestellten Algorithmen erzeugt. Somit ist sichergestellt, dass der genetische Algorithmus immer eine bessere oder zumindest gleich gute Lösung findet wie diese vier Heuristiken.

4.2.7 Vergleich der Ergebnisse

Das Ergebnis der Simulation der einzelnen Scheduling Algorithmen für das Pressenproblem ist in Tabelle 17 zu sehen. Die mit * gekennzeichneten Ergebnisse der genetischen Algorithmen sind jene ohne Seeding. Um die verwendeten Algorithmen und deren Ergebnisse mit einem Optimum vergleichen zu können, wurden aus jeder zu reihenden Auftragsmenge 9 Vorgänge entnommen und mittels vollständiger Permutation das optimale Ergebnis ermittelt. Anschließend wurden diese Vorgänge mit den anderen Algorithmen gereiht und das Ergebnis mit dem Optimum verglichen. Das ergebnis dazu ist ebenfalls in Tabelle 17 zu sehen.

Name	**Bearb.Zeit / Wartezeit**	**Diff. zu bestem Algorithmus**	**Differenz zu Optimum**
Böhler 1	2784h/45h	1,4% / 38h	34h / 3,8%
Böhler 2	2810h/72h	2,3% / 64h	29h / 3,1%
Böhler kombiniert	2772h/33h	0,9% / 26h	26h / 2,8%
Heuristik	2811h/73h	2,4% / 65h	26h / 2,8%
Greedy	2779h/50h	1,2% / 33h	34h / 3,7%
Gen. ordercroosover	2746h/7h	0% / 0h	1h / 0,1%
Gen. cyclecrossover	2746h/7h	0% / 0h	0h / 0%
Gen. slidecrossover	2747h/8h	0% / 1h	0h / 0%
Gen. ordercroosover*	2748h/9h	0% / 2h	1h / 0,1%
Gen. cyclecrossover*	2748h/8h	0% / 2h	0h / 0%
Gen. slidecrossover*	2749h/9h	0% / 3h	0h / 0%

Tabelle 17: Simulationsergenisse Pressenproblem

5 Diskussion der Ergebnisse

Das Simulationsergebnis der Prioritätsregeln, welches ist Tabelle 9 abgebildet ist, zeigt sehr deutlich den Einfluss von Prioritätsregeln auf die Performanz einer Fertigung. Die wesentlichen Kriterien, die über eine Empfehlung einer Prioritätsregel in diesem Fall entscheiden, sind die Anzahl der verspäteten Aufträge, deren Auftragswert und die Summe der Verspätungen. Da das Pressen auf der Spindelpresse noch relativ weit vorne in den Arbeitsplänen der Produkten liegt, haben die Prioritätsregeln keinen so großen Einfluss auf die Anzahl der verspäteten Aufträge an der Presse, wodurch diese Kennzahl nicht in die Empfehlung mit einfließt.
Der Einsatz der Prioritätsregeln *Größte Anzahl Folgevorgänge* und *First in First out* führen zu einem deutlich schlechterem Ergebnis (ca. 20 zusäztlich verspätete Aufträge) und sind somit nicht zu empfehlen. Die Prioritätsregel *Größte Restbearbeitungszeit* erzielt zwar ein gutes Ergebnis, was die Anzahl der verspäteten Aufträge und den Auftragswert betrifft, jedoch ist die Summe der Verspätungen deutlich höher als bei den anderen Regeln, wodurch auch diese Regel nicht zu empfehlen ist.
Die drei besten Prioritätsregeln *Frühester Fertigstellungstermin*, *Kürzeste Schlupfzeit* und *Größter Auftragswert* liefern sehr ähnliche und sehr gute Ergebnisse. Das idente Abschneiden von *Frühester Fertigstellungstermin* und *Kürzeste Schlupfzeit* ist auf den ersten Blick ungewöhnlich, jedoch dadurch zu erklären, dass der sehr hohe Flussfaktor 5 in der Fertigung dazu führt, dass die Bearbeitungszeiten, die in die Berechnung der Schlupfzeit einfließen, nur einen sehr geringen Wert im Verhältnis zur gesamten Auftragsdurchlaufzeit aufweisen, wodurch bei den gegebenen Realdaten, das Ergebnis bei der Priorisierung nach Schlupfzeit und nach Fertigstellungstermin immer das gleiche ist. Sollte sich der Flussfaktor in der Fertigung jedoch verringern, ist die Priorisierung nach der kürzesten Schlupfzeit zu bevorzugen.
Auch wenn bei dieser Simulation die Prioritätsregel *Größer Auftragswert* knapp

das beste Ergebnis erzielt hat, sollte Böhler Schmiedetechnik nicht auf diese Prioritätsregel umstellen, da es bei bestimmten Auftragskonstellationen dazu kommen kann, dass zwar immer die volumsstarken Aufträge rechtzeitig fertiggestellt werden, volumsschwache Aufträge hingegen sehr lange in der Fertigung verbleiben und große Verspätungen erfahren.
Für Böhler Schmiedetechnik bedeutet dieses Ergebnis, dass die Priorisierung nach der kürzesten Schlupfzeit in Zukunft auf alle Fälle zu bevorzugen ist, gegenüber der Priorisierung nach dem frühesten Fertigstellungstermin keinerlei Nachteile aufweist und die Liefertreue als wesentliche zu erfüllende Zielgröße in den Vordergrund stellt.

Das Simulationsergebnis des Pressenschedulings, welches in Tabelle 17 dargestellt ist, unterschiedet sich von den während der Entwicklung der unterschiedlichen Algorithmen und Heuristiken entstandenen Erwartungen. Der Algorithmus Böhler1, der, wie in Kapitel 4.2.1 beschrieben ist, doch wesentiche Schwächen aufweist, liegt sehr knapp am Ergebnis des Greedy Algorithmus, der das beste Ergebnis der nichtevolutionären Algorithmen arzielt. Dies liegt daran, dass die realen Daten Eigenschaften aufweisen, die diesen beiden Algorithmen sehr entgegenkommen.
Es gibt innerhalb einer in der Reihenfolge zu optimierenden Auftragsmenge sehr viele Aufträge mit derselben Ofentemperatur, sowie wenige, aber manchmal große Temperatursprüngen und dadurch resultierende eher lange Pausen. Diese Pausen können teilweise durch nur einen Vorgang des anderen Ofens nicht gefüllt werden, wodurch auch das etwas schlechtere Abschneiden von Algorithmus Böhler2 und Heuristik1 zu erklären ist, welche mit Pausenersetzungen durch alternative Vorgänge arbeiten. Das geringfügig bessere Ergebnis des Greedy Algorithmus im Vergleich zu Algorithmus Böhler1 liegt daran, dass dieser bei den gegebenen Daten sehr ähnlich arbeitet und zusätzlich noch überprüft, ob eine Temperaturänderung tatsächlich zu einer Pause an der Presse führt, wodurch in manchen Fällen ein Ersetzungsvorgang gespart werden kann, der später noch benötigt wird.
Die Analyse der Einzelergebnisse der beiden Algorithmen hat gezeigt, dass es sowohl Fälle gibt in denen Böhler2 besser ist, also auch Fälle in denen die Heuristik ein besseres Ergebnis liefert, was wiederum damit zusammenhängt dass der Algorithmus Heuristik überprüft, ob bei einer Temperaturänderung tat-

sächlich eine Pause entsteht. Dies kann auch negative Folgen für das Ergebnis haben, da es in ungünstigen Fällen dazu kommen kann, dass es beispielsweise aufgrund von zu wenigen Pausen auf Ofen1, am Ofen2 Pausen bestehen bleiben, die nicht bestehen würden, wenn in der Zwischenzeit auf den anderen Ofen gewechselt werden würde.
Ein weitere Eigenschaft der beiden Algorithmen, die während deren Entwicklung nicht bedacht worden ist, ist die Tatsache, dass bei der Auswahl des Ersetzungsvorgangen nicht darauf geachtet wird, ob durch das Entnehmen eines Vorgangs aus einer Ofenliste zusätzliche Rüstzeiten entstehen. Dies ist dann der Fall, wenn beispielsweise zwei gleiche Vorgänge unterschiedlicher Aufträge hintereinander gefertigt werden sollen und somit einmal weniger gerüstet werden muss, jedoch einer dieser beiden Vorgänge für die Ersetzung einer Pause am anderen Ofen ausgewählt wird.
Eine Kombination von Algorithmus Böhler1 und Böhler2, sodass die Ergebnisse immer verglichen werden und die bessere Auftragsreihenfolge gewählt wird, ist ein sinnvoller Ansatz, der auch in der Umsetzung bei Böhler Schmiedetechnik verwendet werden wird. Die Analyse der Simulationsergebnissen hat gezeigt, dass in ca. 75% der Fälle der Algorithmus Böhler 1 ein besseres Ergebnis liefert, durch eine Kombination der beiden Algorithmen das Gesamtergebnis noch um 12h verbessert werden könnte.
Weiters wurde analysiert, ob das gute Abschneiden der genetischen Algorithmen nicht dadurch begründet ist, dass deren Ergebnis nicht nur die Summe des jeweils besten Ergebnisses der nichtevolutionären Algorithmen darstellt. Die Analyse zeigte, dass das nicht der Fall ist, sonder sehr wohl eine Verbesserung gegenüber den jeweils besten Ergebnissen der nichtevolutionären Algorithmen erfolgt.
Die drei genetischen Algorithmen liegen im Ergebnis sehr knapp beieinander, was ebenfalls nicht zu erwarten war, da in der Literatur immer wieder auf die Wichtigkeit des passenden Crossover Operators verwiesen wird, die Ergebnisse aber kaum Unterschiede in den ansonsten gleich konfigurierten Algorithmen aufweisen. Das Simulationsergebnis zeigt aber auch, dass das Ergebnis sehr nahe am Optimum liegen muss, da die entstehenden Hitzewartezeiten bei den drei genetischen Algorithmen nur sehr gering sind und auch rein theoretisch nur noch wenig Verbesserungspotential vorhanden ist.
Die Evaluierung der Algorithmen am Optimum, deren Vorgangsweise in Ka-

pitel 4.2.7 beschrieben wird, bestätigt dies, wodurch man mit der Wahl der Algorithmen durchaus zufrieden sein kann.
Die Durchführung der Simulation des Pressenscheduling ohne Seeding zeigte, dass das Ergebnis der genetischen Algorithmen kaum schlechter ist, als das Ergebnis mit Seeding, und die genetischen Algorithmen immer noch deutlich besser abschneiden als die restlichen Algorithmen, was ebenfalls für die Verwendung genetischer Algorithmen spricht.
Zusammenfassend kann aus den Ergebnissen geschlossen werden, dass es sehr viele Einflussfaktoren und Konstellationen gibt, die sich positiv, bzw. negativ auf das Ergebnis eines bestimmten Algorithmus auswirken und kaum alle in einem einzelnen Algorithmus abzubilden sind. Auch aufgrunddessen ist es sinnvoll, genetische Algorithmen zu verwenden.
Für Böhler Schmiedetechnik bedeutet dieses Ergebnis, dass die bereits im ERP - System implementierten Algorithmen Böhler1 und Böhler2 gut für das Scheduling der Pressen geeignet sind, dass aber eine Implementierung von genetischen Algorithmen eine deutliche Verbesserung der Effizienz und Reduktion der Wartezeiten bringen kann. Aufgrund von in der Realität prozessbedingten Schwankungen der Rüst- und Bearbeitungszeiten ist es notwendig, relativ schnell auf sich ändernde Bedingungen reagieren zu können, wodurch robuste und schnelle Algorithmen zur Neuberechnung des Pressenschedulings notwendig sind. Genetische Algorithmen erfüllen diese Anforderungen, jedoch ist zu prüfen, ob eine Implementierung im ERP-System, aufgrund eingeschränkter Freiheiten in der Programmierung, möglich, bzw. ökonomisch sinnvoll ist.

6 Ausblick

Eine weitere Beschäftigung mit dem Thema des Pressenscheduling ist auf alle Fälle sinnvoll, vor allem die Verfeinerung und weitere Anpassung an die in der Realität herrschenden Fertigungsbedingungen stellt noch einiges an Herausforderung dar, birgt aber auch Verbesserungspotential für die Anwendung in der Praxis.
Die bereits angesprochenen prozessbedingten Schwankungen der Bearbeitungs- und Rüstzeiten stellen dabei ebenfalls ein Betätigungsfeld dar, wie eine detailliertere Modellierung der Öfen, deren Gradienten sich je nach Masse und Werkstoff der zu wärmenden Schmiedestücke ändern. All dies sind Einflüsse, die ein mit den derzeitigen Algorithmen erstelltes Pressenprogramm ungültig machen können, wodurch in der Realität manuell nachgebessert werden muss und die errechnete Dauer und Hitzewartezeit nicht mehr stimmt, und eventuell ein anderer Algorithmus zu einer besseren Lösung geführt hätte.
Auch eine weitere Beschäftigung mit den genetischen Algorithmen macht durchaus Sinn, da das Ergebnis gezeigt hat, dass der optimale Crossover Operator möglicherweise noch nicht gefunden wurde und hier noch Verbesserungspotential besteht. Gleiches gilt für die Mutation, der im Rahmen dieser Arbeit wenig Aufmerksamkeit geschenkt wurde, die aber ebenfalls das Ergebnis eines genetischen Algorithmus wesentlich beeinflussen kann.
Das im Rahmen dieser Arbeit geschaffene Simulationsprogramm bietet noch viel mehr Möglichkeiten zur Analyse von Fertigungssystemen als in dieser Arbeit behandelt werden konnte und stellt somit auch eine Basis für weitere Optimierungen bei Böhler Schmiedetechnik dar.

Literaturverzeichnis

Alvares-Valdes und Tamarit 1989

Alvares-Valdes ; Tamarit: Heuristic algorithms for resource-constrained scheduling problems. In: *Advances in Project Scheduling* (1989) 4.1.4

Brucker 2006

Brucker, Peter: *Scheduling Algorithms*. 5th. Springer, 2006 3.2.1, 3.2.2, 3.3.1, 3.3.2

Brucker u. a. 1999

Brucker, Peter ; Drexl, Andreas ; Möhring, Rolf ; Neumann, Klaus ; Pesch, Erwin: Resource-contrained project scheduling: Notation, classification, models and methods. In: *European Journal of Operational Research* 112 (1999), S. 3–41 4.1.1, 4.1.4

Chung 2004

Chung, Christopher A.: *Simulation Modeling Handbook*. CRC Press LLC, 2004 3.1, 3.1.1

Davis und Patterson 1975

Davis ; Patterson: A comparison of heuristic and optimal solutions in resource constrained project scheduling. In: *Management Science* 21 (1975), S. 944–955 4.1.4

Dill 2001

Dill, Christoph: *Prioritätsregeln*. `http://imihome.imi.uni-karlsruhe.de/nprioritaetsregeln_b.html`. Version: Januar 2001, Abruf: 29.08.2009 4.1.4

Graham u. a. 1976

Graham, R.L. ; Lawler, E.L. ; Lenstra, J.K. ; Kan, A.H.G. R.: Op-

timization and approximation in deterministic sequencing and scheduling: A survey. In: *Annals of Disrete Mathematics* (1976) 3.2.2, 4.1.1

Holland 1975

Holland, John H.: *Adaption in Natural and Artificial Systems*. University of Michigan Press, 1975 4.2.6

Kelton u. a. 2004

Kelton, W. D. ; Sadowski, Deborah A. ; Sadowski, Randall P.: *Simulation with Arena*. Fourth Edition. McGraw-Hill Higher Education, 2004. – ISBN 007250739X 3.1.1

Klein 2000

Klein, Robert: *Scheduling of Ressource Constraint Projects*. Kluwer Academic Publishers, 2000 4.1.4

Kolisch 1996

Kolisch, Rainer: Efficient priority rules for the resource-contrained project scheduling problem. In: *Journal of Operations Management* (1996), Nr. 14, S. 179–192 4.1.4

Oliver u. a. 1987

Oliver, I. M. ; Smith, D. J. ; Holland, J. R. C.: A study of permutation crossover operators on the traveling salesman problem. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*. Hillsdale, NJ, USA : L. Erlbaum Associates Inc., 1987. – ISBN 0–8058–0158–8, S. 224–230 4.2.6.4

Rosin u. a. 2006

Rosin, Reiner ; Milkovits, Peter ; Gräff, Thomas: *Tourenplanung mit genetischen Algorithmen*. `http://rosin-online.de/ki/titel.html`. Version: 2006, Abruf: 28.09.2009 4.2.6.4

Sadiq und Youssef 1999

Sadiq, M. S. ; Youssef, Habib: *Iterative Computer Algorithms with Applications in Engineering*. IEEE Computer Society Press, 1999 4.2.6.3, 4.2.6.4, 4.2.6.4, 4.2.6.5, 4.2.6.5, 4.2.6.6

Stefan und Heiner 2006

Stefan, Dempe ; Heiner, Schreier: *Operations Research*. Teubner, 2006 4.2.4.1

Zetschke 2003

Zetschke, Dr. P.: *Computergestützte PPS*. `http://www.drzetsche.de/Uni_pps_8.pdf`. Version: Januar 2003, Abruf: 26.08.2009 3.2.2.1

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich meine Diplomarbeit mit dem Thema

> *Simulation von Scheduling Algorithmen - Verbesserungspotentiale in der Feinplanung*

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Die Ergebnisse der Arbeit stehen ausschließlich der Böhler Schmiedetechnik GmbH und Co KG zur Verfügung. (**Arbeit mit Sperrvermerk**).

Leoben, den 24. November 2009

Thomas Winkler