TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Diplomarbeit

Entwicklung eines hybriden Algorithmus für die sequenzielle Optimierung der Reihenfolgen- und Maschinenbelegungsplanung

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines

Diplom-Ingenieurs

unter der Leitung von

Univ.-Prof. Dr.-Ing. Dipl. Wirtsch.-Ing. Prof. eh. Dr. h.c. Wilfried Sihn
(E330 Institut für Managementwissenschaften, Bereich: Betriebstechnik und Systemplanung)

Dipl.-Ing. Felix Kamhuber

(E330 Institut für Managementwissenschaften, Bereich: Betriebstechnik und Systemplanung,
Fraunhofer Austria Research GmbH)

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

von

Dominik Jacquelin, BSc

0926924 (066-482)

Kahlenbergerstraße 88, 1190 Wien

Wien, im Dezember 2019

_____
Vorname Nachname

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Ich habe zur Kenntnis genommen, dass ich zur Drucklegung meiner Arbeit unter der Bezeichnung

Diplomarbeit

nur mit Bewilligung der Prüfungskommission berechtigt bin.

Ich erkläre weiters Eides statt, dass ich meine Diplomarbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen selbstständig ausgeführt habe und alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, genannt habe.

Weiters erkläre ich, dass ich dieses Diplomarbeitsthema bisher weder im In- noch Ausland (einer Beurteilerin/einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Wien, im Dezember 2019

_____

Vorname Nachname

Danksagung

“Alone we can do so little. Together we can do so much.”

Helen Keller

An dieser Stelle möchte ich mich bei all jenen bedanken, die mich während meiner Diplomarbeit unterstützt haben und ohne die diese Diplomarbeit in dem Ausmaß sicherlich nicht möglich gewesen wäre.

Ein besonderer Dank gilt meinem Betreuer Dipl.-Ing. Felix Kamhuber, für die hervorragende inhaltliche Betreuung und die zahlreichen langen und konstruktiven Gespräche und durch den ich bei einer Projektarbeit in Kontakt mit diesem Thema gekommen bin.

Des Weiteren möchte ich mich auch bei Markus Spielbichler B.Sc. bedanken, der mich vor allem am Anfang sowohl bei programmiertechnischen Fragen als auch Fragen über das in dieser Arbeit betrachtete Unternehmen unterstützt hat.

Ein herzliches Dankeschön geht auch an meine Mutter und Lola Berger, für ihre Unterstützung beim Korrekturlesen, wodurch meine Gedanken erst ihren wahren Sinn zum Ausdruck bringen konnten.

Zu guter Letzt möchte ich diese Arbeit meinen Eltern widmen, die mich während meines ganzen Studiums und der Diplomarbeit immer bedingungslos unterstützt haben.

Kurzfassung

Für Produktionsunternehmen stellen schwankende Nachfragen eine große Herausforderung dar, die eine wirtschaftlich effiziente Planung schwierig gestalten. Die klassischen Methoden, die einer schwankenden Nachfrage entgegenwirken, werden unter anderem als Produktionsglättung bezeichnet. Diese Methoden betrachten jedoch meist nur die Produktionsmengen, wodurch ausschlaggebende Faktoren und Kosten nicht berücksichtigt werden.

Das Ziel dieser Diplomarbeit ist es, den Produktionsplan eines kunststoffverarbeitenden Unternehmens, der sich aus dem Absatzplan ergibt, bestmöglich zu glätten. Jedoch werden im Gegensatz zu den klassischen Methoden möglichst alle relevanten Kosten (Rüst-, Lager-, Verspätungs-, Betriebs- und Wartezeitkosten) und Restriktionen (Maschinen-, Werkzeug- und Rüstmitarbeiterbelegungen sowie ein vorgegebenes Rüstintervall) berücksichtigt, um einen umsetzbaren Produktionsplan mit möglichst geringen Gesamtkosten zu erzielen.

Das Problem dieser Arbeit lässt sich durch folgende Fragestellung zusammenfassen: In welcher Reihenfolge und auf welcher Maschine müssen die Aufträge eingeplant werden, damit unter Berücksichtigung aller Restriktionen und Kosten minimale Gesamtkosten entstehen? Es liegt also ein Kombinatorikproblem vor, das durch die hohe Anzahl an Aufträgen und Maschinen einen zu großen Lösungsraum besitzt, wodurch das Durchrechnen aller möglichen Kombinationen nicht realisierbar ist. In diesem Fall kommen sogenannte Metaheuristiken zum Einsatz, die Probleme in der Praxis zwar oft nicht exakt lösen können, aber in angemessener Zeit eine zufriedenstellende Lösung finden. Eine bekannte Metaheuristik ist der Genetische Algorithmus (GA), der zum Lösen der Maschinenbelegungsplanung in dieser Arbeit benutzt und in Matlab® umgesetzt wurde. Die Reihenfolgenplanung wird über einen Batchprozess vor dem Einsatz des GA definiert.

Durch eine sehr aufwändige Anpassung des GA an die Problemstellung und dem Einsatz eines zeitintensiven Decodierungsprozesses konnten alle Restriktionen und Kosten berücksichtigt werden. Als Fundament wurde das Prinzip des Grouping Genetic Algorithm (GGA) benutzt. Zum Erstellen einer guten initialen Population mit einer möglichst hohen Diversität wurden Prinzipien aus der Greedy Randomized Adaptive Search Procedure (GRASP) Metaheuristik eingesetzt. Außerdem wurden bei erweiterten Auswertungen für die Erstellung der initialen Population sowie für die Selektion und den Crossover die Distanz zwischen Individuen aktiv berücksichtigt. Letztendlich wurde der GA dann mit einem Local Search (LS) kombiniert, wodurch eine weitere Reduzierung der Gesamtkosten erzielt werden konnte. Durch eine mögliche Parallelisierung in Matlab, konnte die Rechenzeit der zeitintensiven LS Vorgänge und Decodierungsprozesse in Grenzen gehalten werden. Die Untersuchung des Batchprozesses hat gezeigt, dass durch einen sinnvollen Batchprozess und die damit verbundene Reduktion des Lösungsraums, der GA deutlich bessere Ergebnisse findet, selbst wenn dadurch unter Umständen sehr gute Lösungen ausgeschlossen werden. Eine sinnvolle und gut durchdachte Reduktion des Lösungsraums kann somit für Probleme in der Praxis zielführend sein.

Abstract

For manufacturing companies, fluctuating demands pose a major challenge that makes economically efficient planning difficult. The classical methods, which counteract a fluctuating demand, are called production smoothing methods. However, these methods usually only consider the production quantities, which does not take into account crucial factors and costs.

The aim of this thesis is the production smoothing of a plastics processing company. However, in contrast to the classical methods, all relevant costs (setup, storage, delay, operating and waiting time costs) and constraints (assignment of machines, tools and setup employees as well as a specified setup interval) are taken into account in order to achieve a feasible production plan with low overall costs.

The problem of this work can be summarized by the following question: In which order and on which machine do the orders have to be scheduled so that minimal total costs are incurred taking into account all restrictions and costs? So the considered problem is a combinatorial problem, whose solution space is too vast due to the large number of jobs and machines, so that the calculation of all possible combinations is not feasible. In this case, so-called metaheuristics are used, which often can not find the exakt solution of a practical problem, but find a satisfactory solution in a reasonable time. A well-known metaheuristic is the Genetic Algorithm (GA), which was used to solve the machine scheduling problem in this thesis and was implemented in Matlab®. The order sequence is defined via a batch process before the use of the GA.

Due to a very time-consuming adaptation of the GA to the problem and the use of a decoding process, all restrictions and costs could be taken into account. As a foundation, the principle of the Grouping Genetic Algorithm (GGA) was used. To create a good initial population with a high diversity, principles from the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristics were used. Furthermore, in extended evaluations the distance between individuals was actively used for the creation of the initial population as well as for the selection and the crossover operator. Finally, the GA was then combined with a Local Search (LS), which further reduced overall costs. Through a possible parallelization in Matlab, the computation time of the time-consuming LS and decoding processes could be kept within limits. The examination of the batch process has shown that by a reasonable batch process and the resulting reduction of the solution space, the GA finds significantly better results, even if this may consequently exclude very good solutions. A meaningful and well thought-out reduction of the solution space can thus be effective for problems in practice.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACO	Ant Colony Optimization
APO	Advanced Planner and Optimizer
APS	Advanced Planning and Scheduling
ATP	Available to Promise
bzw.	beziehungsweise
ca.	circa
d.h.	das heißt
DP	Demand Planning
ECTS	European Credit Transfer System
EDV	Elektronische Datenverarbeitung
ERP	Enterprise Resource Planning
FIFO	First In First Out
GA	Genetischer Algorithmus
GGA	Grouping Genetic Algorithm
GM	General Motors
GRASP	Greedy Radomized Adaptive Search Procedure
Hrsg.	Herausgeber
IE	Industrial Engineering
JIT	Just-in-time
LPT	Longest Processing Time
LS	Local Search

MA	Metropolis Algorithmus
MRP	Material Requirements Planning
NFL	No Free Lunch
Nr.	Nummer
NSGA	Non-dominated Sorting Genetic Algorithm
PO	Planning Office
PORV	Product and Output Rate Variation
PP/DS	Production Planning and Detailed Scheduling
PRV	Product Rate Variation
QC	Quality Control
RCL	Restricted Candidate Liste
SA	Simulated Annealing
SCC	Supply Chain Cockpit
SCM	Supply Chain Management
SNPD	Supply Network Planning and Deployment
TPS	Toyota Produktionssystem
TQC	Total Quality Control
TS	Tabu Search
u.a.	und andere, unter anderen
USA	United States of America
usw.	und so weiter
vgl.	vergleiche
VNS	Variable Neighbourhood Search
z.B.	zum Beispiel

1 Kapitel 1

Einleitung

1.1 Ausgangssituation

"Die einzige Konstante im Universum ist die Veränderung."

Heraklit von Ephesus 530 - 480 v. Ch.

Wie Heraklit schon anmerkte, ist die einzige Konstante die Veränderung, was heutzutage mehr den je auf den Bereich einer Produktion zutrifft. Es treten viele Unsicherheiten auf, die eine wirtschaftlich effiziente Planung schwierig gestalten und ein bestimmtes Ausmaß an Flexibilität voraussetzen. Produktionsunternehmen haben es immer schwerer sich im Markt zu behaupten, da die individuellen Anforderungen der Kunden immer mehr an Bedeutung gewinnen und die für die schwankende Nachfrage erforderliche Flexibilität und Anpassung seitens der industriellen Produktion eine große Herausforderung darstellen. Am Beispiel der Automobilindustrie, in der die Optionsvielfalt, die dem Kunden geboten wird, ein früher kaum vorstellbares Maß angenommen hat, ist der Druck, dem die Produktionsunternehmen dadurch ausgesetzt sind, gut ersichtlich. Nun ist es das Ziel der Produktionsunternehmen, die Nachfrage der Kunden unter diesen Bedingungen bestmöglich zu befriedigen. Diese Zielsetzung erweist sich jedoch, unter anderem durch die oft stark schwankende Nachfrage, als besondere Herausforderung.

Bereits für Produktionsunternehmen, die eine weitaus weniger komplexe Produktionsvielfalt haben als die Automobilindustrie, ist eine hohe Produktivität und Wirtschaftlichkeit genauso wichtig. Besonders kleinere Unternehmen haben es zunehmend schwerer, mit den großen Unternehmen mitzuhalten. Schaut man sich einige erfolgreiche Produktionsunternehmen im Vergleich an, lässt sich eine Gemeinsamkeit erkennen: Sie alle greifen auf Instrumente und Kenntnisse, die sich seit dem Beginn der Industrialisierung entwickelt haben, zurück und nutzen sie zu ihrem Vorteil. Dieses Wissen ist allgemein zugänglich und wird beispielsweise an Universitäten gelehrt und laufend an den aktuellen State-of-the-Art angepasst. Es gibt sogar Unternehmen, die ihr Wissen teilen und so z.B. erfolgreiche Produktionssysteme, wie das Toyota Produktionssystem (TPS) oder das Bosch Produktionssystem (BPS) für andere zugänglich machen. Dennoch gibt es weiterhin Unternehmen, die auf die Vielfalt an verfügbarem Wissen zu verzichten scheinen.

Jedes Unternehmen hat selbst zu beurteilen, welche Instrumente benutzt und folglich bestmöglich eingesetzt werden sollen. Ein in der Theorie simples, aber in der Anwendung doch komplexes Instrument ist die Reihenfolgenbildung von Aufträgen bzw. die Produktionsglättung, mit welcher sich diese Diplomarbeit beschäftigt. Einerseits ist die Reihenfolgenbildung simpel, da im Unternehmen selbst vorerst nichts grundlegendes verändert werden muss, sondern lediglich eine möglichst optimale Reihenfolge der Aufträge gesucht wird. Andererseits ist die Reihenfolgenbildung durchaus komplex, da eine effektive Kostenreduzierung eine möglichst vollständige Berücksichtigung der realen Kosten sowie aller Restriktionen voraussetzt, was praktisch kaum umsetzbar ist.

1.2 Problemstellung und Begriffserklärung

Es ist wichtig zu verstehen, dass für viele Produktionsunternehmen eine möglichst konstante Produktion von Vorteil ist. Eine konstante Produktion führt zu einer konstanten Ausbringungsmenge und so zu einer gut planbaren und optimalen Auslastung aller beteiligten Kapazitäten und Ressourcen. Schwanken die Produktionsmengen zu stark, kann es zu großen Mehrkosten in der Produktion kommen. Allerdings ist dieses Wunschbild der konstanten Produktion in der Praxis meist nicht umsetzbar. Deshalb ist es für Produktionsunternehmen wünschenswert, trotz schwankender Nachfrage eine möglichst gleichmäßige Produktion anzustreben, um dadurch die Kosten zu minimieren.

Dieses Streben nach einer möglichst gleichmäßigen Produktion wird in der Literatur auch als Produktionsmengenglättung bezeichnet. Ziel dabei ist es, den Wechsel zwischen einer „extrem hochlastigen“ und einer „extrem niedriglastigen“ Produktion zu vermeiden.[1] Je nach Komplexität des Problems kann es zu dessen Lösung von Vorteil sein, Algorithmen zu verwenden.

Die Produktionsglättung hat ihren Ursprung im Toyota Produktionssystem (TPS) und dient, wie bereits erwähnt, zur Vorbeugung von Variabilität im Produktionssystem. Die schwankenden Kundennachfragen können durch Bestände, Kapazitäten oder Zeit, bzw. durch eine Kombination der drei Faktoren gepuffert werden.[2] Es ist bekannt, dass Puffer störanfällige Prozesse bzw. unabgestimmte Kapazitäten verdecken.[3] Um diese möglichst gering zu halten, ist eine sinnvolle Produktionsglättung nötig.

Zum Lösen komplexer Probleme werden in der Literatur oft metaheuristische Verfahren benutzt. Metaheuristiken sind Algorithmen, die eine allgemeine Lösungsmöglichkeit bieten, die für eine Vielzahl von Problemen genutzt werden kann. Dazu verfügen Metaheuristiken oft über einen oder mehrere einstellbare Parameter, über die sich der Suchprozess der Lösungsfindung eines Problems anpassen lässt.[4]

[1] vgl. Tautrim, 2014, S.77
[2] vgl. Tegel, 2013, S.92
[3] vgl. Womack und Jones, 2003
[4] vgl. Rager 2008, S.62

Im Zuge dieser Masterarbeit soll ein Algorithmus zur Optimierung der Reihenfolgen- und Maschinenbelegungsplanung entwickelt werden. Die Reihenfolgenplanung wird über einen einfachen Batchprozess definiert. Der Hauptteil dieser Arbeit besteht in der Optimierung der Maschinenbelegungsplanung über eine Metaheuristik, die an die konkreten Anforderungen eines Unternehmens aus der kunststoffverarbeitenden Industrie angepasst und in Matlab®, unter der Verwendung der *Global Optimization Toolbox*, weiterentwickelt wird. Während der Bearbeitung sollen Nebenbedingungen implementiert und die Metaheuristik ausgiebig mit verschiedenen Optionen und Einstellungen getestet werden.

1.3 Forschungsfrage und Ziel der Arbeit

Ziel der Arbeit ist es also, Produktionsmengen, die sich aus dem Absatzplan des kunststoffverarbeitenden Unternehmens ableiten, bestmöglich zu glätten. In der Literatur werden Produktionsmengen oft isoliert geglättet, wodurch andere zentrale Planungsdimensionen meist vernachlässigt werden, was zu einer guten Glättung der Produktionsmengen führen kann, jedoch die Kosten letztendlich nicht zwangsläufig gesenkt werden. So ist es beispielsweise üblich, die Rüstzeiten bei der Glättung nicht zu berücksichtigen, wodurch die Praxis oft nicht gut wiedergespiegelt wird.[5] Das Hauptziel dieser Arbeit wird es daher sein zu überprüfen, inwiefern es machbar ist, ein möglichst ganzheitliches Bild der Produktionskosten zu berücksichtigen und dieses in die Produktionsglättung mit einzubeziehen. Dies wird durch die Suche nach einer optimalen Auftragsreihenfolge in der Maschinenbelegungsplanung erreicht. Letztendlich soll überprüft werden, ob dadurch eine relevante Kostenreduktion erzielt werden kann bzw. ob überhaupt alle Anforderungen effektiv mit der ausgewählten Metaheuristik umgesetzt werden können.

1.4 Methodisches Vorgehen und Struktur der Arbeit

Der Absatzplan des kunststoffverarbeitenden Unternehmens sowie alle zusätzlich benötigten Informationen werden automatisch aus einer Excel Datei übernommen. In einer ersten Initialisierungsphase (vor der Optimierung) wird untersucht, welche Aufträge über den Bestand abgedeckt sind und welche eine Produktion benötigen. Des Weiteren wird die Reihenfolgenplanung über einen sinnvollen Batchprozess festgelegt, so wie er auch in der Praxis im betrachteten Unternehmen eingesetzt wird, um die Rüstzeiten und -kosten in Grenzen zu halten. Danach werden mehrere gültige initiale Produktionspläne erstellt, die bestmöglich mit einer Metaheuristik optimiert werden sollen. Der Initialisierungs- und Optimierungsprozess wird über ein Programm, das in Matlab geschrieben wird, realisiert.

Das Besondere an der in dieser Arbeit durchgeführten Glättung ist, dass möglichst alle relevanten Kosten und Restriktionen berücksichtigt werden. Darunter fallen Rüst-, Lager-,

[5] vgl. Yavuz und Akçali, 2007, S.3583

Verzugs-, Wartezeit- sowie Maschinenbetriebskosten. Darüber hinaus werden Restriktionen, wie die Maschinen-, Werkzeug- und Rüstmitarbeiterbelegung, mit einbezogen, um die Produktion eines jeden Produktes nur dann freizugeben, wenn das benötigte Werkzeug, ein Rüstmitarbeiter und die benötigte Maschine frei sind. Die Gesamtkosten werden für jeden erstellten Produktionsplan in einer sogenannten Fitnessfuntkion bzw. Zielfunktion ausgewertet. Die durchgeführte Glättung kann durch folgende Frage zusammengefasst werden: In welcher Reihenfolge und auf welcher Maschine müssen die Aufträge eingeplant werden, damit unter Berücksichtigung aller Nebenbedingungen minimale Kosten entstehen?

Zu Beginn dieser Arbeit wird in Kapitel 2 auf die Grundlagen der Produktionsglättung eingegangen. Dabei werden die Entstehung und die klassischen Methoden der Produktionsglättung beschrieben und wie diese eine große Auswirkung auf die Wirtschaftlichkeit von Produktionsunternehmen haben können.

In Kapitel 3 wird sodann auf die Entstehung der EDV-Systeme eingegangen, die in Unternehmen heutzutage üblich sind, um Produktionspläne zu erstellen und zu optimieren, nämlich MRP-, ERP- und APS-Systeme.

Des Weiteren wird in Kapitel 4 auf die konkreten Algorithmen eingegangen, die in der Literatur zur Optimierung von Produktionsplänen benutzt werden. In diesem Fall sind das Metaheuristiken, die zur Lösung von Reihenfolgen- und Maschinenbelegungsproblemen eingesetzt werden. Im Vordergrund steht eine komplexe Auseinandersetzung mit der Definition von Metaheuristiken und den verwandten Heuristiken. Davor muss jedoch auf die Komplexitätstheorie und die Definition von Algorithmen eingegangen werden, um verstehen zu können, wann und weshalb Metaheuristiken überhaupt eingesetzt werden. Außerdem werden erfolgreiche Metaheuristiken vorgestellt und deren Vor- und Nachteile erläutert. Schließlich wird eine der vorgestellten Metaheuristiken ausgewählt.

Anschließend wird in Kapitel 5 ein sinnvoller Batchprozess definiert und die ausgesuchte Metaheuristik an die Anforderungen des betrachteten Unternehmens angepasst. Dies beinhaltet die Integration von Nebenbedingungen, wie z.B. das Berücksichtigen von Maschinenbelegungen, Werkzeugbelegungen, Rüstzeiten, Arbeitsplänen und Beständen sowie die Anpassung der Zielfunktion an die auftretenden Kosten.

Folgend wird in Kapitel 6 die angepasste Metaheuristik ausgiebig mit verschiedenen Optionen und Tuning-Varianten getestet und v.a. in Bezug auf die Lösungsqualität (möglichst geringe Gesamtkosten) und Laufzeit bewertet. Außerdem werden erweiterte Auswertungen durchgeführt, um zu ermitteln, inwiefern die Lösungsqualität weiter verbessert werden kann. Unter anderem wird untersucht, wie sich der Batchprozess auf die Lösungsqualität der Metaheuristik auswirkt.

Abschließend wird in Kapitel 7 eine Zusammenfassung der Erkenntnisse dieser Arbeit präsentiert und ein allgemeiner Ausblick gegeben.

2 Kapitel 2

Grundlagen der Produktionsglättung

Im folgenden Kapitel werden die Produktionsglättung und die in ihrem Zusammenhang in der Literatur verwendeten Begriffe erläutert. Das ist deshalb erforderlich, weil es keine einheitliche Definition der Produktionsglättung gibt, da diese unter unzählig vielen Aspekten betrachtet werden kann.

2.1 Allgemeine Definition

Einfach und anschaulich betrachtet, wird die Produktionsglättung verwendet zur:

> *„... Vermeidung von Wechseln 'extrem hochlastiger' zu 'extrem niedriglastiger' Produktion. Vermeidung des Produktionsverhaltens 'erst Vollgas' bei Spitzen, dann 'Vollbremsung' bei Auftragsmangel. Stattdessen [eine] gleichmäßige, gemittelte, ruhige Produktionslast.“*[6]

Das Ziel der Produktionsglättung ist es also, eine möglichst konstante Produktion zu ermöglichen, um so letztendlich Kosten zu sparen. Bei einer hochlastigen Produktion bzw. Produktionsüberlastung ist es oft erforderlich diesen Engpass durch Überstunden, zusätzliches Personal oder zusätzliche Kapazitäten auszugleichen, um noch rechtzeitig liefern zu können. Liegt hingegen ein Auftragsmangel vor, stehen Maschinen still bzw. werden vorhandene Kapazitäten nicht genutzt. In beiden Fällen kommt es zu unnötigen Kosten bzw. zu Verschwendungen.

Man kann sich die Produktionsglättung etwa so wie das gleichmäßige Gießen einer Blume vorstellen. Verreist man z.B. für einen Monat und gibt der Blume vor der Abfahrt die gesamte Menge Wasser, die sie in diesem einen Monat benötigt, auf einmal, wird die Blume diesen Monat höchstwahrscheinlich nicht überleben. Einerseits wird die Blume am Anfang im Wasser untergehen und andererseits wird sie irgendwann überhaupt kein Wasser mehr haben und verdursten. Benutzt man hingegen eine einfache Gießanlage, die die Gesamtmenge des benötigten Wassers über die gesamte Zeit in regelmäßigen Abständen verteilt, gedeiht die Blume bestens, wenn man von der einmonatigen Reise zurückkommt. Die Blume stellt in diesem Fall die Kapazitäten der Produktion dar und das Wasser die

[6] Tautrim, 2014, S.77

Nachfrage bzw. Aufträge. Maschinen können an einem Tag nur eine bestimmte Menge an Aufträgen bewältigen, sowie auch Blumen nur eine maximale Menge an Wasser an einem Tag vertragen. Auf den ersten Blick ist die Produktionsglättung ein sehr einfaches Prinzip, das jedoch bei der Umsetzung in Produktionsunternehmen um einiges komplexer ist, als es den Anschein hat.

In der Literatur wird häufig die *Produktionsnivellierung* synonym zur Produktionsglättung verwendet. Eine vollständigere Beschreibung in diesem Zusammenhang stellt folgende Definition dar:

> *„Hiermit ist zum einen der Ausgleich der Arbeitsinhalte mit dem Ziel einer Fließfertigung gemeint, also die Abbildung eines gleichmäßigen Taktes. Andererseits umfasst dies die Anpassung und das Modellieren der Arbeitsinhalte, um möglichst wenig Verschwendung zu erreichen. Letztlich beinhaltet die Produktionsglättung aber auch die Pufferbildung und das Vermeiden des Peitschenefektes, mit dem Ziel, einen kontinuierlichen Materialfluss und JIT-orientierte Prozesse zu erreichen.“*[7]

In dieser Definition ist besser ersichtlich, was mit einer gleichmäßigen Produktion in einem Produktionsunternehmen gemeint ist. In der Literatur wird die Produktionsglättung oft mit der Glättung von Produktionsmengen verbunden, nämlich mit einem kontinuierlichen Materialfluss bzw. gleichmäßigen Takt. Obwohl dieser Blickwinkel der Ursprung der Produktionsglättung ist, sind die Produktionsmengen in der Praxis keineswegs die einzigen Aspekte, die beachtet werden müssen.

2.2 Ursprung

Ihren Ursprung hat die Produktionsglättung in der Automobilindustrie bzw. im Toyota Produktionssystem (TPS), entwickelt von Taiichi Ohno. Um diesen Ursprung genauer zu verstehen, ist es notwendig die Entstehung des TPS genauer unter die Lupe zu nehmen.

Während in der Nachkriegszeit viele Automobilunternehmen in Europa und den USA noch vom Taylorismus geprägt und auf Massenanfertigung ausgelegt waren, fehlte in Japan dafür das nötige Kapital. Wie Taiichi Ohno in seinem Buch über das TPS schreibt, steckte hinter dem japanischen Gedanken, viele verschiedene Modelle in kleiner Stückzahl zu produzieren und trotzdem Kosten zu sparen.[8] Deshalb auch der Untertitel *Beyond Large-Scale Production*, also über die Massenproduktion hinaus.

Zusammengefasst kann man sagen, dass die Massenproduktion gerade nach der Ölkrise in 1973 immer mehr an Potential verloren hatte und eine neue Denkweise nötig war, um wirtschaftlich zu bleiben. Einen solchen Ansatz brachte das TPS.

[7] Dickmann, 2015, S.11
[8] vgl. Ohno, 1988, S.1

2.2.1 Toyota Produktionssystem

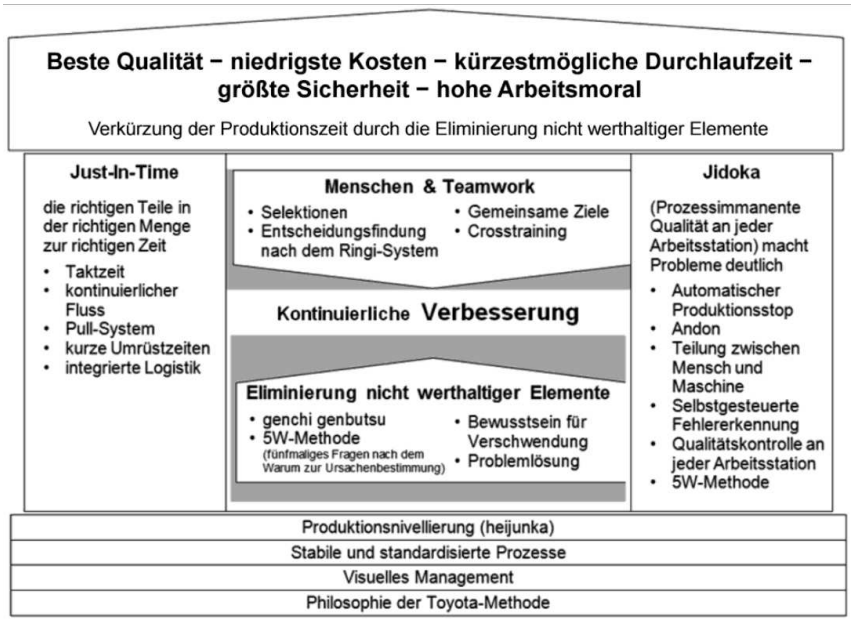

Abbildung 2.1: Das TPS-Haus (Brunner, 2017, S.127)

Taiichi Ohno schafft es, in seinem Buch über das TPS, komplexe Zusammenhänge einfach zu erklären. Doch nicht alles aus der amerikanische Automobilindustrie war für Ohno schlecht. Er lobte diese z.B. für ihre effizienten Produktionsmanagement Methoden, wie Quality Control (QC), Total Quality Control (TQC) und Industrial Engineering (IE). Jedoch geschah es gerade aus der Not heraus, um mit der amerikanischen Automobilindustrie mitzuhalten, dass Toyota über den Tellerrand hinausschauen musste, denn die Arbeitsproduktivität zwischen einem japanischen und amerikanischen Arbeiter betrug damals (1939) eins zu neun.[9] Irgendwo entstand also ein großer Anteil an Verschwendung, der gezielt eliminiert werden musste, um die Produktivität zu steigern. Genau aus diesem Gedanken entstand das Toyota Produktionssystem.[10]

Über viele Jahre wurde das TPS weiterentwickelt und vereint mittlerweile viele Konzepte, die im TPS-Haus in Abbildung 2.1 zu sehen sind. Im Fundament des Hauses findet man den Japanischen Begriff Heijunka, welcher übersetzt *Produktionsnivellierung* bzw. *Produktionsglättung* bedeutet. Die Produktionsglättung stellt also einen der Grundbausteine dar, auf dem andere Konzepte aufbauen.[11] Ohne Heijunka könnten andere Konzepte also erst gar nicht richtig funktionieren bzw. nicht in ihrem vollem Potential.

Das TPS wurde nicht nur von anderen Automobilherstellern, wie zum Beispiel Audi oder BMW, abgewandelt übernommen, sondern auch zahlreiche Unternehmen in anderen

[9] Dieses Verhältnis stammt von einer Anekdote, die Ohno erzählt und ist somit nicht mit Fakten hinterlegt, jedoch verschafft es einem einen guten Eindruck über die damaligen Unterschiede zwischen der amerikanischen und der japanischen Automobilindustrie.

[10] vgl. Ohno, 1988, S.3

[11] vgl. Redlich und Wulfsberg, 2011, S.30

Branchen, wie die Lebensmittelindustrie oder Hersteller von Elektrowerkzeugen haben die Prinzipien des TPS adaptiert und übernommen.[12] Die Produktionsglättung kann also in verschiedenen Branchen angewendet werden, jedoch muss vielleicht eine spezifische Anpassung an die jeweilige Branche erfolgen.

Um die Funktion der Produktionsglättung vollständig verstehen zu können, ist es wichtig deren Auswirkungen zu kennen. Deswegen wird hier auf einige Begriffe eingegangen, die in direkter Verbindung mit der Produktionsglättung stehen. Als kurze Veranschaulichung sind in Tabelle 2.1 (aus einer Studie aus 1992), die beeindruckenden Auswirkungen des TPS anschaulich dargestellt. In dieser Tabelle ist ersichtlich, dass General Motors (GM) Toyota in allen fünf Vergleichskriterien eindeutig unterlegen ist. Durch die Vermeidung von Verschwendung erreicht Toyota eine beachtliche Produktivität, woher auch oft in der Literatur der synonyme Begriff *Lean Production* kommt.

Vergleichskriterium	GM Framingham	Toyota Takaoka
Brutto-Montagestunden	40,7	18
Montagestunden pro Auto	31	16
Montagefehler pro 100 Autos	130	45
Montagefläche pro Auto	0,75	0,45
Teilelagerbestand (Durchschnitt)	2 Wochen	2 Stunden

Tabelle 2.1: Vergleich Automobilwerk von GM und Toyota in 1992 (Womack, 1992)

2.2.2 Definition der Produktionsglättung im TPS

Es wurden bereits zwei Definitionen der Produktionsglättung angeführt, jedoch vermitteln diese nicht den genauen Zusammenhang mit dem TPS. Deswegen wird hier nochmals genauer auf die Definition der Produktionsglättung in Bezug auf das TPS eingegangen.

> „*Durch das Heijunka-Prinzip werden die exogenen Bedarfsschwankungen nicht direkt an die Kanban-Steuerung gegeben, sondern zunächst gefiltert und gemäß einem zuvor erstellten Produktionsplan weitergeleitet.*“[13]

Es wird also im Falle der angewandten Glättung die Nachfrage in einem bestimmten Planungshorizont betrachtet und zu hohe bzw. niedrige Nachfragen im gesamten Planungszeitraum möglichst gleichmäßig verteilt. Das Heijunka-Prinzip ist somit die Schnittstelle zwischen Nachfrage und der Kanban-Steuerung.

[12] vgl. Tegel 2013, S.1f
[13] Tegel, 2013, S.17

2.2.3 KANBAN

KANBAN ist ein wichtiger Bestandteil des TPS und bedeutet wörtlich übersetzt "Karte". Die Produktionssteuerung erfolgt im TPS durch das sogenannte Kanban-Prinzip, das die Nachproduktion von Vor- bzw. Endprodukten verbrauchsorientiert steuert.[14] Es wird immer nur das produziert, was auch wirklich benötigt wird. Es funktioniert nach dem Pull-Prinzip und verläuft flussaufwärts, also vom Endprodukt aufwärts. Die KANBAN Karte gibt dazu die Verbrauchsinformation an die voranliegende Bearbeitungsstation weiter.

Die Bereitstellung der Vorprodukte verläuft nach dem *Just-in-Time* (JIT) Prinzip und soll dadurch die Bestände möglichst klein halten. Die Grundidee von JIT besteht darin, aufeinanderfolgende Prozesse so zu synchronisieren, dass jeder Prozess das Material genau dann bereitstellt, wenn der jeweilige Nachfolgeprozess es benötigt, sprich "just in time".[15]

Damit KANBAN nach dem Pull- und JIT-Prinzip in seinem vollen Potential funktionieren kann, sollte die Nachfrage bestmöglich geglättet werden. Idealerweise wird die Nachfrage also über einen bestimmten Planungshorizont geglättet und erst dann an die Kanban-Steuerung weitergeleitet. Gleichmäßigkeit und Produktionsnivellierung sind somit eine unerlässliche Voraussetzung für ein effektives Kanban-System.[16]

Um den Zusammenhang zwischen Produktionsglättung und KANBAN zu veranschaulichen, bedarf es abermals des Beispiels der Blume. Die Gießanlage hat die Aufgabe, die Blume innerhalb eines Monats z.B. mit insgesamt 3 L Wasser zu begießen. Ohne bestimmte Regeln, könnte die Gießanlage die 3 L Wasser gleich zu Beginn verbrauchen und die restliche Zeit untätig sein. Die Aufgabe ist zwar erfüllt, der Zweck aber nicht. Damit die Gießanlage diesem nachkommen kann, müssen also bestimmte Regeln aufgestellt werden. Wie viel und wann soll die Gießanlage gießen? Sprich, wie sollen die 3 L Wasser auf den Monat verteilt werden? Der Ablauf kann also folgendermaßen zusammengefasst werden: Eine bestimmte Menge an Wasser (die Nachfrage) muss über einen bestimmten Zeitraum (Planungshorizont) aufgeteilt und abgegeben werden. Die Gesamtmenge wird also nicht direkt an die Gießanlage weitergeleitet, sondern wird zuerst geglättet. Die Glättung bestimmt, wann Informationen an die Gießanlage weitergeleitet werden (also wann gegossen wird) und wieviel. Die Gießanlage kann also als eine Art Input der KANBAN-Steuerung gesehen werden. Die Gießanlage leitet die geglättete Nachfrage (Wasser) weiter an die Kapazitäten (Bedarf der Blumen).

[14] vgl. Tegel 2013, S.17
[15] vgl. Arnold, 2008, S.10
[16] vgl. Ohno, 1988, S.79

2.3 Auswirkungen der Produktionsglättung

Wie bereits erwähnt, kann die Produktionsglättung als eine Art Basis gesehen werden, auf der andere Konzepte aufbauen. Die Produktionsglättung sorgt für eine möglichst ruhige und gleichmäßige Produktion, wodurch Verschwendungen reduziert werden können. Ziel ist es immer, Bestände und Bedarfsschwankungen innerhalb der Produktion in Bezug auf Sorte und Menge zu vermeiden.[17]

Es ist wichtig zu verstehen, dass sich die Kapazitäten und der Bestand in einer Produktion im Allgemeinen immer an den Bedarfsspitzen orientieren,[18] da es anderenfalls zu Lieferverzug kommt, der oft mit erheblichen Mehrkosten verbunden ist. Durch das Reduzieren von Bedarfsspitzen können also auch die benötigten Kapazitäten und der Bestand reduziert werden. Die Produktionsglättung reduziert eben diese Bedarfsspitzen, indem diese möglichst gleichmäßig über den Planungshorizont verteilt werden. Dadurch kommt es zu einer gleichmäßigeren Auslastung der Kapazitäten, was einerseits zu weniger Leerlaufzeiten führt und andererseits zu einer Reduzierung der Überstunden, die oft benötigt werden, um Bedarfsspitzen auszugleichen. Das Augenmerk liegt nicht auf dem einzelnen Bereich bzw. einer einzelnen Maschine, sondern auf der Leistung der gesamten Produktion. Betrachtet man das ganze entlang einer ganzen Supply-Chain, können erheblich Kosten eingespart werden, denn Ruhe und Rhythmus sorgen für Verlässlichkeit in einer Supply-Chain.[19] Was uns zum sogenannten Bullwhip-Effekt führt.

2.4 Bullwhip-Effekt

Wird die Nachfrage nicht geglättet, kann dies unter anderem den sogenannten Bullwhip-Effekt (auch Peitscheneffekt gennant) begünstigen, der zu erheblichen Mehrkosten in der gesamten Logistikkette führen kann. Der Name entstand 1992 und spielt auf die Ähnlichkeit zum Schlag einer Peitsche an.[20] Die Nachfrage nimmt flussaufwärts immer mehr zu und der Effekt ist demzufolge auch als aufschaukelnde Logistikkette bekannt (siehe Abbildung 2.2).

Der Bullwhip-Effekt ist ein sehr komplexes Phänomen, das von vielen Faktoren abhängt und in der Literatur ein viel erforschtes Thema ist. Das *Beer Distribution Game*[21] ist ein bekanntes Experiment, das veranschaulicht, wie der Bullwhip-Effekt die Kosten entlang einer Logistikkette erhöht. Das Ergebnis des Experiments zeigt, dass die Leistung des Logistiknetzwerkes verringert wird, sowie die Gesamtkosten aller beteiligten Unternehmen um das 5-10-fache über den optimalen Kosten liegen.

[17] vgl. Grinninger, 2012, S.163
[18] vgl. Grinninger, 2012, S.163
[19] vgl. Bullinger u. a. 2009, S.585
[20] Sellers, P. (1992) The dumbest marketing ploy. Fortune 126 (1992) 7, S. 88-93
[21] *Sterman, 1989, S.321-339*

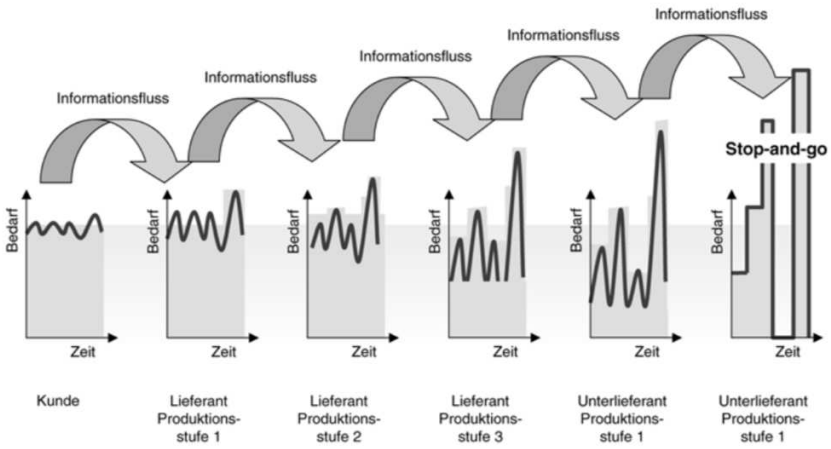

Abbildung 2.2: Der Bullwhip-Effekt (Dickmann, 2015, S.186)

Um sich den Bullwhip-Effekt besser vorstellen zu können, nehmen wir als einfachtes Beispiel die Müllentsorgung von Glasflaschen in einer Stadt. Die Einwohner der Stadt (Kunden) bestimmen, durch die Menge des Glasmülls, die Nachfrage an Glasmülltonnen und wie oft diese von der Müllabfuhr geleert werden. Die Anzahl an Glasmülltonnen und die Intervalle, in denen diese geleert werden, sollten von der Müllabfuhr so gewählt werden, dass die Mülltonnen nie übergehen. Wenn alle Einwohner ihren Glasmüll regelmäßig entsorgen (geglättete Nachfrage), kann die Müllabfuhr die Kapazitäten (Mülltonnen, Müllwägen, Personal) so auslegen, dass diese gut ausgelastet sind und somit die Mülltonnen nie übergehen, aber auch nicht entleert werden, wenn diese fast leer sind. Beschließen jedoch die Einwohner plötzlich, den Glasmüll in sehr unregelmäßigen Abständen zu entsorgen (die Gesamtmenge an Glasmüll bleibt gleich), könnte es passieren, dass die Mülltonnen in einzelnen Wochen übergehen (Bedarfsspitzen). Die Müllabfuhr würde versuchen, die Anzahl an Mülltonnen zu erhöhen bzw. die Abstände, in denen die Mülltonnen geleert werden, zu verringern. Die Nachfrage hätte somit aus Sicht der Müllabfuhr zugenommen, obwohl die Gesamtmenge an Glasmüll gleich geblieben ist. Passt die Müllabfuhr in diesem Fall ihre Kapazitäten an die Bedarfsspitzen an, gehen die Mülltonnen zwar nicht mehr über, aber sie würden eventuell geleert werden, obwohl diese fast leer sind (Bedarfstäler). Die Müllabfuhr kann also unter diesen Bedingungen ihre Kapazitäten nicht gut auslegen und es kommt zu Verschwendungen (Leerfahrten, unnötig viele Mülltonnen, Personal und Müllwägen). Die Kapazitäten der Müllabfuhr können also nur dann effektiv ausgelegt werden, wenn die Einwohner ihren Müll regelmäßig entsorgen, also die Nachfrage geglättet ist.

Am einfachen Beispiel der Müllabfuhr ist gut ersichtlich, was für Auswirkungen eine unregelmäßige Nachfrage haben kann. Wie groß die Verschwendungen entlang einer komplexen Supply-Chain werden können, ist dennoch schwer vorstellbar. Der Bullwhip-Effekt kann auch innerhalb eines Produktionsunternehmens entlang mehrere Produktionsstufen

beobachtet werden.[22] Dementsprechend ist die Produktionsglättung eine gute Methode, den Bullwhip-Effekt auch innerhalb eines Produktionsunternehmens zu dämpfen und somit die entstehenden Kosten zu reduzieren.

2.5 Glättungsmethoden

Bis jetzt wurde genauer auf den Ursprung und die Definition der Produktionsglättung eigegangen und inwiefern diese sich bemerkbar macht bzw. welche Auswirkungen sie hat. In diesem Abschnitt wird genauer auf die klassischen Glättungsmethoden eingegangen. Diese Methoden können zwar nicht direkt zur Lösung des in dieser Arbeit behandelten Problems genutzt werden, dennoch ist es wichtig, diese klassischen Methoden zu kennen, um schließlich das Prinzip der Produktionsglättung vollständig verstehen zu können.

Die Glättungsmethoden können vereinfacht in zwei Kategorien eingeteilt werden:

1. **Glättung von Produktionsreihenfolgen**: Eine bestimmte Menge an Aufträgen ist über einem bestimmten Planungshorizont bekannt. Gesucht wird eine optimale Reihenfolge, in der die Aufträge produziert werden sollen. Diese Glättung, mit der sich die vorliegende Arbeit befasst, kann als deterministisch angesehen werden, allerdings ist es durchaus möglich, diese mit stochastischen Bedingungen zu kombinieren.
2. **Bedarfsseitige Glättung in der Absatzplanung**: Die zukünftige Absatzmenge ist unbekannt, weshalb eine Bedarfsprognose erstellt werden muss. Anhand der Absatzmengen der Vergangenheit wird versucht, die zukünftige Absatzmenge vorherzusagen und zu glätten. Es wird also aktiv die Absatzmenge geglättet, wodurch die Produktion indirekt geglättet wird.

Die Bedarfsglättung wird in der Literatur auch als *demand leveling* bezeichnet und fällt hauptsächlich in den Aufgabenbereich der Marketing und Sales Abteilung. Die Produktionsglättung hingegen ist auch als *production leveling* bekannt und fällt unter die Zuständigkeit der Fertigungsabteilung.[23] Eine weitere Methode stellt die Losgrößenoptimierung dar, auf die aber in dieser Arbeit nicht genauer eingegangen wird.

2.5.1 Glättung von Produktionsreihenfolgen

Eine sehr verbreitete Art der Glättung in der Literatur ist die Optimierung der Produktionsreihenfolgen, wobei hier bei der klassischen Methode nur die Produktionsmengen betrachtet werden. Bei dieser Glättung geht es darum, eine bestimmte Anzahl an Endprodukten bzw. auch Vorprodukten möglichst gleichmäßig zu produzieren.

[22] vgl. Wiendahl, 2011, S.19
[23] vgl. Grimaud u. a. 2014, S.20f

Sehr anschaulich wird diese Methode in der Arbeit von Tegel (2013) erläutert. Tegel benutzt als Beispiel eine Produktion von drei Endprodukten A, B und C mit respektive einem Tagesbedarf von 5, 6 und 7 Einheiten. In jedem Takt kann immer nur eine Einheit eines Endproduktes produziert werden, somit werden insgesamt 18 Takte (5+6+7) pro Tag benötigt. Ein naiver erster Ansatz wäre, zuerst alle fünf Einheiten von Produkt A, dann alle sechs Einheiten von Produkt B und zu guter Letzt die restlichen sieben Einheiten von Produkt C zu produzieren (siehe Abbildung 2.3 oben). Dieser Ansatz wäre allerdings der unregelmäßigste, auch wenn das auf den ersten Blick nur schwer nachvollziehbar ist. Große Lose führen zu Bedarfsspitzen, die die vorgelagerten Prozesse zwingen, große Materialbestände vorzuhalten.[24] Bereits Ohno hat in seinem Buch über das TPS betont:

> „*It was considered common sense to produce in the largest lots possible (..). The Toyota production system however, requires (..) the smallest lots possible even though it seems contrary to conventional wisdom.*"[25]

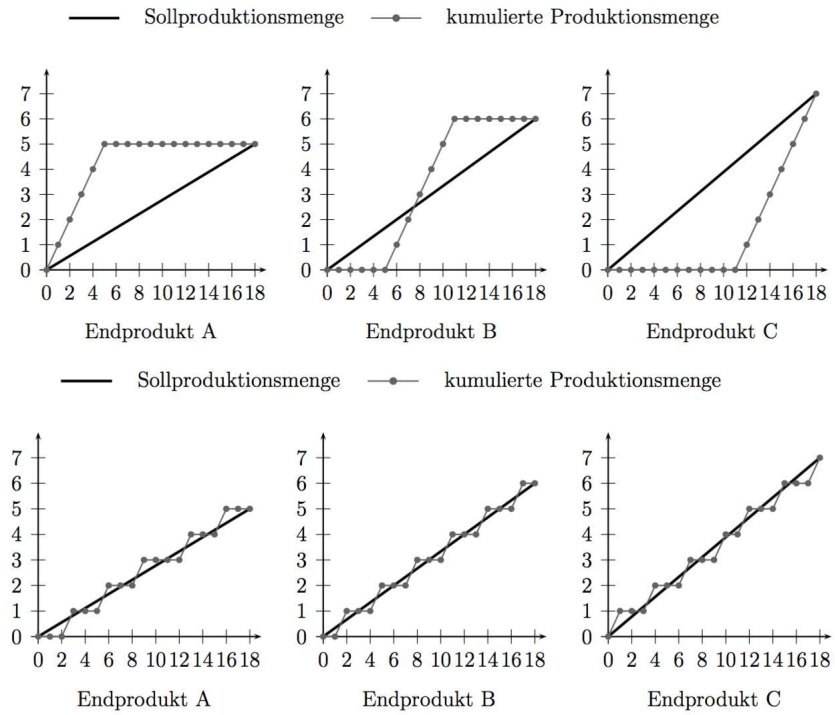

Abbildung 2.3: Produktionsreihenfolge: Naiv (oben), geglättet (unten) (Tegel, 2013, S.49ff)

Die regelmäßigste Produktion wäre also jene, die entlang der Sollproduktionsmengen verläuft. Da dies aber nicht möglich ist, weil in jedem Takt nur eine Einheit von einem Produkt produziert werden kann, muss eine Näherungslösung gefunden werden. Dieses Problem wird in der Literatur als *Product Rate Variation* (PRV) Problem bezeichnet, bei dem nur die Endprodukte betrachtet werden. Bei der Lösung des PRV Problems wird versucht folgende Zielfunktion zu minimieren:

[24] vgl. Hütter, 2008, S.73
[25] Ohno, 1988, S.38f

$$\sum_{t=1}^{18}\sum_{p=1}^{3}|x_{ptSOLL} - x_{ptIST}|, \tag{2.1}$$

in der x_{ptSOLL} die Sollmenge des Endproduktes p im Takt t darstellt und x_{ptIST} die wirklich auftretende Istmenge des Endproduktes p im Takt t ist.

Die Gleichung 2.1 gibt an, um wie viel die Kurven der kumulierten Produktionsmengen von den Sollproduktionskurven abweichen. Die erste Summe stellt die Summe über alle 18 Takte dar. Die zweite Summe stellt die Summe über alle drei Endprodukte dar. Löst man dieses Minimierungsproblem, kommt man auf die Verläufe, die in Abbildung 2.3 unten zu sehen sind.[26] Werden zusätzlich zu den Endprodukten auch die Gleichmäßigkeit der Vorprodukte berücksichtigt, spricht man in der Literatur vom *Product and Output Rate Variation* (PORV) Problem, das, abgesehen von der zusätzlichen Abweichung der Vorprodukte in der Zielfunktion, mit der gleichen Logik gelöst werden kann, wie das PRV Problem. Das PORV Problem wird vor allem dann gelöst, wenn die Endprodukte unterschiedliche Bedarfe an Vorprodukten haben. In diesem Fall kann die zusätzliche Glättung der Vorprodukte zu einem besseren Ergebnis führen.[27]

Aber wieso wirkt auf den ersten Blick die naive Reihenfolge aus Abbildung 2.3 so viel sinnvoller als die geglättete Reihenfolge? Die geglättete Reihenfolge ist mit einer hohen Anzahl an Rüstvorgängen verbunden, deren Menge den Eindruck von Ineffizienz vermitteln. Rüstvorgänge sind eine nicht wertschöpfende Tätigkeit und werden deshalb als Verschwendung angesehen.[28] Diese Einschätzung verleitet dazu in großen Losen zu produzieren, um die Anzahl an Rüstvorgängen so gering wie möglich zu halten. Das Problem an dieser Vorgehensweise ist, dass dadurch die Verschwendung der Rüstvorgänge durch Verschwendungen wie Überproduktion und hohe Lagerbestände ersetzt werden, welche einen erheblichen Anteil der betrieblichen Ineffizienz darstellen.[29] Es werden also Verschwendungen durch Verschwendungen ersetzt. Eine Lösung wäre folglich die Minimierung bzw. vollständige Eliminierung der Rüstzeiten. Hier kommt der Gedanke der Produktionsglättung ins Spiel, der dazu verleitet, die Rüstzeiten auf ein Minimum zu reduzieren, um eine effektive Glättung umsetzten zu können. Demzufolge eliminieren bereits die Voraussetzungen für eine effektive Produktionsglättung Verschwendungen. Deswegen wird in der Literatur die Produktionsglättung gelegentlich auch mit dem Streben nach minimalen Rüstzeiten und Fertigungslosen verglichen.[30]

Viele Ideen des TPS sind einfach zu verstehen, allerdings erfordert die tatsächliche Umsetzung und die Anpassung an die Gegebenheiten ein tiefes Verständnis für die Mechanismen.[31]

[26] Im TPS wird zum Lösen dieses Problems die sogenannt *Goal-Chasing-Heuristik* verwendet.
[27] vgl. Tegel, 2013, S.51
[28] vgl. Erlach, 2010, S.118
[29] vgl. Klevers, 2013, S.159
[30] vgl. Slack u. a., 2005, S.535
[31] vgl. Hopp und Spearman, 2000, S.178f

Dementsprechend gilt dies auch für das Prinzip der Produktionsglättung. Für Produktionsunternehmen, bei denen die Rüstvorgänge viel Zeit in Anspruch nehmen, wird die geglättete Reihenfolge in Abbildung 2.3 mit erheblichen Mehrkosten verbunden und somit nicht ohne Anpassung umsetzbar sein. Infolgedessen müssen in der Praxis bei der Reihenfolgenbildung, zusätzlich zur Glättung der Produktionsmengen, auch weitere Aspekte berücksichtigt werden, um letztendlich die Kosten tatsächlich zu reduzieren.

2.5.2 Bedarfsseitige Glättung in der Absatzplanung

Wie bereits erwähnt, wird mit dieser Methode die Produktion nur indirekt geglättet. Diese Methode vervollständigt jedoch sehr gut den Gedanken der Produktionsglättung, weswegen hier kurz auf die wichtigste Methode eingegangen wird, nämlich der exponentiellen Glättung.

Mit der exponentiellen Glättung lässt sich für einen kurzen Prognosezeitraum der Mittelwert und die Streuung des Periodenbedarfs aus dem Absatz der letzen Perioden berechnen.[32] Der Vorteil dieses Verfahrens (z.B. gegenüber der einfachen arithmetischen Mittelwertbildung) ist, dass Absatzmengen, die dem Prognosezeitpunkt am nächsten liegen, mit höherem Gewicht in die Berechnung eingehen, als weiter zurückliegende Absatzmengen. Erfahrungsgemäß hat sich gezeigt, dass es in der Praxis wesentlich ist, den jüngsten Entwicklungen mehr Gewicht zu geben als weiter zurückliegenden Ergebnissen.[33] Die exponentielle Glättung erster Ordnung lässt sich folgendermaßen berechnen:

$$\lambda_{(t)} = \lambda_{(t-1)} + \alpha \cdot (y_{(t-1)} - \lambda_{(t-1)}), \tag{2.2}$$

wobei $\lambda_{(t)}$ der neue Prognosewert, $\lambda_{(t-1)}$ der Prognosewert der Periode davor, $y_{(t-1)}$ den tatsächlich eingetretenen Wert der Periode davor und α der Glättungsfaktor (üblicherweise $0 \leq \alpha \leq 1$) ist. Werden aus der Vergangenheit mehr als nur eine Periode betrachtet, kommt man auf folgende Summenfunktion:

$$\lambda_{(t)} = \alpha \cdot \sum_{n=0}^{t} (1-\alpha)^n \cdot \lambda_{(t-1-n)}, \tag{2.3}$$

wobei n die Zählervariable der Perioden darstellt, die aus der Vergangenheit mit einbezogen werden. In Formel 2.3 ist gut ersichtlich, dass ein um n Perioden zurückliegender Wert gegenüber dem aktuellen Wert mit dem Gewichtsfaktor $(1-\alpha)^n$ abnimmt und infolgedessen exponentiell abnimmt.[34] Durch den Gewichtsfaktor ist gut erkennbar, dass je kleiner man α wählt, umso stärker werden die in der Vergangenheit liegenden Werte gewichtet und umso stärker ist dann zufolge die durchgeführte Glättung. Ist ein linearer Trend in der

[32] vgl. Gudehus, 2012, S.40
[33] vgl. Scharnbacher, 1992, S.153
[34] vgl. Gudehus, 2012, S.41

Nachfrage vorhanden, wird meist die exponentielle Glättung zweiter Ordnung benutzt bzw. bei Saisonalitäten die exponentielle Glättung dritter Ordnung.[35]

Die exponentielle Glättung kann also verwendet werden, um den Bedarf der Endprodukte zu glätten. Mit dem geglätteten Bedarf aller Endprodukte kann dann ein Produktionsplan erstellt werden, der wiederum mit einer Reihenfolgenoptimierung versehen wird. Sprich, beide Methoden können in der Praxis dazu beitragen, eine ruhigere und gleichmäßigere Produktion zu erzielen.

[35] vgl. Thonemann und Albers, 2010, S.65

3 Kapitel 3

Erstellen von Produktionsplänen in der Praxis

In Kapitel 2 wurde ein kurzer Überblick über die Grundlagen der Produktionsglättung gegegeben, wie dieser Gedanke überhaupt entstanden ist und welche Auswirkungen die Produktionsglättung hat. Demzufolge ist der nächste logische Schritt zu untersuchen, wie Produktionspläne in der Praxis erstellt werden und ob eine Glättung der Produktionspläne überhaupt *state of the Art* ist.

3.1 MRP-Systeme

Die meisten Produktionsplanungssysteme bauen auf dem von Joseph Orlicky bei IBM in den 70er Jahren entwickelten *Material Requirements Planning* (MRP) System auf. Obwohl es auch einige andere Autoren gab, die sich mit diesem Thema auseinandergesetzt haben, ist die Dokumentation und effektive Umsetzung dieser Systeme fraglich. Die ursprüngliche Idee der heutzutage umgesetzten MRP-Syteme stammt von Frederick W. Taylor's *Planning Office*[36] *(PO),* wobei die Ideen vom PO zu komplex und schwer verständlich waren und erst effektiv durch den Einsatz von IT-Systemen umgesetzt werden konnten.[37]

Wie der Name bereits sagt, erfüllen MRP-Systeme die Funktion der Materialbedarfsplanung. Die Idee dahinter ist, dass sich der Bedarf für Bauteile und Vorprodukte aus dem Bedarf der Endprodukte ergibt, also über den Primärbedarf sich der Sekundärbedarf herleiten lässt. Der Primärbedarf muss dementsprechend in Form einer Termin-/Mengenliste mit Einzelmengen pro Zeiteinheit vorliegen. Ist der Primärbedarf bekannt, gelangt das MRP-System über Stücklisten auf die Materialebene, dabei werden unter anderem Bestände, Durchlaufzeiten, Sicherheitsbestände und Losgrößen berücksichtigt.[38]

Zusammengefasst ist die Funktion von MRP-Systemen in folgender Definition sehr gut beschrieben:

> "*MRP-Systeme erzeugen, basierend auf der Primärbedarfsermittlung, also den Kundenbedarfen und Ersatzteilaufträgen unter Berücksichtigung der Lagerbestände, ein Produktionsprogramm mit Planaufträgen. Über die Verknüpfung*

[36] Taylor, F. W. (1903); *Shop Management.* ASME Transactions 24: 1337–1480.
[37] vgl. Wilson, 2016, S.1550
[38] vgl. Pohl, 2002, S.6

> *von Arbeitsplänen und Stücklisten werden in den Produktionsaufträgen auf der Sekundärebene wiederum Bedarfe erzeugt, aus denen in weiteren darunter liegenden* Ebenen, *Planaufträge sowie letztlich Produktions- und Beschaffungsaufträge generiert werden.*"[39]

MRP-Systeme erstellen dementsprechend nur theoretische Produktionspläne. Sie informieren darüber, wann und in welcher Menge produziert werden muss, jedoch ist nicht sichergestellt, dass dieser Produktionsplan auch umsetzbar, geschweige denn optimal ist. Daher werden MRP-Systeme als datenorientierte Systeme betrachtet und nicht optimierungsorientiert. Es handelt sich somit nicht um Planung im eigentlichen Sinne, sondern um das Bereitstellen der benötigten Informationen. Mit solchen Systemen lassen sich folglich bei beschränkten Ressourcen nur schwer funktionierende Produktionspläne erstellen.[40]

Obwohl ein bedeutender Kritikpunkt an MRP-Systemen in der Literatur das Nicht-Berücksichtigen begrenzter Kapazitäten ist, stellen MRP-Systeme weiterhin eine unverzichtbare Basis dar, auf der gut aufgebaut werden kann.

> "*It may seem an ultra-refinement to specify every nut, washer and cotter pin, but it must be borne in mind that the right quantities have to be learnt at some stage of production for issuing the details to the fitters and erectors.*"[41]

Um die Schwächen des klassischen MRP-System zu beseitigen, wurde in den 80er Jahren das MRP-II-System entwickelt. Damit es besser vom klassischem System unterschieden werden konnte, wurde es dann auch bekannt als *Manufacturing Resource Planning.*[42] Der für diese Arbeit relevanteste Unterschied zum klassischen MRP-System ist die Einführung eines Kapazitätsbelastungsausgleichs. Das Gesamtangebot an Kapazitäten wird ermittelt und Kapazitätsüber- und -unterschreitungen werden dargestellt. Es ist dann möglich durch iterative Verfahren, manuell Aufträge zu verschieben.[43] Das MRP-II-System informiert also darüber, wenn bei Überschreiten der vorhandenen Kapazitäten, der Produktionsplan nicht umgesetzt werden kann, jedoch war es das auch schon. Dadurch, dass die Anpassung des Produktionsplans manuell durchgeführt werden muss, werden deren Auswirkungen nur schwer abschätzbar und können vom Programm nur unzureichend berücksichtigt werden.[44]

Es lässt sich also zusammenfassen, dass das MRP-II-System zwar eine verbesserte Unterstützung für die Produktionsplanung darstellt, jedoch die realen Gegebenheiten noch immer unzureichend bis gar nicht berücksichtigt, wodurch in den meisten Fällen weiterhin unzulässige Produktionspläne erstellt werden, die manuell angepasst werden müssen. Zusätzlich fehlt hier immer noch der Optimierungsgedanke. Die einzige Optimierung findet manuell

[39] Dickmann, 2015, S.524f
[40] vgl. Reusch, 2006, S.27
[41] Elbourne, E. D. (1918). Factory Administration and Accounts. 3rd ed. London: The Library Press.
[42] vgl. Robert Jacobs und Ted Weston, 2007, S.359f
[43] vgl. Dickmann, 2015, S.525
[44] vgl. Reusch, 2006, S.30f

statt und selbst hier können die Auswirkungen, wegen mangelnder Berücksichtigung von Restriktionen, nur unzureichend vom Programm überprüft werden.

Die MRP-Systeme sind somit in der Praxis unzureichend, was eine Umfrage zeigt, in der 90% der Befragten (in Logistik, Einkauf und Produktion) angaben, neben MRP-Systemen zusätzliche Anwendungen zu benötigen. Manche Unternehmen gaben an, bis zu 250 solcher Zusatzprogramme zu nützen.[45] Dabei sind die Standardlösungen der großen MRP-Anbieter unflexibel und aufwändig in ihrer Anpassung und Pflege. Eine Änderung seitens der MRP-Hersteller ist oft mit hohen Kosten verbunden, nicht erwünscht oder auch einfach nicht möglich.[46]

Ein letzter interessanter Aspekt der MRP-Systeme ist, dass diese in der Regel auf Bedarfsprognosen basieren und somit nach dem Push-Prinzip funktionieren.[47] Sucht man im Buch von Klug über *Logistikmanagement in der Automobilindustrie*[48] nach dem Begriff MRP, findet man nur einen Treffer, der als Gegenbeispiel zum Pull-Prinzip dient. Das Buch von Willibald über *Neue Wege in der Automobilindustrie*[49] weist sogar keinen einzigen Treffer auf. MRP-Systeme in ihrer üblichen Anwendung scheinen für auftragsorientierte Unternehmen nicht unbedingt geeignet zu sein.

3.2 ERP-Systeme

MRP-Systeme waren zwar eine notwendige und wichtige Entwicklung, mit der Zeit wurde allerdings deutlich dass diese nicht ausreichend waren. Auf den MRP-Systemen aufbauend, entstanden die *Enterprise Resource Planning* (ERP) Systeme. Der Ausdruck ERP wurde in den frühen 90er Jahren durch die *Gartner Group* geprägt. Der Hauptgedanke war es, ein System zu entwickeln, das alle Bereiche eines Unternehmens miteinander verbindet.[50] Es lässt sich bereits aus dem Namen schließen, dass die Ressourcen im gesamten Unternehmen betrachtet werden, wodurch die Prozesse entlang der gesamten internen Supply Chain ganzheitlich betrachtet werden. ERP Systeme bestehen aus diversen Modulen, zu denen unter anderem die Bereiche der Finanz- und Buchhaltung, Personalwesen, Produktionsplanung- und -steuerung sowie Logistik gehören. Jedes Modul ist für sich ein eigenes Programm und kann somit auch unabhängig von den anderen Modulen betrieben werden. Allerdings ist es oft ratsam ein komplettes und einheitliches ERP-System zu benutzen, um Probleme zwischen den einzelnen Modulen zu vermeiden.[51]

[45] Lepros GmbH. (2007). *Fakten zu Lean* Umfrage 2007. Grafing b. München: lepros GmbH.
[46] vgl. Dickmann, 2015, S.612
[47] Zur Erinnerung, das TPS basiert auf dem Pull-Prinzip und ist somit Auftragsorientiert.
[48] Klug, Florian. (2010). *Logistikmanagement in der Automobilindustrie: Grundlagen der Logistik im Automobilbau*, Berlin, Springer Verlag.
[49] Günthner, Willibald A. (2007). *Neue Wege in der Automobillogistik: die Vision der Supra-Adaptivität*, Berlin, Springer Verlag.
[50] vgl. Robert Jacobs und Ted Weston 2007, S.361
[51] vgl. Wannenwetsch 2014, S.458

Der bemerkenswerte Vorteil, den ERP-Systeme mit sich bringen, ist ein Echtzeitinformationsfluss innerhalb des gesamten Unternehmens. Zusätzlich fallen, durch eine zentrale Speicherung aller Daten, redundante Informationen weg.[52] Obwohl ERP-Systeme eine beachtliche Verbesserung für Unternehmen mit sich bringen, gibt es keine für diese Arbeit relevanten Neuerungen. Das Modul der Produktionsplanung und -steuerung kann dem MRP-II-Systemen gleich gesetzt werden. Die Umsetzbarkeit von Produktionsplänen ist also weiterhin ein großer Mangel bei ERP-Systemen. Durch die ERP-Systeme ist es jedoch möglich *Supply Chain Management* (SCM) Systeme effektiv einzusetzen, die den Informationsfluss entlang der gesamten Supply-Chain gewährleisten.[53]

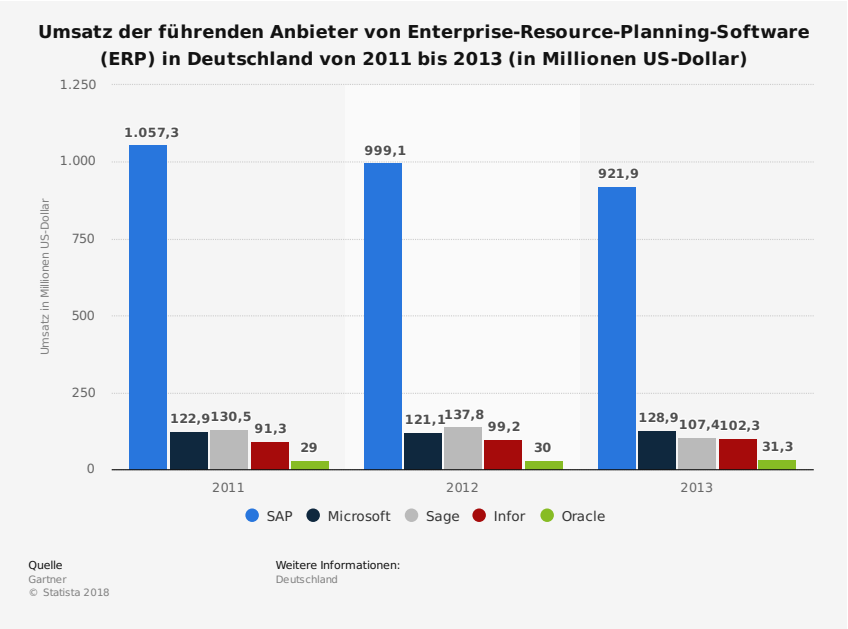

Abbildung 3.1: Umsatz der führenden ERP-Anbieter in Deutschland (2011-2013)

Einer der bekanntesten ERP-Anbieter ist SAP, der in Deutschland mit großem Vorsprung Marktführer ist (siehe Abbildung 3.1[54]). Aber auch weltweit ist SAP Marktführer und kann sich vor Konkurrenten wie z.B. FIS Global, Oracle, Infor oder Microsoft behaupten.[55]

3.3 APS-Systeme

Um die Schwächen der ERP-Systeme zu beheben, wurden die sogenannten *Advanced Planning and Scheduling* (APS) Systeme eingeführt. APS-Systeme sind Erweiterungen der ERP-Systeme und stimmen die Aktivitäten entlang der gesamten Sypply Chain synchron aufeinander ab. Ziel ist es, ERP-Systeme durch Module zu erweitern, die die

[52] vgl. Zheng u. a., 2000, S.88ff
[53] vgl. Dickmann 2015, S.525f
[54] https://de.statista.com/statistik/daten/studie/262273/umfrage/umsaetze-der-anbieter-von-erp-software-in-deutschland/
[55] https://de.statista.com/statistik/daten/studie/262342/umfrage/marktanteile-der-anbieter-von-erp-software-weltweit/; Statistik aus dem Jahre 2015

fehlenden Planungsfunktionalitäten umsetzen.[56] Die Grundidee basiert auf dem Prinzip der hierarchischen Planung, die in der von *Hax und Meal* (1975)[57] verfassten Arbeit entspringt. In dieser Publikation geht es primär darum, dass die Restriktionen der verschiedenen Planungsebenen miteinander verkoppelt sein müssen. Eine Terminplanung der einzelnen Aufträgen sollte zum Beispiel schon die Kapazitätsrestriktionen der Produktionsplanung berücksichtigen. Durch die geeignete Abstimmung der Planungsebenen kann schließlich eine optimale Gesamtlösung erzielt werden.[58]

Die hierarchischen Planung ist beispielhaft bei der Erstellung eines Semesterplans im Masterstudium anwendbar. Die Grundbasis stellen die insgesamt 30 zu absolvierenden ECTS (European Credit Transfer System) dar. Der erste Schritt besteht also darin, eine bestimmte Anzahl an Lehrveranstaltungen zu suchen, die ingesamt 30 ECTS ergeben. Die daraus folgende Liste der Lehrveranstaltungen kann mit einem theoretischen Produktionsplan verglichen werden. Jede Lehrveranstaltung hat vorgegebene Termine für Vorlesungen und Prüfungen, sowie auch jeder Auftrag in einer Produktion Start- und Endzeiten hat. Das Problem besteht darin, dass sich die Termine der Vorlesungen und Prüfungen überschneiden können bzw. es aus anderen Gründen nicht möglich ist, zur vorgeschrieben Zeit anwesend zu sein. Der theoretische Semesterplan wäre in diesem Fall in der Praxis nicht umsetzbar, sowie der Produktionsplan von MRP-II-Systemen gegebenenfalls nicht realisiert werden kann. Das Ziel der hierarchischen Planung wäre es, die einzelnen Termine aller Lehrveranstaltungen sowie alle übrigen Restriktionen bereits in der ersten Planungsphase zu berücksichtigen, um letztendlich einen realisierbaren Semesterplan erstellen zu können.

Zusammengefasst kann man also sagen, dass APS-Systeme die realen Gegebenheiten in allen Planungsphasen erfolgreicher berücksichtigen und letztendlich auch der Optimierungsgedanke besser umgesetzt werden kann. Die für diese Arbeit relevanteste Neuerung in APS-Systemen ist der Einsatz von Lösungsansätzen, wie Heuristiken bzw. Metaheuristiken,[59] wodurch nicht nur Restriktionen besser berücksichtigt, sondern auch Produktionspläne optimiert werden können.

3.4 APS Beispiel: SAP APO

Da, wie bereits erwähnt, SAP seit langem einer der führenden ERP-Anbieter ist, wird kurz auf das APS-System von SAP eingegangen, das unter dem Namen *Advanced Planner and Optimizer* (APO) bekannt ist.[60] Abbildung 3.2 gibt einen Überblick über die fünf

[56] vgl. Wannenwetsch 2014, S.490f
[57] Hax, A. C. / Meal, H. C. (1975): Hierarchical Integration of Production Planning and Scheduling, in: Geisler, M. A. (Hrsg.): Logistics, TIMS Studies in Management Science, Bd. I, Amsterdam, S. 53-69.
[58] vgl. Gebhard und Kuhn 2009, S.5
[59] vgl. Stadtler 2005, S.578
[60] Andere bekannte APS-Anbieter sind z.B. i2 Technologies, Manugistics oder J.D. Edwards.

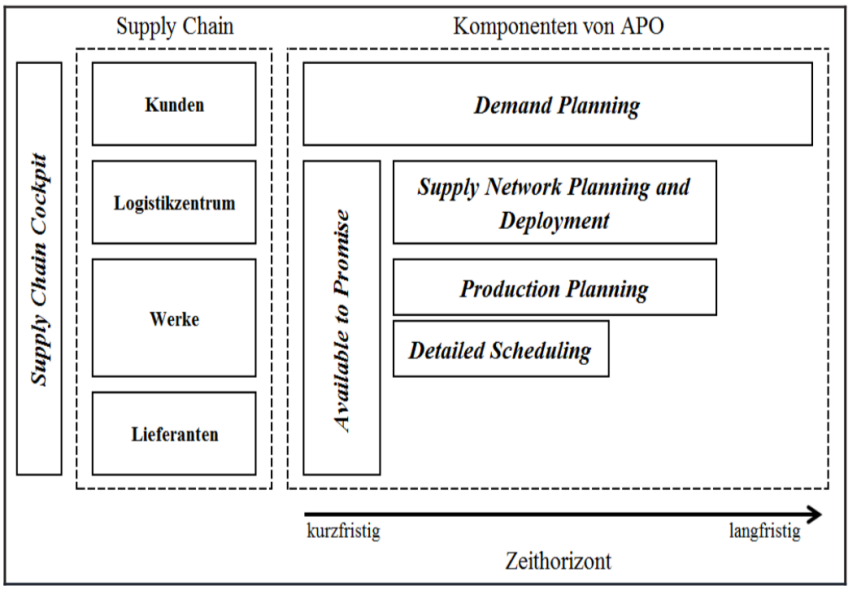

Abbildung 3.2: APO Module(Wannenwetsch, 2014, S.493)

Module, die im APO vorhanden sind:

- Supply Chain Cockpit (SCC),
- Demand Planning (DP),
- Supply Network Planning and Deployment (SNPD),
- Production Planning and Detailed Scheduling (PP/DS),
- Available to Promise (ATP),

wobei in dieser Arbeit nur auf das Modul *Production Planning and Detailed Scheduling* eingegangen wird.

APO setzt das ERP-System vom SAP voraus, das als Grundgerüst des Gesamtsystems gesehen werden kann (was wiederum die Notwendigkeit der MRP-Systeme als Grundlage unterstreicht). Das PP/DS ermöglicht durch die Berücksichtigung von Restriktionen und dem daraus entstehenden präzisen Produktionsplan eine sofortige Reaktion auf sich ändernde Marktbedingungen. Hierbei werden Aufträge bei ständiger Optimierung des Ressourceneinsatzes in ihrer Reihenfolge geplant. Folglich ist eine realistische, also umsetzbare, Lierferterminbestimmung möglich.[61]

Das PP/DS Modul in SAP bedient sich zur Erstellung und Optimierung von Produktionsplänen mehrerer Algorithmen, wie z.B. des Genetischen Algorithmus (GA), Multi-Level Planungs-Heuristiken und manuelle Planungs-Heuristiken (inclusive drag and drop). Wobei auch selbst erstellte Algorithmen eingespielt werden können, die vielleicht die spezifischen Restriktionen im Unternehmen besser wiederspiegeln, falls die Standard Algorithmen nicht ausreichen. Der Benutzer kann also sowohl eigene als auch bereits integrierte Algorithmen

[61] vgl. Wannenwetsch 2014, S.495

benutzen bzw. diese kombinieren, um einen bestmöglichen Produktionsplan zu erstellen. Zu den berücksichtigten Restriktionen zählen: Liefertermine, Lagerkapazität oder Reihenfolgen abhängige Rüstkosten. Die Kostenfunktion versucht Ziele wie Rüstkosten, Lagerkosten oder Verspätungen zu minimieren. Welche Kosten optimiert werden sollen, kann der Benutzer selbst angegeben.[62] Wie der Genetische Algorithmus genau programmiert ist, lässt sich leider nicht herausfinden, was sicherlich auch ein Betriebsgeheimnis von SAP ist.

Zusammenfassend lässt sich also sagen, dass der APO Restriktionen bereits sehr gut beim Erstellen von Produktionsplänen berücksichtigt. Demzufolge können realisierbare Produktionspläne automatisch erstellt werden, die anschließend auch optimiert werden können. Dadurch, dass sich SAP an ein großes Spektrum an Kunden richtet, dürfte allerdings z.B. der integrierte Standard GA sehr allgemein programmiert sein, wodurch spezifische Restriktionen einzelner Unternehmen nicht berücksichtigen werden können.[63] Dies zeigt sich unter anderem am Beispiel von Daimler, Mercedes und Chrysler, die teilweise für die Auftragseinplanung das individuelle System ASF benutzen, das die individuellen Anforderungen besser abdeckt, als Standardprodukte.[64] Nichtsdestotrotz bietet, wie bereits erwähnt, das PP/DS Modul die Möglichkeit, eigene Algorithmen zu benutzen, wodurch jedes Unternehmen spezifische Restriktionen in der Produktionsplanung und -optimierung berücksichtigen kann. Genau in diesem Punkt kommt man wieder zur eigentlichen Zielsetzung dieser Arbeit, die die Entwicklung einer Metaheuristik ist, die als Ziel das Erstellen und Optimieren von Produktionsplänen eines kunststoffverarbeitenden Unternehmens hat, mit möglichst genauer Einhaltung aller spezifischen Restriktionen. Diese Metaheuristik könnte dann z.B. im Falle von SAP im APO-PP/DS Modul eingespielt und verwendet werden.

Neben SAP sind bekannte APS Anbieter unter anderem: [65]

- **IBM**: vor allem eingesetzt in der Papier Industrie.
- **AMD**: eingesetzt in der Halbleiterindustrie.
- **ASPROVA**: der bekannteste Anbieter in Japan (z.B. verwendet von Canon, Honda und Toyota), der aber auch weltweit zum Einsatz kommt.
- **Preactor**: ein britischer APS Anbieter, der von Siemens AG aufgekauft wurde und nun Teil von Siemens Industry Automation Division ist. Kommt ebenfalls weltweit zum Einsatz.
- **ORTEMS**: ein französischer APS Anbieter, der in mehr als 60 Ländern eingesetzt wird und in verschiedenen Industrien verwendet werden kann (unter anderen bei General Electric, Nestle und Peugeot/Citroen).

[62] vgl. Pinedo, 2016, S.511ff
[63] Hierbei handelt es sich um eine Vermutung, da der genaue Quellcode nicht bekannt ist.
[64] vgl. Fritzsche, 2009, S.20f
[65] vgl. Pinedo, 2016, S.509ff

4 Kapitel 4

Methoden zur Optimierung von Produktionsplänen

Im letzen Kapitel wurde genauer auf die Entstehung der EDV-Systeme eingegangen, die in Unternehmen heutzutage üblich sind, um generell alle Abläufe koordinieren und optimieren zu können, nämlich einer Kombination aus MRP-, ERP- und APS-Systemen. Der Vollständigkeit halber sei erwähnt, dass es noch viele andere Systeme gibt, die allerdings meist auf den genannten Systemen aufbauen oder letztere ergänzen, da diese in gewisser Weise als Grundbausteine dienen.

Der nächste logische Schritt wäre es nun, die Methoden genauer zu untersuchen, die in APS-Systemen eingesetzt werden, um Produktionspläne zu erstellen und zu optimieren. Diese Methoden sind unter anderen Heuristiken und Metaheuristiken. Jedoch bevor genauer auf diese Methoden eingegangen wird, muss genauer auf den Begriff der Komplexitätstheorie und der Modellbildung eingegangen werden.

4.1 Komplexitätstheorie

Wie der Name schon sagt, geht es hier um die Definition von Komplexität bzw. wann etwas als komplex angesehen wird. Komplexität mag erstmals sehr subjektiv wirken, ausgehend davon, dass diese von der Sicht eines wertenden Subjekts abhängt. Allerdings betrachtet die Komplexitätstheorie die Komplexität eines Problems aus der Sicht eines Computers. Die Frage ist also: Was ist für einen Computer komplex?

4.1.1 Definition und Einordnung

Um die Komplexitätstheorie verstehen zu können, ist es also notwendig sich in einen Computer hineinzuversetzen. Computer sind keine Zaubermittel, sondern führen im Endeffekt nur durch den Menschen definierte Schritte aus. Hinter jedem Schritt steckt eine genau definierte Vorgehensweise, die vom Menschen vorgegeben wird. In dieser Hinsicht kann der Computer nichts, was der Mensch nicht kann. Was den Computer so unglaublich nützlich macht, ist, die Schnelligkeit mit welcher definierte Prozesse umgesetzt werden können. Diese Geschwindigkeit ist für Menschen unerreichbar, aber auch für Computer nicht unbegrenzt.

4.1.1.1 Definition Allgemein

Einfach ausgedrückt, wird in der Komplexitätstheorie die Komplexität eines Problems über die benötigte Anzahl an Rechenschritten gemessen, die ein Computer braucht, um ein Problem zu lösen. Welche Rechenschritte wann ausgeführt werden, wird über sogenannte Algorithmen vorgegeben. Die Anzahl an Rechenschritten hängt natürlich nicht nur von der Komplexität des Problems, sondern auch von der Größe ab. Ein Maß für die Größe eines Problems in der Produktion ist z.B. die Anzahl an Aufträgen in einem System und gegebenenfalls auch die Anzahl an vorhandenen Maschinen, mit denen diese Aufträge bearbeitet werden können.[66] Sucht man z.B. nach einer optimalen Reihenfolge für 5 Aufträge auf einer Maschine, sind im ungünstigsten Fall $5! = 120$ Möglichkeiten durchzurechnen. Für 10 Aufträge wären das schon $10! = 3.628.800$ Möglichkeiten, was für einen Menschen unvorstellbar wäre, für einen Computer aber weiterhin schnell zu lösen ist. Die Anzahl an Rechenschritten wächst also mit der Größe des Problems, die relativ einfach zu bestimmen ist. Jedoch wächst die Anzahl an Rechenschritten ebenfalls mit der Komplexität des Problems an, die um einiges schwieriger zu definieren ist, als die Größe des Problems. Grundsätzlich ist ein Problem nur so komplex ist, wie der effektivste Algorithmus zum Lösen dieses Problems, wobei dieser nicht unbedingt bekannt sein muss.[67] Das bedeutet, dass ein Problem schwieriger erscheinen kann, als es tatsächlich ist, weil bisher noch kein effektiver Algorithmus gefunden wurde.

Um diesen Gedanken besser verstehen zu können, nehmen wir als simples Beispiel das Schneiden von Gras in einem Garten. Unser Problem ist, dass das Gras schon zu lange ist und deshalb geschnitten werden muss. Doch wie komplex ist das Schneiden von Gras wirklich? Aus der Sicht der Komplexitätstheorie ist jedes Problem eben nur so schwer, wie die Werkzeuge, die einem zur Verfügung stehen. Und so ist es auch mit dem schneiden von Gras. Steht uns nur eine Nagelschere zur Verfügung, scheint das gegebene Problem unmöglich zu sein. Steht einem jedoch ein Rasenmäher zur Verfügung, kann man schon fast gar nicht mehr von einem Problem reden. In der Komplexitätstheorie sind die Werkzeuge zum Lösen von Problemen Algorithmen. Ein Problem wird also nur als komplex angesehen, solange es keinen effektiven Algorithmus zum Lösen dieses Problems gibt.

Die Komplexität eines Problems wird in der Komplexitätstheorie in zwei Kategorien unterteilt: P-Schwere Probleme und NP-Schwere Probleme.

4.1.1.2 P-Schwere Problem

P-Schwere Probleme sind eigentlich “einfache” Probleme, die durch bekannte Algorithmen effektiv gelöst werden können. Das P steht für *polynomial* und bedeutet, dass die Anzahl

[66] vgl. März und Krug 2011, S.37
[67] vgl. Falkenauer 1998, S.6

der Rechenschritte mit der Größe des Problems "nur" polynomiell zunimmt.[68] Einfach ausgedrückt bedeutet dass, das P-Schwere Probleme von Computern relativ schnell gelöst werden können, mehr oder weniger unabhängig von der Größe des Problems, wobei die Größe des Problems die Rechenzeit natürlich immer beeinflusst. Im Falle der P-Schweren Probleme wirkt sich die Größe des Problems jedoch nur beschränkt auf die Rechenzeit aus, weshalb auch relativ große Probleme in einer vernünftigen Zeit gelöst werden können (siehe Tabelle 4.1 n bis n^5). Mit einem Rasenmäher ist das Schneiden einer Wiese also ein P-Schweres Problem. Auch wenn die benötigte Zeit für mehrere Hektar (Größe des Problems) linear zunimmt und somit länger dauert, als bloß für einen kleinen Garten, kann selbst eine größere Fläche in einer vernünftigen Zeit gemäht werden.

4.1.1.3 NP-Schwere Probleme

NP-Schwere Probleme stellen die wirklich schweren Probleme dar, bei denen die Anzahl der Rechenschritte mit der Größe des Problems exponentiell zunimmt. In anderen Worten ausgedrückt lässt sich sagen, dass NP-Schwere Probleme sich nicht effizient lösen lassen.[69] In Tabelle 4.1 ist sehr gut ersichtlich, was ein exponentielles Wachstum bedeutet. Die Komplexität 2^n ist zwar für kleine Problemgrößen noch schnell zu lösen, nimmt aber mit der Größe des Problems exponentiell zu, was dazu führt, dass für $n = 50$ das Problem nicht mehr in angemessener Zeit gelöst werden kann und für $n = 60$ sogar bis zu 366 Jahrhunderte (!) dauern kann. Mit einer Nagelschere kann z.B. das Schneiden einer Wiese als NP-Schweres Problem gesehen werden. Ein paar Grashalme stellen kein Problem dar und sind vielleicht sogar schneller geschnitten als mit dem Rasenmäher, allerdings wird das Schneiden mehrerer hundert Grashalme schon zur Qual und die Bewältigung eines Hektars Wiese ist schon gar nicht mehr vorstellbar.

	Problemgröße n					
Komplexität	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 60$
n	0.00001 s	0.00002 s	0.00003 s	0.00004 s	0.00005 s	0.00006 s
n^2	0.0001 s	0.0004 s	0.0009 s	0.0016 s	0.0025 s	0.0036 s
n^3	0.001 s	0.008 s	0.027 s	0.064 s	0.125 s	0.216 s
n^5	0.1 s	0.32 s	24.3 s	1.7 min	5.2 min	13 min
2^n	0.001 s	1 s	17.9 min	12.7 $Tage$	35.7 $Jahre$	366 $Jahrh.$
3^n	0.059 s	58 min	6.5 $Jahre$	3855 $Jahrh.$	2×10^8 $Jahrh.$	1.3×10^{13} $Jahrh.$

Tabelle 4.1: Evolution der Laufzeit für unterschiedliche Komplexitäten (vgl. Garey und Johnson, 1979, S.7)

Jetzt könnte man natürlich argumentieren, dass die Tabelle 4.1 aus dem Jahre 1979 ist und die Computer heutzutage um ein Vielfaches schneller sind und auch in der Zukunft stetig schneller werden. Das große Problem dabei ist jedoch, dass z.B. ein Problem der

[68] vgl. März und Krug 2011, S.38f
[69] vgl. März und Krug 2011, S.38

Komplexität 2^n mit einem Computer, der zwar 1000 Mal schneller ist, nicht 1000 Mal größere Probleme gelöst werden können (was jedoch für eine lineare Komplexität sehr wohl der Fall ist). Garey und Johnson (1979) zeigen, dass ein Problem der Komplexität 2^n mit einer bestimmten Größe N_1, das aktuell in einer Stunde gelöst werden kann, mit einem 1000 Mal schnelleren Computer in der selben Zeit das gleiche Problem nur der Größe $N_1 + 9.97$ gelöst werden kann.[70] Das bedeutet z.B., dass, wenn ein Problem der Komplexität 2^n mit der Größe $N_1 = 30$, die z.B. die Anzahl der Aufträgen in einer Produktion darstellt, ein 1000 Mal schnellerer Computer in der selben Zeit nur das Problem der Größe $N_2 = 40$ lösen kann, also mit 10 Aufträgen mehr. Demzufolge kann das Problem der NP-Schweren Probleme auch in naher Zukunft mit schnelleren Computer nicht gelöst werden.

4.1.2 Schlussfolgerung der Komplexitätstheorie für diese Arbeit

Auch wenn die Komplexitätstheorie in den letzten Unterabschnitten nur vereinfacht dargestellt ist, reicht es aus, um eine wichtige Aussage für diese Arbeit mitzunehmen: Es gibt (NP-Schwere) Probleme, die so schwer sind, dass es nicht möglich ist einen effektiven Algorithmus zu finden, der das Problem optimal in angemessener Zeit lösen kann. Diese Information ist unglaublich wertvoll und hilft bei der Herangehensweise an ein Problem.

Ist bekannt, dass ein Problem zu den NP-Schweren Problemen gehört, sollte somit der Suche nach einem exakten Algorithmus nur beschränkt Aufmerksamkeit geschenkt werden. Es sollte viel eher nach Algorithmen gesucht werden, die verschiedene Spezialfälle des betrachteten Problems effektiv und möglicherweise optimal lösen können. Ebenfalls kann nach Algorithmen gesucht werden, die das Problem nicht unbedingt optimal lösen, aber dafür in angemessener Zeit ein zufriedenstellendes Ergebnis liefern.[71]

4.2 Definition Algorithmus

Da die Komplexität eines Problems von der Effektivität eines Algorithmus abhängt, wird kurz auf die Definition von Algorithmen eingegangen. In der Literatur gibt es viele verschiedene Definitionen, was unter einem Algorithmus verstanden werden kann, wobei folgende Definition eine gute Zusammenfassung der verschiedenen Begriffserklärungen darstellt:

> *"Ein Algorithmus ist eine vollständige, präzise und in einer Notation oder Sprache mit exakter Definition abgefasste, endliche Beschreibung eines schrittweisen Problemlösungsverfahrens zu Ermittlung gesuchter Datenobjekte (ihrer Werte) aus gegebenen Werten von Datenobjekten, in dem jeder Schritt aus*

[70] vgl. Garey und Johnson, 1979, S.8
[71] vgl. Garey und Johnson, 1979, S.3f

> *einer Anzahl ausführbarer, eindeutiger Aktionen und einer Angabe über den nächsten Schritt besteht."*[72]

Wobei mit Datenobjekten Konstanten und Variablen gemeint sind.

Das Wort *Problemlösungsverfahren* fasst die Funktion eines Algorithmus gut zusammen. Vereinfacht, kann ein Algorithmus als eine wohldefinierte Rechenvorschrift gesehen werden, die eine Größe oder eine Menge von Größen (Input) verwendet und eine Größe oder eine Menge von Größen als Output erzeugt.[73] Somit ist ein Algorithmus eine Folge von Rechenschritten, die einen Input in einen Output umwandeln.

In einem anderen Kontext kann ein Algorithmus auch als eine Art Rezept gesehen werden, das vorgibt, was unter welchen Umständen gemacht werden soll, um ein gewisses Ziel zu erreichen. Ein Kochrezept kann z.B. mit einem Algorithmus verglichen werden: "Zuerst das Wasser zum Kochen bringen. Wenn das Wasser kocht, folgende Zutaten hinzufügen ...". Der Input sind die Zutaten und der Output ist ein fertiges Gericht. Der große Unterschied zwischen einem Algorithmus und einem Kochrezept liegt jedoch darin, dass ein Algorithmus für jede mögliche eintretende Situation eine Vorgehensweise parat haben muss, also um einiges genauer sein muss. In einem Kochrezept wird man z.B. nie eine Vorgehensweise finden, die umgesetzt werden muss, wenn das Wasser den Siedepunkt nicht erreicht. In diesem Fall wäre der Kochrezept-Algorithmus nicht anwendbar und würde unendlich laufen.[74]

Um eine Verbindung zur Komplexitätstheorie herzustellen, kann gesagt werden, dass je nach Größe des Inputs und Effektivität des Algorithmus, der Output mehr oder weniger lange auf sich warten lässt. Das übliche Maß für Effizienz ist in diesem Fall eben die Geschwindigkeit.[75] Ein Algorithmus ist also dann effizient, wenn die benötigte Zeit von Input bis Output mit der Größe des Inputs nur polynomiell zunimmt. Wobei Unterschiede in der Effizienz eines Algorithmus viel erheblicher sein können, als die durch Hardware und Software bedingten Unterschiede.[76] Dieser Aspekt ist wiederum durch den Gedanken der Komplexitätstheorie gut wiedergegeben, da, wenn ein effektiver Algorithmus für ein Problem bekannt ist, dieses Problem dementsprechend zu den P-Schweren Problemen gehört und somit relativ schnell gelöst werden kann. Die Nagelschere und der Rasenmäher sind beide Werkzeuge (Algorithmen), die das Problem des zu langen Grases lösen, der Rasenmäher ist allerdings um einiges effizienter, vor allem dann, wenn der Rasen eine große Fläche einnimmt. Es würde keinen so großen Unterschied machen, wenn statt der Nagelschere eine übliche Küchenschere (bessere Hardware) benutzt wird, da das Schneiden einer großen Wiese noch immer zu ineffizient wäre. Erst durch den Einsatz eines Rasenmähers (effektiver Algorithmus) ist es möglich eine größere Fläche zu mähen.

[72] Pomberger und Dobler, 2008, S.33
[73] vgl. Cormen u. a. 2004, S.5
[74] vgl. Falkenauer 1998, S.3
[75] vgl. Cormen u. a., 2004, S.9
[76] vgl. Cormen u. a. 2004, S.11

Aber nicht jeder NP-Schwere Algorithmus ist ein schlechter Algorithmus. Ein bekanntes Beispiel ist der Simplex-Algorithmus, der zur Lösung von linearen Optimierungsproblemen benutzt wird. Es ist bekannt, dass der Simplex-Algorithmus exponentieller Natur ist und trotzdem wird er in der Praxis erfolgreich eingesetzt.[77]

4.3 Definition Modellbildung

Algorithmen basieren üblicherweise auf Modellen, die die Realität nicht eins zu eins darstellen. Aber weshalb ist eine Modellbildung überhaupt nötig? Um das zu verstehen, muss kurz auf den Prozess der Modellbildung genauer eingegangen werden.

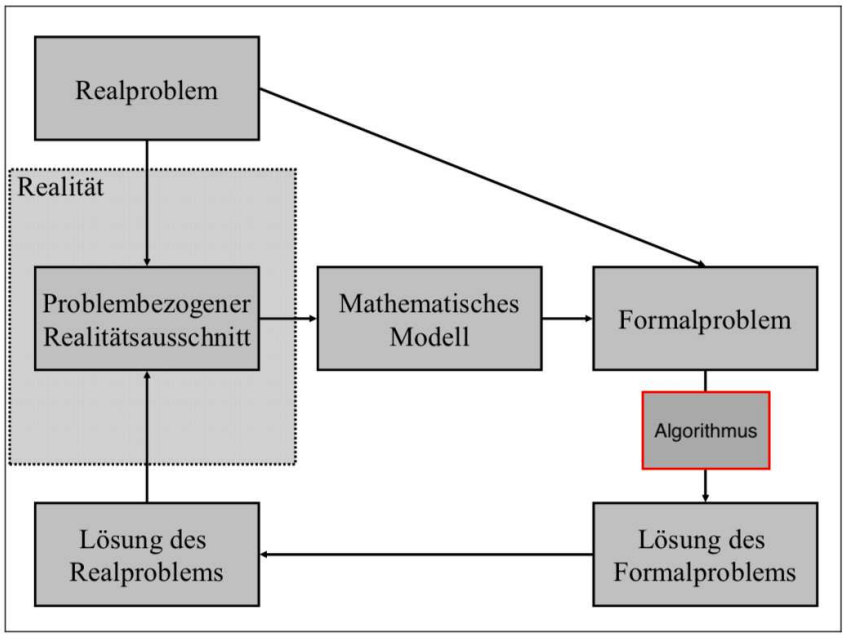

Abbildung 4.1: Schematische Darstellung des Modellierungsprozesses (vgl. Reusch, 2006, S.13) (angelehnt an Müller-Merbach, 1973, S.14)

Die Voraussetzung für eine Modellbildung ist die Nachbildung der Realität in einem mathematischen Modell. Da die Realität aber nie vollständig in einem mathematischen Modell erfasst werden kann, wird nur der Teil der Realität ausgeschnitten, der sich in einem Modell nachbilden lässt. Ein Modell ist also eine idealisierende Beschreibung der Realität, in welcher die Realität nur in Bezug auf die problemrelevanten Tatbestände wiedergegeben wird. Überträgt man die Fragestellung des Realproblems auf das mathematische Modell, entsteht das sogenannte Formalproblem (siehe Abbildung 4.1). Das Formalproblem gilt es nun mit mathematischen Methoden bzw. Algorithmen zu lösen. Die Lösung des formalen Problems kann dann auf die Realität zurückgeführt werden und gibt einen Entscheidungsvorschlag für das reale Problem. Die größte Herausforderung beim Erstellen des Modells besteht darin, den richtigen Realitätsausschnitt zu betrachten. Fehlen im Realitätsausschnitt relevante Aspekte des Problems oder gehen wichtige Beziehungen zu benachbarten

[77] vgl. Falkenauer 1998, S.10

Realitätsbereichen verloren, kann die Lösung des Formalproblems unbrauchbar sein. Ein dem Problem nicht entsprechender Realitätsausschnitt kann grundsätzlich aus drei Gründen entstehen. Einerseits kann es sein, dass die Kenntnis über die Realität unzureichend ist und dadurch die Struktur und die Zusammenhänge nicht richtig wiedergegeben werden können. Andererseits kann es aber auch sein, dass im Hinblick auf die Lösbarkeit des realen Problems Faktoren, die sehr wohl einen Einfluss auf das Ergebnis haben, wegen begrenzter rechnerischer Möglichkeiten bewusst ignoriert werden. Letztendlich kann es auch sein, dass die Person, die das Modell erstellt, die benötigten mathematischen Instrumente nicht kennt und folglich den Realitätsausschnitt so vornimmt, dass ein Modell entsteht, das sie mit den ihr bekannten Methoden lösen kann.[78]

Alle drei genannten Gründe für einen ungenügenden Realitätsausschnitt sind für diese Arbeit von höchster Relevanz. Erstens müssen natürlich die Struktur und die Zusammenhänge des vorliegenden Problems genau verstanden werden, um auch wirklich die relevanten Aspekte zu berücksichtigen und ins Modell mit aufzunehmen. Wird ein ausschlaggebender Aspekt nicht berücksichtigt, können die Resultate vollkommen unbrauchbar sein. Genauso wichtig ist es, die Erkenntnisse aus der Komplexitätstheorie zu nutzen, um sich der Grenzen der Algorithmen bewusst zu sein und gegebenenfalls alternative Lösungen zu suchen, die möglicherweise keine optimale Lösung liefern, aber dafür eine zufriedenstellende in einer annehmbaren Zeit. Und letztlich muss durch eine gründliche Literaturrecherche eine möglichst vollständige Liste der heutzutage wichtigsten Methoden bereitstellen werden.

Zusammengefasst lässt sich sagen, dass der Modellierungsprozess als ein kontinuierlicher Verbesserungsprozess gesehen werden kann (siehe untere Schleife in Abbildung 4.1). Ist nach Rückführung der Lösung des Formalproblems auf das Realproblem die Lösung des Realproblems nicht zufriedenstellend, muss der Modellierungsprozess neu begonnen werden. Es kann z.B. in einem neuen Anlauf zielführend sein, den Realitätsausschnitt neu zu definieren, wodurch sich auch das Formalproblem ändert oder es können neue bzw. angepasste Algorithmen zum Lösen des Formalproblems angewandt werden.

4.4 Scheduling Modelle

Bevor ein Algorithmus für ein Scheduling Problem erstellt werden kann, muss in erster Linie ein Modell erstellt werden, das die relevanten Aspekte in einem Realitätsausschnitt darstellt, sodass das Realproblem effektiv gelöst werden kann. Doch was genau ist Scheduling überhaupt? Scheduling kann als ein entscheidungsfindender Prozess gesehen werden, der sich mit der Zuteilung von beschränkten Ressourcen zu Aufgaben bzw. Aufträgen über einem bestimmten Zeithorizont beschäftigt. Dieser Prozess findet regelmäßig statt und hat die Optimierung eines Kriteriums bzw. mehrerer Kriterien zum Ziel.[79] Aus der Sicht der

[78] vgl. Müller-Merbach, 1973, S.14f
[79] vgl. Pinedo, 2016, S.1

Produktion ist Scheduling somit das Festlegen der Startzeiten von Aufträgen, die darauf warten, auf einer oder mehreren Maschinen (Ressourcen) bearbeitet zu werden.[80] Im Falle dieser Arbeit sind die Ressourcen Maschinen, Werkzeuge und Rüstpersonal, die Aufträgen von vorhandenen Bestellungen zugeteilt werden. Ziel ist es, diese Auftragszuteilung (wann welcher Auftrag von welchen Ressourcen bearbeitet wird) so optimal wie möglich zu gestalten, damit mehrere Kriterien wie z.B. Rüstkosten oder Verspätungen optimiert werden.

Grundsätzlich können Scheduling Modelle in zwei Kategorien eingeteilt werden: Stochastische und deterministische Modelle, wobei sich diese Arbeit mit deterministischen Scheduling Modellen beschäftigt. Der Vollständigkeit halber wird aber dennoch kurz auf die Definition von stochastischen Modellen eingegangen.

4.4.1 Stochastische Modelle

Das Umfeld der Produktion ist mit vielen Unsicherheiten und Zufällen verbunden. Unsicherheiten, die erhebliche Auswirkungen haben können, sind z.B. Maschinenausfälle, unvorhergesehene Aufträge mit hoher Priorität oder auch variable Bearbeitungszeiten. Ein gutes Modell sollte diese Kriterien berücksichtigen. Stochastische Modelle berücksichtigen diese Unsicherheiten indem z.B. Maschinenausfälle als eigener stochastischer Prozess gesehen werden, der definiert, wann eine Maschine verfügbar ist und wann nicht.[81] Um das Maß der Unsicherheit zu definieren, werden also Methoden der Wahrscheinlichkeitstheorie genutzt. Wie wahrscheinlich ist es, dass eine Maschine ausfällt? Wie sehr schwankt die Bearbeitungszeit eines Vorgangs auf einer Maschine bzw. wie hoch ist überhaupt die Wahrscheinlichkeit, dass die Bearbeitungszeit von der geplanten Bearbeitungszeit abweicht? Mit diesen Fragen beschäftigen sich die stochatischen Scheduling Modelle, indem Unsicherheiten in der Erstellung des Produktionsplans mit einbezogen werden.

4.4.2 Deterministische Modelle

Werden Unsicherheiten wie z.B. Maschinenausfälle nicht berücksichtigt, spricht man von deterministischen Modellen. In deterministischen Modellen sind alle Gegebenheiten mit Gewissheit bekannt, es treten also keine Unsicherheiten auf. Die Anzahl an Aufträgen im betrachteten Zeitabschnitt sind festgelegt, die Maschinen sind durchgehend einsatzbereit.

Um in der Literatur und auch darüberhinaus in der Praxis Scheduling Modelle besser vergleichen zu können, wurde in der Literatur ein Framework aufgestellt, das als Ziel eine möglichst einheitliche Definition von Scheduling Problemen hat. Dieses Framework hilft auch, Scheduling Probleme leichter als P-Schwer oder NP-Schwer einzuordnen, was in der

[80] vgl. Sarin u. a. 2010, S.1
[81] vgl. Pinedo 2016, S.245

Praxis für die Herangehensweise an ein Scheduling Problemen besonders hilfreich sein kann.

4.4.3 Definition des Scheduling Problems dieser Arbeit

Ein Scheduling Model wird in der Literatur grundsätzlich durch drei Felder definiert: [82]

$$\alpha| \ \beta \ | \ \gamma.$$

Das α-Feld beschreibt die Maschinenumgebung und beinhaltet nur einen Eintrag. Das β-Feld liefert Informationen über die Job Charakteristiken und Restriktionen und kann einen, mehrere oder auch keine Einträge beinhalten. Das γ-Feld beschreibt das zu optimierende Kriterium und beinhaltet einen oder auch mehrere Einträg. In den meisten Scheduling Problemen werden fast immer eine endliche Anzahl an Jobs und Maschinen betrachtet. Die Anzahl der Jobs ist meistens mit einem n gekennzeichnet, wohingegen die Anzahl an Maschinen durch ein m angegeben wird. Der Index j bezieht sich üblicherweise auf einen Job und der Index i auf eine Maschine. Wenn also z.B. ein Job auf einer bestimmten Maschine bearbeitet werden muss, spricht man von der Bearbeitung von Job j auf Maschine i. Ein Job (Auftrag) kann somit einen oder mehrere Bearbeitungsschritte haben. Ein Job ist dann fertig gestellt, wenn alle benötigten Bearbeitungsschritte durchgeführt wurden. Für eine genaue Beschreibung dieses Framework siehe Anhang 8.1.

4.4.3.1 Allgemeine Beschreibung des Scheduling Problems dieser Arbeit

Alle benötigten Informationen werden über ein Excel-File bereitgestellt. In diesem File sind alle noch nicht abgefertigten Bestellungen bzw. Aufträge (insgesamt $n = 714$) aufgelistet. In jeder Bestellung wird angegeben, wie viel (Menge in Stück), von welchem Artikel (Artikelnummer und Farbe), wann geliefert werden soll. Des Weiteren befinden sich im Excel-File auch alle weiteren Informationen, die für die Produktion notwendig sind.

Insgesamt gibt es 17 parallel laufende Maschinen, die in drei Maschinengruppen aufgeteilt sind. Jede Maschinengruppe hat einen eigenen Maschinenstundensatz. Für jede Artikelnummer gibt es mindestens ein Werkzeug bis mehrere Werkzeuge, die nur auf bestimmten Maschinen gerüstet werden können. Es gibt Produktfamilien, d.h. dass es Werkzeuge gibt, die mehrere Artikelnummern produzieren können. Dadurch, dass nicht jedes Werkzeug auf jeder Maschine gerüstet werden kann, kann auch nicht jeder Artikel auf jeder Maschine produziert werden.

Die Bearbeitungszeit eines Auftrags hängt vom benutzten Werkzeug ab. Die Maschine beeinflusst die Bearbeitungsgeschwindigkeit nicht. Gibt es für eine Artikelnummer mehrere

[82] vgl. Pinedo, 2016, S.13ff

Werkzeuge, können, aber müssen diese nicht unterschiedliche Bearbeitungsgeschwindigkeiten haben. Jeder Auftrag benötigt nur einen Bearbeitungsschritt und wird deshalb nur auf einer der möglichen Maschinen abgefertigt. Das Unterbrechen eines Auftrags ist nicht erwünscht. Die Bearbeitungsdauer eines Auftrags hängt dementsprechend nur von der bestellten Menge und dem benutzten Werkzeug ab.

Produziert werden kann grundsätzlich an jedem Werktag von 0-24 Uhr von Montag bis Freitag. Allerdings werden 1.2 Stunden (5%) pro Tag für Instandhaltungsarbeiten benötigt, wodurch insgesamt nur 22.8 Stunden pro Tag für die Produktion vorhanden sind. Da das Rüstpersonal nur von 6-19 Uhr anwesend ist, können Aufträge, die einen Rüstvorgang benötigen, nur innerhalb dieser Zeiten begonnen werden. Es stehen maximal vier Rüstmitarbeiter zur Verfügung. Die Rüstdauer ist nur vom Werkzeug abhängig und somit unabhängig von den Maschinen.

Es gibt Werkzeuge, die mehrere Aufträge gleichzeitig produzieren können, allerdings verlangsamt sich die Zykluszeit mit der Anzahl der Aufträge im Batch proportional (Faktor 1). Das bedeutet, dass wenn z.B. drei gleichlange Aufträge gleichzeitig produziert werden, die Zykluszeiten sich verdreifachen. Es können dadurch zwar alle drei Aufträge zum frühesten Zeitpunkt begonnen werden, fertig sind aber alle Aufträge erst zu dem Zeitpunkt, an dem der letzte der drei Aufträge fertig geworden wäre, wären alle drei Aufträge nacheinander produziert worden. Dadurch, dass der Fertigstellungszeitpunkt und die damit verbundene Lieferfähigkeit des Auftrags wichtiger ist, als der Startzeitpunkt, ist es sinnvoller, eine effektive Reihenfolge der Aufträge zu suchen, als diese in einem Batch gleichzeitig zu produzieren. Das betrachtete Unternehmen benutzt das parallele Produzieren von mehreren Aufträgen mit einem Werkzeug vor allem, um in überlasteten Perioden Teillieferungen durchführen zu können. Da dies in dieser Arbeit nicht berücksichtigt werden kann, wird folglich auf diese Funktion verzichtet.

Optimiert werden sollen folgende Kriterien:

- Die Rüstkosten haben eine hohe Priorität und sollen auf ein Minimum reduziert werden. Da die Anzahl der Rüstmitarbeiter beschränkt ist, soll ebenfalls darauf geachtet werden, ob es zu großen Wartezeiten auf Rüstmitarbeitern kommt.
- Die höchste Priorität haben die Verspätungen. Im optimierten Produktionsplan dürfen keine Verspätungen mehr auftreten.
- Die Lagerkosten sollen soweit es geht minimiert werden. Zusätzlich stehen im Lager für die betrachteten Produkte insgesamt nur 500 Palettenplätze zur Verfügung.
- Die Maschinen sollten alle möglichst gut ausgelastet sein. Sprich Maschinenstillstände sollen auf ein Minimum reduziert werden. Maschinenstillstände können dann auftreten, wenn auf ein Werkzeug oder den Rüststart gewartet wird. Warten auf den Rüststart kommt dann zustande, wenn die geplante Startzeit auf eine Uhrzeit außerhalb des Rüstintervalls (6-19 Uhr) fällt und ein Rüstvorgang notwendig ist.

In seltenen Fällen kann es vorkommen, dass alle vier Rüstmitarbeiter besetzt sind und deshalb auch auf einen Rüstmitarbeiter gewartet werden muss. Gegebenenfalls kann auch eine Kombination der drei Wartezeiten (Werkzeug, Rüststart und Rüstmitarbeiter) auftreten.

- Die Maschinenbetriebskosten sollen auf ein Minimum reduziert werden. Das bedeutet, dass die Aufträge möglichst mit dem schnelleren Werkzeug und auf der Maschine mit dem niedrigeren Stundensatz produziert werden sollten.

4.4.3.2 Beschreibung des Scheduling Problems mit dem üblichen Framework

Das α-Feld kann folgendermaßen definiert werden:

- Es werden 17 parallel laufende Maschinen betrachtet. Die Geschwindigkeit, mit der die Aufträge bearbeitet werden, ist unabhängig von der benutzen Maschine. Somit kann die Maschinenumgebung als *identical machines in parallel* (Pm) betrachtet werden. Für $m = 17$ wird also im α-Feld $P17$ angegeben.

Das β-Feld kann wie folgt beschrieben werden:

- Die Bearbeitungszeit von Aufträgen ist unabhängig von der benutzen Maschine ($p_{ij} = p_j$).
- Es gibt Werkzeuge, die mehrere Artikelnummern produzieren können, somit sind *job families* vorhanden ($fmls$).
- Es liegen fixierte Instandhaltungszeiten vor, dementsprechend werden *breakdowns* berücksichtigt ($brkdwn$).
- Jeder Auftrag kann nur auf bestimmten Maschinen bearbeitet werden (M_j).

Das γ-Feld lässt sich folgendermaßen beschreiben:

- Es wird versucht die Summe der positiven Verspätungen (T_j) bestmöglich zu reduzieren. Die Gewichtung (ω_j) hängt unter anderem von der Größe der Bestellungen ab, die sich verspätet ($\sum \omega_j T_j$).
- Ist ein Auftrag vor dem geplanten Lieferdatum fertig, muss dieser bis zum Liefertermin gelagert werden. Es wird versucht, die Lagerkosten zu reduzieren, dementsprechend wird versucht, die Summe der negativen Verspätungen (E_j) auf ein Minimum zu reduziert. Die Gewichtung (ω_j) hängt unter anderem von der Größe der Bestellung ab, die gelagert wird ($\sum \omega_j E_j$).
- Maschinenstillstände und -betriebskosten sollen optimiert werden. Vereinfacht kann diese Optimierung mit der Reduzierung der *total weighted completion time* verglichen werden. Sowohl die Reduzierung der Maschinenstillstände, als auch die Wahl eines schnelleren Werkzeuges (geringere Maschinenbetriebskosten) führen zu einer früheren Fertigstellung (C_j) der Aufträge ($\sum \omega_j C_j$).

- Die Rüstkosten sollen auf ein Minimum reduziert werden (TSC).

Die Problemstellung dieser Arbeit kann also mit dem Framework aus der Literatur folgendermaßen beschrieben werden:

$$P17 \mid p_{ij} = p_j,\ fmls,\ brkdwn,\ M_j \mid \sum \omega_j T_j,\ \sum \omega_j E_j,\ \sum \omega_j C_j,\ TSC$$

Folgende wichtige Restriktionen werden in diesem Framework nicht berücksichtigt:

- Werkzeuge können nicht parallel auf verschiedenen Maschinen gleichzeitig benutzt werden. Dementsprechend können Aufträge, die das gleiche Werkzeug benötigen, nicht gleichzeitig auf verschiedenen Maschinen produziert werden. Die Werkzeugbelegung muss also berücksichtigt werden, um sehr lange Wartezeiten zu vermeiden, die bei Nichtberücksichtigung zu einem unbrauchbaren Ergebnis führen könnten.
- Die Bearbeitungsgeschwindigkeit ist zwar unabhängig von der benutzten Maschine, aber dennoch abhängig vom eingesetzten Werkzeug.
- Rüstvorgänge können nur innerhalb des Rüstintervalls (6-19 Uhr) stattfinden. Demzufolge sollten die Zeitpunkte der Rüstvorgänge berücksichtigt werden, da es ansonsten ebenfalls zu langen Wartezeiten kommen kann, die zu einem unbrauchbaren Ergebnis führen könnten.
- Es können maximal vier Rüstmitarbeiter gleichzeitig im Einsatz sein. Diese Restriktion ist von niedriger Bedeutung, da es nur selten vorkommt, dass mehr als vier Rüstmitarbeiter gleichzeitig benötigt werden und selbst wenn dieser seltene Fall eintreten sollte, kommt es nur zu relativ kurzen Wartezeiten. Nichtsdestotrotz sollte genauer untersucht werden, ob diese Restriktion wirklich vernachlässigt werden kann.
- Der Gesamtlagerbestand ist auf insgesamt 500 Paletten begrenzt.

Zusammengefasst kann man sagen, dass die größte Herausforderung das effektive Berücksichtigen der Werkzeugbelegung und des Rüstintervalls sein wird. Das Überprüfen und Anpassen des Produktionsplans an die Restriktionen der Werkzeugbelegung und des Rüstintervalls stellt in diesem Sinne kein Problem dar. Was aber auf jeden Fall eine große Herausforderung darstellt, ist das effektive Berücksichtigen dieser Restriktionen während des Optimierungsprozesses innerhalb der Metaheuristik. Zusätzlich führt die hohe Anzahl an Aufträgen und Maschinen sicherlich unausweichlich zu einer relativ langen Laufzeit der Metaheuristik, auch bei einer effektiven Umsetzung aller Restriktionen.

4.5 Klassifikation der Komplexität des Problems dieser Arbeit

In den letzten Abschnitten wurde genauer auf die Komplexitätstheorie eingegangen und ein Framework präsentiert, das es möglich macht, das Problem dieser Arbeit mit ähnlichen, bereits in der Literatur bearbeiteten Problemen zu vergleichen. Es lässt sich durch diese Klassifikation folgender wesentlicher Schluss ziehen:

In der Literatur ist kein Algorithmus bekannt, der das *identical parallel machine* Problem mit *multi-objectives* in polynomieller Zeit lösen kann (es scheitert sogar oft schon an der Optimierung eines Kriteriums).[83] Demzufolge ist das in dieser Arbeit betrachtete Problem NP-schwer. Es ist also sehr unwahrscheinlich, dass im Rahmen dieser Diplomarbeit die Entwicklung eines optimalen Algorithmus möglich ist. Deshalb wird sich diese Arbeit auf Algorithmen konzentrieren, die das in dieser Arbeit betrachtete Problem zwar nicht optimal lösen können, aber in einer angemessener Zeit ein zufriedenstellendes Ergebnis liefern. Für ebendiese Herangehensweise sind Heuristiken und Metaheuristiken entwickelt worden.

4.6 Übersicht und Einteilung von Algorithmen

Im Allgemeinen können Optimierungsalgorithmen in zwei Kategorien eingeteilt werden: deterministische und stochastische Algorithmen (siehe Abbildung 4.2).

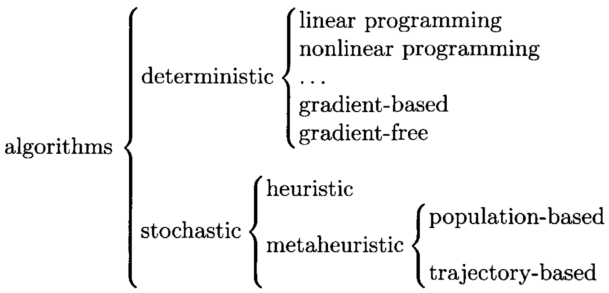

Abbildung 4.2: Einteilung von Optimierungsalgorithmen (Yang, 2010, S. 21)

Die meisten konventionellen Algorithmen sind deterministisch und erzielen für mehrere Optimierungsläufe immer das gleiche Ergebnis. So ist in *linear programming* der Simplex Algortihmus sehr weit verbreitet. Des Weiteren sind *gradient-based* Algorithmen wie z.B. der Newton-Raphson Algortihmus, der die Funktionswerte und deren Ableitungen nutzt, ebenfalls sehr bekannt. Liegen nicht stetige Funktionen vor, für die keine Ableitung gebildet werden kann, kommen oft *gradient-free* Algorithmen, wie z.B. der *Nelder-Mead downhill simplex* Algorithmus, zum Einsatz.[84] Diese Art von Algorithmen sind jedoch für das diskrete Optimierungsproblem dieser Arbeit nicht geeignet, weshalb auf diese auch nicht weiter eingegangen wird.

Für diese Arbeit sind vielmehr stochastische Algorithmen geeignet, die unter anderen als Heuristiken bzw. Metaheuristiken bekannt sind. Diese Art von Algorithmen sind sowohl in der Praxis als auch in der Literatur weit verbreitet, und stellen effektive Lösungsmethoden dar, die für das in dieser Arbeit betrachtete Problem in Frage kommen.

[83] vgl. Wang u. a., 2008, S.384
[84] vgl. Yang, 2010, S.21

4.7 Heuristiken

Ist man mit NP-schweren Problemen konfrontiert, sollte das Ziel ein Kompromiss zwischen Lösungsqualität und Rechenzeit sein. Wie bereits erwähnt, ist es bei NP-schweren Problemen oft sehr schnell der Fall, dass der Lösungsraum eine zu große Anzahl an durchzurechnenden Möglichkeiten erreicht, die zu lange benötigen, um alle einzeln beurteilen zu können. Es muss also ein Weg gefunden werden, den Lösungsraum zu durchsuchen, ohne jede einzelne Lösung durchzurechnen, wobei noch immer das Ziel einer möglichst optimalen Lösung in Betracht gezogen werden soll. Letztendlich muss einem jedoch bewusst sein, dass in diesem Fall das Finden einer optimalen Lösung keineswegs garantiert ist.[85]

4.7.1 Definition

Das Wort *Heuristik* ist abgeleitet vom griechischen Verb *heuriskein* und bedeutet etwas finden bzw. entdecken. Benutzt wurde dieses Verb in seiner Vergangenheitsform von Archimedes, als er aus der Badewanne sprang und sagte: "*eureka*", was so viel bedeutet wie "ich habe es gefunden". Es wäre allerdings fälschlich, aufgrund der Etymologie anzunehmen, dass eine Heuristik das Optimum garantiert findet.[86]

Alan Turing war vermutlich einer der ersten, der Heuristiken im zweiten Weltkrieg nutzte, um den berüchtigten deutschen Enigma Code zu entschlüsseln. *The Bombe* (der Name der eingesetzte elektromechanischen Maschine) benutzte heuristische Algorithmen, um aus einem Lösungsraum von etwa 10^{22} Möglichkeiten die korrekte Lösung zu finden und so die Nachricht der Deutschen zu entschlüsseln. Turing nannte seine eingesetzte Suchmethode *heuristic search*, da diese Methode die meiste Zeit zum Erfolg führt, es allerdings keine Garantie gab, dass die Nachrichten wirklich entschlüsselt werden.[87]

Eine Heuristik lässt sich also folgendermaßen definieren:

> *"A heuristik is a technique wich seeks good (i.e. near-optimal) solutions at a reasonable computational cost without being able to guarantee either feasability or optimality, or even in many cases to state how close to optimality a particular solution is."*[88]

Aus einem anderen Blickwinkel betrachtet, können Heuristiken als Ansätze betrachtet werden, die bewusst Erfahrungswerte einfließen lassen, um den Lösungsraum effektiv zu durchsuchen.[89] Um diesen Gedanken besser verstehen zu können, betrachten wir die klassische Metapher der Suche einer Nadel im Heuhaufen. Man kann ganz naiv vorgehen und jeden Kubikzentimeter im Heuhaufen durchsuchen. Diese Vorgehensweise ist vergleichbar

[85] vgl. Burke und Kendall 2013, S.8
[86] vgl. Reeves 1996, S.123
[87] vgl. Yang, 2010, S.8
[88] vgl. Rayward-Smith u. a. 1995, S.3 (vom Buchteil)
[89] vgl. Rimscha 2010, S.41

mit der Methode, jede der Möglichkeiten im Lösungsraum zu betrachten. Für einen sehr kleinen Heuhaufen reicht diese Methode vollkommen aus, ab einer bestimmten Größe ist diese Vorgehensweise allerdings sehr ineffektiv. Um große Heuhaufen effektiv zu durchsuchen, können Erfahrungswerte in die Suche mit einbezogen werden. Ein Erfahrungswert wäre z.B., die Erkenntnis, dass sich die Nadel meistens in der Nähe des Bodens befindet. In diesem Fall kann sich die Suche in erster Linie auf diesen Bereich konzentrieren. Es ist dadurch nicht garantiert, dass die Nadel in diesem Bereich gefunden wird, aber dadurch steigt die Wahrscheinlichkeit, die Nadel in angemessener Zeit zu finden. Es sollte einem jedoch bewusst sein, dass die heuristischen Verfahren in dem Maße versagen werden, in dem die Erfahrungswerte nicht der Realität entsprechen. Ein Schachspieler wird zum Beispiel öfter verlieren, wenn er den Läufer für wertvoller als die Dame hält.[90]

Demzufolge lassen sich Heuristiken auch folgendermaßen definieren:

> "*...,* [heuristcs are] *popularly known as rules of thumb, educated guesses, intuitive judgements or simply common sense. In more precise terms, heuristics stand for strategies using readily accessible though loosely applicable information to controll problem-solving processes in human beings and machine.*"[91]

Im Alltag werden heuristische Suchverfahren sehr oft unbewusst eingesetzt. Wenn man zum Beispiel zu Hause das Handy verlegt hat, sucht man zuerst dort, wo man es am häufigsten liegen lässt bzw. überall dort wo man war, nachdem man es das letzte Mal benutzt hat. Dieses Suchverfahren ist oft sehr effizient und führt meistens dazu, das Handy schnell zu finden. Niemand würde auf die Idee kommen, sein Handy zuerst in der Spülmaschine zu suchen. Erst aus Verzweiflung beginnt man dort zu suchen, wo das Handy eigentlich gar nicht sein könnte, weil man überall anders schon nachgeschaut hat. Tritt dieser Fall ein, bedeutet das nicht, dass das Suchverfahren schlecht ist, es heißt einfach nur, dass es für diesen speziellen Fall (z.B. ein Familienmitglied hat das Handy unabsichtlich mitgenommen) nicht ausgelegt ist, es also keine Garantie gibt, das Handy jedes Mal schnell zu finden, geschweige denn es überhaupt zu finden.

4.7.2 Einteilung der Heuristiken

Heuristiken können grundsätzlich in folgende Gruppen unterteilen werden:[92]

1. **Eröffnungsverfahren** zur Bestimmung einer initialen zulässigen Lösung. Oft auch mit dem Ziel eine möglichst gute Initiallösung bereitzustellen. Zu dieser Gruppe von Heuristiken zählen z.B. Prioritätsregelverfahren, die vor allem für Reihenfolgenprobleme eingesetzt werden. Prioritätsregelverfahren sortieren einzuplanende Objekte, wie z.B. Aufträge, nach einer bestimmten Prioritätsregel (Sortierregel) und planen

[90] vgl. Rimscha, 2010, S.46
[91] vgl. Pearl, 1985, Preface.vii
[92] vgl. Domschke, 1993, S.42

diese dann in der sortierten Reihenfolge ein. Das Ergebnis ist in diesem Fall ein initialer zulässiger Produktionsplan.

2. **Verbesserungsverfahren** zur Verbesserung einer gegebenen zulässigen Lösung. Zu dieser Gruppe zählen z.B. Vertauschungsverfahren, die ausgehend von einer zulässigen Lösung einzelne Variablen vertauschen, um schließlich eine Menge benachbarter Lösungen zu erhalten. Diese Verfahren werden in der Literatur auch oft als *Neighborhood Search* bezeichnet.

3. **Unvollständig exakte Verfahren**, wie z.B. vorzeitig abgebrochene exakte Verfahren. Exakte Verfahren werden dann abgebrochen, wenn diese z.B. zu lange brauchen würden, um die optimale Lösung zu finden, jedoch nach einer bestimmten Zeit schon eine zufriedenstellende Lösung bieten können und somit abgebrochen werden. Auf dieser Lösung kann dann aufgebaut werden.

4. Eine Kombination aus 1-3.

Des Weiteren können Heuristiken folgende zwei Grundeigenschaften haben:[93]

- **Deterministische Heuristiken**, die bei mehrfacher Anwendung auf ein und dasselbe Problem mit gleichen Startbedingungen stets die gleiche Lösung liefern.
- **Stochastische Heuristiken**, die eine Zufallskomponente beinhalten, durch die bei wiederholter Anwendung auf ein und dasselbe Problem in der Regel unterschiedliche Lösungen ausgegeben werden.

Zusammengefasst kann man sagen, dass Heuristiken schnell eine zufriedenstellende Lösung liefern und somit z.B. als Eröffnungsverfahren sehr gut geeignet sind, aber, vor allem für komplexe Probleme, eine solche Lösung oft alles andere als optimal ist. Das zentrale Problem von Heuristiken, die z.B. als Verbesserungsverfahren eingesetzt werden, ist das Steckenbleiben in einem lokalen Optimum. Um dieses Problem zu bekämpfen, wurden ausgefeilte Techniken entwickelt, die es möglich machen, lokalen Optima zu entkommen, um so den Lösungsraum effektiver zu durchsuchen. Diese Techniken sind unter anderem als Metaheuristiken bekannt, die vereinfacht gesagt Heuristiken steuern.[94]

4.8 Metaheuristiken

Der Ausdruck Metaheuristik ist in der Literatur zwar weit verbreitet, jedoch kommt es immer wieder vor, dass Metaheuristiken trotzdem als Heuristiken bezeichnet werden. Deswegen wird kurz auf die Herkunft des Wortes eingegangen.

[93] vgl. Domschke, 1993, S.42
[94] vgl. Zäpfel und Braune, 2005, S.28

4.8.1 Definition

Die Vorsilbe *meta* kommt, wie der Begriff Heuristik, aus dem Griechischen und bedeute "dahinter" bzw. "über etwas". In einer Hierarchie bezeichnet die Vorsilbe *meta* z.B. eine darüberliegende Ebene. Metaheuristiken können demnach als übergeordnete Heuristiken von Heuristiken gesehen werden. Demzufolge ist es nicht falsch, Metaheuristiken als Heuristiken zu bezeichnen. Einer der wohl bekanntesten Begriffe mit der Vorsilbe *meta*, ist die Metaphysik. In der Metaphysik beschäftigt man sich z.B. nicht mit der Gravitation selber (physikalische Definition), sondern stellt sich die philosophische Frage, wieso es diese überhaupt gibt. Beispielsweise spricht man auf einer Metaebene nicht über ein Problem selber, sondern darüber, wie man Probleme löst.[95] Von diesem Standpunkt aus betrachtet sind Metaheuristiken Algorithmen, die sich nicht mit dem Problem selber beschäftigen, sondern mit der Lösung des Problems. Es sind also die Heuristiken, die das Problem an sich bearbeiten. Dementsprechend können Metaheuristiken folgendermaßen definiert werden:

> "*A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions.*"[96]

Vereinfacht können Metaheuristiken mit Navigationssystemen verglichen werden (wobei Ziele mit einem Navigationssystem meistens erreicht werden, während mit einer Metaheuristik die optimale Lösung (das Ziel) keinesfalls garantiert ist). Das Navigationssystem gibt grundsätzlich einen Weg vor, wobei die eigentliche Umsetzung von den Methoden und Entscheidungen des Fahrers abhängig ist, die vergleichbar mit Heuristiken sind. Der Vorteil von Metaheuristiken kann anhand des Vergleichs zwischen Navigationssystem und Stadtplan veranschaulicht werden. Mit einem Stadtplan ist es genauso möglich, sein Ziel zu erreichen. Die eingesetzten Methoden des Fahrers bleiben gleich, genauso wie die Heuristiken auch ohne Metaheuristiken eingesetzt werden können. Jedoch gibt ein Navigationssystem einen Art Leitfaden (empfohlene Route) vor, der das Finden einer schnellen Route stark vereinfacht und es möglich macht, viele verschiedene Kriterien zu berücksichtigen (z.B. Staus oder Baustellen), die mit einem einfachen Stadtplan nicht berücksichtigt werden können. Mit modernen Navigationssystemen ist es möglich, Staus oder Baustellen effektiv zu entkommen. Dementsprechend können Metaheuristiken auch folgendermaßen definiert werden:

"*Metaheuristics, in their original definition, are solution methods, that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of solution*

[95] https://de.wiktionary.org/wiki/meta-; https://de.wikipedia.org/wiki/Metaphysik; https://www.urbandictionary.com/define.php?term=meta; (alle besucht am 20.10.18)
[96] vgl. Osman und Laporte, 1996, S.3

space."[97]

Einer der großen Vorteile von Metaheuristiken ist demzufolge das Entkommen von lokalen Optima, was das Erreichen einer optimale Lösung wahrscheinlicher gestaltet. Des Weiteren können Metaheuristiken für eine Vielzahl von Problemen eingesetzt werden und sind nicht nur auf ein bestimmtes Problem beschränkt.[98] Ein Navigationssystem ist z.B. auch nicht auf eine Stadt beschränkt. Möchte man eine neue Stadt erkunden, muss kein komplett neues Navigationssystem gekauft werden, es reicht lediglich sich einen neuen, entsprechenden Stadtplan herunterzuladen. Das Navigationssystem ist dann ebenfalls mit dem neuen Stadtplan einsetzbar. Ähnlich ist es auch mit Metaheuristiken. Metaheuristiken müssen zwar meistens an ein spezifisches Problem angepasst werden, das Grundprinzip kann jedoch beibehalten werden.

Auch wenn Metaheuristiken für eine Vielzahl von Problemen eingesetzt werden können, sind diese alles andere als plug-and-play freundlich. Die Anpassung der Operatoren und Parameter der jeweiligen Metaheuristiken nimmt eine erhebliche Zeit in Anspruch. Es ist für komplexe Probleme sogar oft der Fall, dass eine Metaheuristik mit Operatoren, die nicht effektiv an die Problemstruktur angepasst sind, unbrauchbar ist.[99]

4.8.2 Einteilung der Metaheuristiken

Viele Metaheuristiken sind von der Natur inspiriert, die sich über mehrere Milliarden von Jahren entwickelt hat und somit unglaublich effektive Lösungen für viele Probleme gefunden hat. Es bietet sich deshalb an, Lösungsansätze von der Natur abzuschauen.[100] Allerdings gibt es genauso Metaheuristiken, die nicht von der Natur inspiriert sind. Metaheuristiken lassen sich grundsätzlich folgendermaßen klassifizieren.[101]

- **Nature-Inspired vs. Non-Nature Inspired**: Es gibt von der Natur inspirierte Metaheuristiken, wie den Genetischen Algorithmus oder den Ameisen Algorithmus und dann gibt es Metaheuristiken, wie Tabu Search oder Iterated Local Search, die nicht von der Natur inspiriert sind.
- **Population-Based vs. Single Solution Based**: Eine andere Möglichkeit Metaheuristiken zu klassifizieren, ist die Anzahl an Lösungen, die gleichzeitig betrachtet werden. Wird eine ganze Population (viele Lösungen gleichzeitig) betrachtet oder immer nur eine einzige Lösung? Der Genetische Algorithmus ist z.B eine populationsbasierte Metaheuristik, wohingegen Simulated Annealing ein Einzellösungsalgorithmus ist.

[97] Glover, 2003, Preface
[98] vgl. Falkenauer, 1998, S.18
[99] vgl. Borenstein und Moraglio, 2014, Preface xiii
[100] vgl. Yang, 2010, S.22
[101] vgl. Sujata u. a., 2017, Preface xxii

- **Dynamic vs. Static Objective Function**: Diese Unterscheidung basiert auf den Eigenschaften der Zielfunktion. Bleibt die Zielfunktion im Laufe der Suche unverändert, spricht man von einer *static objective function*. Verändert sich die Zielfunktion im Laufe des Suchprozesses, spricht man von einer *dynamic objective function*. Diese Art der Zielfunktion wird z.B. bei Guided Local Search Algorithmen eingesetzt, um aktiv Informationen zu nutzen, die im Laufe der Suche auftreten, mit dem Ziel lokalen Optima zu entkommen.
- **One vs. Various Neighbourhood Structures**: Diese sogenannte Nachbarschaftsstruktur gibt die erlaubten Bewegungen im Lösungsraum vor. Ist nur eine bestimmte Bewegung erlaubt, spricht man von *single neighbourhood structure*, wohingegen Metaheuristiken, wie z.B. Variable Neighbourhood Search, mit einer Vielzahl von Bewegungen arbeiten und somit eine umfangreichere Erkundung des Lösungsraums ermöglichen.
- **Memory Usage vs. Memory-Less Methods**: Nutzen Metaheuristiken eine Art Suchprotokoll, spricht man von einer Memory-Funktion. Diese Funktion soll Metaheuristiken davon abhalten Lösungen zu betrachten, die bereits betrachtet wurden. Eine bekannte Metaheuristik mit Memory-Funktion ist die Tabu Search Metaheuristik.

4.8.3 Begriffsbestimmungen im Kontext der Optimierung mit Metaheuristiken

Bevor genauer auf einzelne Metaheuristiken eingegangen wird, ist es wichtig, häufig auftretende Begriffe in der Optimierung mit Metaheuristiken kurz zu erläutern.

4.8.3.1 Lösungsraum

Oft auch Suchraum (*search space*) genannt. Vereinfacht gesagt, lässt sich ein diskretes Optimierungsproblem auf das Durchsuchen eines endlichen Lösungsraums nach einer optimalen Lösung zusammenfassen. Dabei stellt der Lösungsraum die Menge der zulässigen Lösungen des betrachteten Problems dar.[102] Durch das einfache Reihenfolgenproblem der Einplanung von drei Aufträgen A, B und C auf einer Maschine, ergibt sich ein diskreter Lösungsraum Ω von $3! = 6$ Lösungen: $s(x_1, x_2, x_3) \in \{ABC, ACB, BAC, BCA, CAB, CBA\}$. Der Suchraum ergibt sich also aus der Anzahl an Variablen (x_i) und deren zulässigen Werten ($x_i \in \{A, B, C\}$), die diese einnehmen können. In vielen Fällen gibt es sogenannte Restriktionen (*constraints*), die den Lösungsraum einschränken. Eine offensichtliche Restriktion am soeben genannten Beispiel ist, dass ein Auftrag nicht an mehreren Positionen gleichzeitig eingeplant werden kann bzw. alle Aufträge genau einmal vorkommen müssen, sodass Kombinationen wie z.B. AAB oder CCC nicht vorkommen dürfen. Anders formuliert bedeutet das, dass z.B. die Variable x_3 zwar den Wert A einnehmen darf, jedoch nicht wenn x_1 oder x_2 den Wert A bereits eingenommen haben.

[102] vgl. Zäpfel und Braune, 2005, S.125

4.8.3.2 Zielfunktion

Sobald ein Optimierungsproblem betrachtet wird, ist eine Zielfunktion (*fitness function*, *objective function* bzw. auch oft *cost function*) notwendig, die die Güte der verschiedenen Lösungen angibt. Eine Zielfunktion ist also generell eine Vorschrift, die einer Menge von Konfigurationen eine Menge an Zahlen zuordnet.[103] Die Darstellung des Suchraums durch die Lösungen mit ihren Fitnesswerten definiert bildlich die Fitnesslandschaft.[104] Wird wieder das Reihenfolgenproblem von vorher betrachtet, dann wird jeder zulässigen Kombination $x_1x_2x_3$ im Lösungsraum Ω durch die Zielfunktion $f : \Omega \to \mathbb{R}$ ein reeller Fitnesswert zugeordnet. Die Zielfunktion f bewertet somit jede Lösung im Lösungsraum und wird deswegen auch oft Bewertungsfunktion genannt. Bewertet die Zielfunktion z.B. die Anzahl an Rüstvorgängen der einzelnen Lösungen im Lösungsraum Ω, wobei Auftrag A und C mit dem gleichen Werkzeug produziert werden können, sind die Reihenfolgen ABC und CBA mit einer höheren Anzahl an Rüstvorgängen verbunden als die restlichen Kombinationen, da A und C in diesen beiden Fällen nicht nacheinander produziert werden. Die Zielfunktion muss dementsprechend an das zu optimierende Ziel angepasst werden. Verschiedene Zielfunktionen erzeugen für ein und dasselbe Problem verschiedene Fitnesslandschaften. Je nachdem ob ein Maximierungs- oder Minimierungsproblem vorliegt, wird das Maximum bzw. respektive Minimum der Zielfunktion gesucht.

4.8.3.3 Nachbarschaftsstruktur

Um der Definition von Heuristiken gerecht zu werden und den Lösungsraum nicht rein zufällig zu durchsuchen, wird oft der Begriff der sogenannten Nachbarschaftssuche (*Neighborhood Search*) benutzt, die als Wanderung entlang einer Kette benachbarter Lösungen betrachtet werden kann. Die Nachbarschaft $N(s)$ einer Lösung $s \in \Omega$ ist die Menge aller Lösungen, die von s aus durch einfache Modifikation erreichbar sind.[105] Die Funktion N ist somit eine Funktion, die einer Lösung durch eine bestimmte Modifikation der Variablen, eine Menge an möglichen Lösungen zuordnet und somit gilt $N(s) \subseteq \Omega$. Dementsprechend ist die Definition der durchzuführenden Modifikation ausschlaggebend für die Definition der Nachbarschaftsstruktur. Am Beispiel des bereits betrachteten Reihenfolgenproblems, ist eine mögliche Modifikation die Vertauschung zweier Variablen. Für $s_1 = ABC$ gilt $N_1(s_1) \in \{BAC, ACB, CBA\}$, welches die Nachbarschaft der Lösung ABC durch die einfache Vertauschung von zwei Variablen darstellt. Demzufolge sind für diese Modifikation die Lösungen BCA und CAB nicht in der Nachbarschaft von ABC, da diese nicht durch die einmalige Vertauschung zweier Variablen erreicht werden können.

Da die Nachbarschaftsstruktur einen der wichtigsten Begriffe darstellt, wird ein weiteres

[103] vgl. Dittes, 2015, S.31
[104] vgl. Zäpfel und Braune, 2005, S.125
[105] vgl. Zäpfel und Braune 2005, S.27

explizites Beispiel betrachtet: Man kann sich die Nachbarschaftsstruktur wie die verschiedenen möglichen Züge der Figuren im Schach vorstellen. Das Schachbrett ist der Lösungsraum und jedes Feld stellt eine mögliche Lösung dar. Ein Zug ist somit vergleichbar mit der Modifikation einer Lösung, die zu einer anderen Lösung im Lösungsraum führt. Jede Figur hat ihre eigene Nachbarschaftsstruktur und die damit verbundenen Modifikationen. Die wohl kleinste Nachbarschaftsstruktur hat in diesem Fall der Bauer, der in den meisten Fällen nur ein Feld nach vorne gehen darf und sich somit nur auf einer sehr kleinen Anzahl an Feldern bewegen kann. Ein Bauer alleine kann also den Lösungsraum nur sehr begrenzt durchsuchen und kann sogar sehr leicht stecken bleiben, wenn er von anderen Figuren blockiert wird. Die wohl größte Nachbarschaftsstruktur hat die Dame, die sich in alle Richtungen bewegen darf. Die Dame kann theoretisch jedes Feld am Schachbrett erreichen, jedoch nicht immer in einem Zug. Der Vergleich zwischen Läufer und Turm zeigt, wie ausschlaggebend die Definition der Nachbarschaftsstruktur ist. Der Turm und der Läufer haben zwar ca. die gleiche Größe der Nachbarschaftsstruktur, jedoch ist es nur dem Turm möglich, jedes Feld am Schachbrett zu erreichen. Der Läufer, der sich auf einem weißen Feld befindet, kann sich durch die Definition seiner Züge nur auf weißen Feldern bewegen. Dementsprechend kann dieser Läufer durch seine Definition der Nachbarschaftsstruktur und der damit verbundenen Modifikationen einer Lösung, keine Lösungen auf schwarzen Feldern erreichen bzw. darstellen. Unter anderem ist das auch ein Grund, weshalb der Turm im Schach mehr Wert ist als der Läufer. Allerdings ist diese Schlussfolgerung nicht unbedingt direkt auf den Prozess der Optimierung mit Metaheuristiken übertragbar. Ist z.B. bekannt, dass die optimale Lösung sich auf einem weißen Feld befindet, würde sich die Nachbarschaftsstruktur des Läufers, der sich auf einem weißen Feld befindet, als sehr effektiv erweisen, da durch diese Nachbarschaftsstruktur die Lösungen auf den schwarzen Feldern ausgeschlossen werden und somit der Lösungsraum effektiver durchsucht werden kann.

Es ist also entscheidend, sowohl die Struktur als auch problemspezifische Informationen des betrachteten Problems effektiv in die Nachbarschaftsstruktur einzubauen.[106] Je besser das gelingt, umso effektiver kann der Lösungsraum durchsucht werden und umso wahrscheinlicher ist es, das globale Optimum zu finden.

4.8.3.4 Lokales vs. globales Optimum

Das Ziel von Metaheuristiken ist es, die optimale Lösung eines Problems zu finden, wobei der Begriff Optimum für den Suchprozess mit Metaheuristiken nochmals in lokalem und globalem Optimum unterschieden wird. Diese Unterscheidung ist einer der Hauptgründe für die Notwendigkeit von Metaheuristiken.

Ein lokales Optimum wird folgendermaßen definiert: Sei $N_k(s)$ die Nachbarschaft der

[106] vgl. Dorigo und Stützle, 2019, S.315

Lösung $s \in \Omega$, wobei Ω den Lösungsraum darstellt. s wird als lokales Optimum bezeichnet, wenn es keine Lösung $s' \in N_k(s) \subseteq \Omega$ gibt, für die gilt $f(s') < f(s)$, wobei f die Zielfunktion eines Minimierungsproblems darstellt.[107] Für ein Maximierungsproblem gilt $f(s') > f(s)$. In Worten ausgedrückt bedeutet das, dass durch die möglichen Modifikationen einer Lösung, die durch die Nachbarschaftsstruktur N_k vorgegeben werden, alle möglichen Lösungen in der Nachbarschaft $N_k(s)$ eine schlechtere Bewertung der Zielfunktion erzielen als die Lösung s. Infolgedessen hängt die Definition eines lokalen Optimums nicht nur vom Verlauf der Zielfunktion ab, wie es fälschlicherweise oft behauptet wird, sondern eben auch von der Definition der Nachbarschaftsstruktur.[108]

Ein globales Optimum ist unabhängig von der Nachbarschaftsstruktur und ist folgendermaßen definiert: Eine Lösung $s \in \Omega$ wird als globales Optimum bezeichnet, wenn es keine Lösung $s' \in \Omega$ gibt, für die gilt $f(s') < f(s)$. Für ein Maximierungsproblem gilt wieder $f(s') > f(s)$. Das bedeutet, dass es im gesamten Lösungsraum Ω keine bessere Lösung als s gibt und somit die Lösung s das gesuchte Optimum des Problems darstellt. Ein globales Optimum kann auch als ein lokales Optimum definiert werden, das für alle möglichen Nachbarschaftsstrukturen gilt.[109] Aus diesem Blickwinkel betrachtet bedeutet das, dass eine Lösung, die kein lokales Optimum darstellt, auch kein globales Optimum sein kann. Deswegen sollte die Suche nach lokalen Optima nicht unterschätzt werden.

4.8.3.5 Breitensuche vs. Tiefensuche

Für die Effizienz einer Metaheuristik ist weniger entscheidend, ob diese eine Vielzahl von Prinzipien beinhaltet, sondern viel mehr, ob sie Suchprinzipien verwirklicht, die sowohl eine Breitensuche (*diversification* bzw. *exploration*) als auch eine Tiefensuche (*intensification* bzw. *exploitation*) im Lösungsraum ermöglichen. Eine Breitensuche wird dadurch charakterisiert, dass ausgehend von einer Lösung in weiteren Iterationen "neue Regionen" (bzw. unerforschte Regionen) im Lösungsraum erreicht werden, die eine bestimmte Distanz von der ursprünglichen Lösung entfernt sind. Eine Tiefensuche hingegen lässt sich dadurch charakterisieren, dass im nächsten Schritt eine Suche im lokalen Sinne stattfindet, die lokale bzw. naheliegende Lösungen in der "gleichen Region" des Lösungsraums untersucht.[110]

Das Konzept der Tiefensuche ist ein eher intuitiver Prozess, der meistens dadurch gekennzeichnet ist, dass vielversprechende Regionen des Lösungsraums intensiver durchsucht werden sollten, um dort gezielt die beste Lösung dieser Region zu finden. Da sich am Anfang eines Optimierungsproblems meistens nicht genau sagen lässt, welche Regionen vielversprechend sind, ist eine vielversprechende Region oft dadurch gekennzeichnet, dass in dieser Region bereits gute Lösungen im Laufe des Suchprozesses gefunden wurden.

[107] vgl. P. Hansen u. a., 2019, S.59
[108] vgl. Moscato und Cotta, 2019, S.282
[109] vgl. P. Hansen u. a., 2019, S.60
[110] vgl. Zäpfel und Braune, 2005, S.129

Jedoch sollte mit der Tiefensuche nicht übertrieben werden, da sonst Gefahr läuft, nur einen sehr begrenzten Abschnitt des Lösungsraums zu durchsuchen und unter Umständen nur Lösungen zu finden, die sehr weit vom globalen Optimum entfernt sind. Um dem entgegenzuwirken, sollte ebenfalls die Breitensuche genutzt werden, die die Suche auf unerforschte Regionen des Lösungsraums lenkt, und so zu einer besseren globalen Durchsuchung des Lösungsraums beiträgt.[111] Die Breiten- und Tiefensuche kann demnach als globale bzw. respektive lokale Suche im Lösungsraum zusammengefasst werden.

Um besser verstehen zu können, weshalb die Unterscheidung zwischen Breiten- und Tiefensuche so wichtig ist und wieso eine ausgewogene Kombination oft erstrebenswert ist, sollten diese beiden Suchprinzipien nochmals in Verbindung mit der Definition von Heuristiken bzw. Metaheuristiken gebracht werden. Wenn wir uns erinnern, werden Heuristiken bevorzugt dort eingesetzt, wo der Lösungsraum eine zu große Anzahl an Lösungen darstellt, und dementsprechend das Durchrechnen jeder einzelnen Lösung zu viel Zeit in Anspruch nehmen würde. Heuristiken liefern in relativ kurzer Zeit zufriedenstellende Ergebnisse, die sich jedoch im Hinblick auf die Lösungsqualität oft doch relativ weit entfernt vom globalen Optimum befinden. Das liegt oft daran, dass sich einfache Heuristiken zu sehr auf die Tiefensuche konzentrieren und somit eine zu lokale Suche ausüben, was dazu führt, dass meistens nur lokale Optima gefunden werden. Das Ziel, das globale Optimum zu finden, rückt dementsprechend oft ungewollt in den Hintergrund. Durch die Anwendung von Metaheuristiken, die, vereinfacht gesagt, Heuristiken im Lösungsraum steuern, wird oft der Gedanke der Breitensuche forciert, indem die lokale Suche der Heuristiken in einer Vielzahl von verschiedenen Regionen des Lösungsraums eingeleitet wird, um das Ziel das globale Optimum zu finden, wieder mehr in den Vordergrund zu rücken. Das Durchsuchen „aller Regionen“ wäre im Allgemeinen mit der Betrachtung aller Lösungen im Lösungsraum gleichzustellen und wäre somit nicht im Sinne der Definition von Heuristiken bzw. Metaheuristiken, die den Lösungsraum clever durchsuchen sollten. Dementsprechend entsteht für Metaheuristiken ein gewisses Dilemma zwischen Breiten- und Tiefensuche, welches sich oft nur durch ein effektives Zusammenspiel beider Suchprinzipien lösen lässt, das sich vor allem durch das geschickte Nutzen bzw. Einbauen problemspezifischer Strukturen und Informationen während des Suchprozesses umsetzen lässt.

4.8.4 Die 'No Free Lunch' Theorie

Nachdem genauer auf einzelne Charakteristiken eingegangen wurde, die eine Metaheuristik verfolgen sollte, stellt sich die Frage, ob es denn überhaupt möglich wäre, eine “allmächtige” Metaheuristik zu entwickeln, die jedes Problem effektiv lösen könnte. Dieser Gedanke wird rasch durch die '*No Free Lunch*' (NFL) Theorie widerlegt, die vereinfacht gesagt zeigt, dass das mittlere Verhalten zweier beliebiger, vernünftiger Suchverfahren auf dem Raum

[111] vgl. Gendreau und Potvin, 2019, S.45f

aller möglichen Zielfunktionen über einem Lösungsraum stets gleich gut ist.[112] Daraus ist insbesondere abzuleiten, dass wenn z.B. ein Algorithmus A einen Algorithmus B für eine diskrete Zielfunktion f_1 übertrifft (also bessere Ergebnisse liefert), dann existiert eine diskrete Zielfunktion f_2, für die der Algorithmus B den Algorithmus A übertrifft. Aussagen wie: "Im Durchschnitt ist der vorgestellte Algorithmus über die betrachteten Funktionen der beste", wie sie oft bei den Performance Analysen von Metaheuristiken gemacht werden, sind somit nicht sehr aussagekräftig.[113] Es ist sogar oft der Fall, dass die Referenzfunktionen (*benchmark functions*), mit denen Metaheuristiken üblicherweise verglichen werden, mit in der Praxis auftretenden Funktionen wenig gemeinsam haben.[114] Es kann sogar gezeigt werden, dass, selbst wenn in der Praxis Funktionen auftreten, die einer Referenzfunktion "sehr ähnlich" sind, voraussichtlich trotzdem nur unbefriedigende Resultate erzielt werden können, obwohl der verwendete Algorithmus für die Referenzfunktion sehr gut abgeschnitten hat.[115] Ein weiterer Kritikpunkt, der durch die Benutzung von Referenzfunktionen entsteht, ist, dass Algorithmen durch übertriebenes Feintuning für die betrachteten Referenzfunktionen zwar gut abschneiden, aber im Endeffekt dann oft ungeeignet für Probleme in der Praxis sind.[116]

Allerdings halten die Voraussetzungen der NFL Theorie in der Praxis oft nicht stand. Einerseits ist die Betrachtung aller möglichen diskreten Funktionen eines Problems oft nicht repräsentativ für reale Probleme[117] und andererseits gelten die Voraussetzungen ebenfalls nicht, sobald Restriktionen im Lösungsraum vorhanden sind (was auf die meisten Probleme in der Praxis zutrifft).[118] Dennoch ist eine wertvolle Schlussfolgerung der NFL Theorie für die Praxis, dass Metaheuristiken problemspezifisches Wissen nutzen müssen, um bessere Ergebnisse liefern zu können, als eine zufällige Durchsuchung des Lösungsraums.[119]

Die NFL Theorie ist ein in der Literatur vielfach diskutiertes Thema, das zu einem kritischen Blick auf Metaheuristiken inspiriert. Besonders relevant für diese Arbeit ist dabei, dass es kaum (wenn überhaupt) möglich ist, mit gutem Grund zu argumentieren welche Metaheuristiken für das Problem der Diplomarbeit am besten geeignet sind. Selbst wenn in dieser Diplomarbeit eine effektive Metaheuristik entwickelt wird, ist nicht sichergestellt, dass diese auch in Zukunft für alle sinnvollen möglichen Inputs des Problems weiterhin zuverlässig gute Ergebnisse liefert. Diese Aussage wird durch die Definition von Heuristiken bzw. Metaheuristiken bestätigt, die keine Garantie auf das Finden einer optimalen bzw. zufriedenstellenden Lösung bieten. Einer der wichtigsten Punkte ist somit das wiederholt erwähnte Nutzen von problemspezifischen Informationen und Strukturen während des

[112] vgl. Fritzsche, 2009, S.41
[113] vgl. Igel, 2014, S.6
[114] vgl. Igel, 2014, S.19f
[115] vgl. Igel, 2014, S.15
[116] vgl. Whitley und Watson, 2005, S.20
[117] vgl. Whitley und Watson, 2005, S.7
[118] vgl. Igel, 2014, S.11
[119] vgl. Wolpert und Macready, 2005, S.1

Suchprozesses mit Metaheuristiken.

4.8.5 Kritische Reflexion über die Vielzahl an Metaheuristiken in der Literatur

In der Literatur lässt sich eine Unmenge an Metaheuristiken finden, wodurch die Auswahl einer geeigneten Metaheuristik für ein bestimmtes Problem erschwert wird. Es scheint, als könnte jede Tierart, jedes Insekt und jedes natürliche bzw. physikalische Phänomen als Vorbild für eine effektive Metaheuristik dienen. Darüberhinaus scheinen neue Metaheuristiken immer bessere Resultate zu erzielen, als die meisten bisher präsentierten Metaheuristiken in der Literatur. Eine sehr aktuelle und kritische Sichtweise der Metaheuristik-Literatur wird in Sörensen (2015) präsentiert. Die Hauptprobleme in der zugehörigen Literatur lassen sich in zwei Punkten zusammenfassen:

> "The more recent surge of 'novel' methods (...) seems to have very little on offer besides a new way of selling existing ideas [with new vocabulary]."[120]

Demnach ist ein Problem, dass neue Metaheuristiken oft das Rad neu erfinden, indem alte Prinzipien mit neuen Metaphern und dem dazugehörigen Vokabular neu verkauft werden. Sörensen betont ebenfalls, dass, wenn ein und dieselbe Idee durch eine Vielzahl von verschiedenen Ausdrücken bzw. Metaphern dargestellt wird, dies nicht nur den Vergleich von Metaheuristiken, sondern auch das Identifizieren von wirklich neuen und innovativen Prinzipien erschwert.

> "(...) the ultimate goal of science is to understand. True innovation in metaheuristics research therefore does not come from yet another method that performs better than its competitors, certainly if it is not well understood why exactly this method performs well. (...) Those authors that manage to tune their algorithm's parameters to just the right values for it to perform well on the standard benchmark sets are published, the others not."[121]

Hier wird auf das bereits in der NFL Theorie erwähnte Problem des Feintunings von Metaheuristiken für ausgewählte Referenzfunktionen hingewiesen. Die Referenzfunktionen haben also oft nicht viel mit der Praxis gemeinsam und auch, wieso eine Metaheuristik besser abschneidet als andere, ist oft nicht nachvollziehbar. Zusätzlich besteht die Gefahr, dass Metaheuristiken, die ein innovatives Prinzip nutzen, nicht veröffentlicht werden, weil sie auf den Referenzfunktionen nicht gut abschneiden bzw. nicht besser als die anderen Metaheuristiken.

Aus diesen zwei Punkten lässt sich zusammenfassen, dass neue und folglich laut Literatur auch oft "bessere" Metaheuristiken kritisch betrachtet werden sollten. Es sollte den Resultaten der Anwendung auf Referenzfunktionen nicht zu viel Aufmerksamkeit geschenkt werden.

[120] Sörensen, 2015, S.9
[121] Sörensen, 2015, S.13

Eher sollte darauf geachtet werden, welche Prinzipien neu und welche in bereits bekannten Metaheuristiken vorhanden sind und angesichts welcher Prinzipien eine Metaheuristik in bestimmten Situationen zu besseren Ergebnissen führt als andere. Dementsprechend lässt sich die Suche nach einer geeigneten Metaheuristik für ein bestimmtes Problem durch folgende Frage zusammenfassen: Welche Kombination von Prinzipien ist für das betrachtete Problem am vielversprechendsten?

4.8.6 Vielversprechende Metaheuristiken in der Literatur

Durch die Vielzahl an Metaheuristiken in der Literatur ist es oftmals schwierig, erfolgsversprechende Prinzipien zu identifizieren. Im folgenden Abschnitt werden vielversprechende Metaheuristiken präsentiert, die weitverbreitete Prinzipien zur Metaheuristik-Literatur beigetragen haben. Das Augenmerk in diesem Abschnitt liegt nicht im Detail der vielen Varianten einer Metaheuristik, sondern viel mehr auf dem Hervorheben der vielversprechenden Prinzipien, die die einzelnen Metaheuristiken beinhalten. Zumal einige Metaheuristiken eine *local search* (LS) Heuristik nutzen, wird diese kurz definiert.

Local Search

Eine LS Heuristik geht von einer initialen Lösung x aus und durchsucht die Nachbarschaft $N_{LS}(x)$ nach einer besseren Lösung. Die Nachbarschaft $N_{LS}(x)$ ist für eine LS Heuristik oft so definiert, dass sie die kleinste mögliche Veränderung einer Lösung betrachtet. Ist eine bessere Lösung in der Nachbarschaft $N_{LS}(x)$ vorhanden, wird diese übernommen und die Nachbarschaft der neuen Lösung wird ebenfalls nach einer besseren Lösung durchsucht. Dieser Vorgang wird so oft wiederholt, bis keine bessere Lösung mehr gefunden wird und somit in den meisten Fällen ein lokales Optimum vorliegt. Es besteht die Möglichkeit, in jeder Iteration entweder die beste Lösung der gesamten betrachteten Nachbarschaft zu übernehmen (auch als *best improvement* Heuristik bekannt) oder die erstbeste betrachtete Lösung der Nachbarschaft zu übernehmen (auch als *first improvement* Heuristik bekannt). Die *best improvement* Heuristik ist sehr zeitintensiv, da für jede Lösung die gesamte Nachbarschaft durchsucht werden muss. Wohingegen bei der *first improvement* Heuristik jede Nachbarschaft nur solang durchsucht werden muss, bis eine bessere Lösung gefunden wird, auch wenn diese meistens nicht die beste der Nachbarschaft ist.[122]

4.8.6.1 Variable Neighborhood Search

Die klassische Variable Neighbourhood Search (VNS) Metaheuristik zählt wohl zu den einfachsten Metaheuristiken in der Literatur. Sie kombiniert eine simple LS Heuristik mit mehreren Nachbarschaftsstrukturen $N_k(s)$, daher auch der Name V*ariabel Neigborhood*

[122] vgl. Hansen und Mladenović, 2009, S.4f

Search. Wobei $k = 1, ..., k_{max}$ und k_{max} die Anzahl der verschiedenen Nachbarschaftsstrukturen darstellt. Der einzige Parameter (zusätzlich zu einem Abbruchkriterium) ist somit die Anzahl der Nachbarschaftsstrukturen, die während der Suche benutzt werden.[123] Nur einen Parameter zu haben, macht es, im Vergleich zu anderen Metaheuristiken mit vielen Parametern, um einiges einfacher nachzuvollziehen, wann bzw. weshalb VNS effektiv ist und wann nicht.[124]

VNS basiert folglich auf dem gezielten Nutzen der Änderung der Nachbarschaftsstruktur, um sowohl eine Tiefensuche als auch eine Breitensuche zu verfolgen. Einerseits wird so gezielt nach lokalen Optima gesucht und andererseits ist es durch die Veränderung der Nachbarschaftsstruktur möglichen, lokalen Optima zu entkommen. Dementsprechend macht sich VNS die bereits erwähnte Definition eines lokalen Optimums zunutze, die besagt, dass ein lokales Optimum unter anderem von der Nachbarschaftsstruktur abhängig ist.[125]

4.8.6.2 Greedy Radomized Adaptive Search

Greedy Randomized Adaptive Search Procedure (GRASP) ist eine iterative Metaheuristik, in der jede Iteration aus zwei Phasen besteht: einer Konstruktionsphase und einer LS Phase. Die beste Lösung aller Iterationen wird dann als Lösung herangezogen. In der Konstruktionsphase werden Schritt für Schritt die Elemente (bzw. Variablen) einer zulässigen Lösung zusammengestellt. Diese Phase wird durch eine sogenannte *greedy* Heuristik durchgeführt, die jedem möglichen Element eines Schrittes seinen Nutzen bzw. Gewinn zuordnet. Diese Heuristik wird oft als kurzsichtig bezeichnet, da sie den Nutzen jedes möglichen Elements nur aus der Perspektive des aktuellen Schrittes bewertet und somit die Auswirkungen auf zukünftige Schritte vernachlässigt. Aus der Liste der bestmöglichen Kandidaten für einen Schritt (oft auch *restricted candidate list* (RCL) genannt), wird dann oft zufällig ein Kandidat ausgewählt. Würde für jeden Schritt immer nur das beste Element ausgewählt werden, wäre die mögliche Anzahl an verschiedenen Lösungen sehr eingeschränkt. Die durch die Konstruktionsphase erzeugte Lösung ist oft kein lokales Optimum, weshalb anschließend noch eine LS Heuristik eingesetzt wird, die die Nachbarschaft einer Lösung nach lokalen Optima durchsucht. Der Schlüssel zum Erfolg einer effektiven LS Heuristik ist einerseits das Nutzen einer an das Problem angepassten Nachbarschaftsstruktur und andererseits bereits bei einer relativ guten Lösung zu starten, was durch den Einsatz einer *greedy* Heuristik in der Konstruktionsphase von GRASP gewährleistet ist.[126]

Die klassische GRASP Metaheuristik hat, ähnlich wie VNS, neben dem Abbruchkriterium oft nur einen Parameter, der die Qualität der Kandidaten in der RCL bestimmt.[127] Dieser

[123] vgl. Mladenović und P. Hansen, 1997, S.1f
[124] vgl. Hansen und Mladenović, 2009, S.2
[125] vgl. P. Hansen u. a., 2019, S.58f
[126] vgl. Feo und Resende, 1995, S.110ff
[127] vgl. Resende und Ribeiro, 2019, S.172

Parameter gibt an, welche bzw. wieviele Kandidaten in jedem Schritt für die zufällige Auswahl eines Elements zur Verfügung stehen. Außerdem bietet sich die Konstruktionsphase von GRASP sehr gut für die Zusammenstellung einer initialen Population des genetischen Algorithmus an.[128] Ein großer Nachteil von GRASP ist, dass die Iterationen unabhängig voneinander sind und demzufolge die Konstruktionsphase im Laufe der Optimierung nicht aus den vorherigen Iterationen dazulernt. Um diesem Nachteil entgegenzuwirken, ist es möglich, Informationen von guten Lösungen im Laufe der Optimierung zu speichern, um die Auswahlwahrscheinlichkeit der Kandidaten in der RCL zu beeinflussen.[129] Dieses Prinzip ist sehr ähnlich dem Prinzip, das bei der Ant Colony Optimization genutzt wird.

4.8.6.3 Ant Colony Optimization

Ant Colony Optimization (ACO) basiert auf einem Experiment von Goss et al. (1989) mit argentinischen Ameisen (siehe Abbildung 4.3). Das Experiment zeigt, wie effektiv Ameisen zwischen einer Futterstelle und ihrem Nest den kürzesten Weg finden, indem sie über eine sogenannte Pheromonspur miteinander kommunizieren. In dem Experiment wird das Nest (N) mit einer Futterstelle (F) mit zwei identischen Modulen (Brücken) verbunden, wobei jedes Modul aus zwei Wegen besteht, die unterschiedlich lang sind. Sobald die erste Ameise die Futterstelle gefunden hat, hinterlässt sie auf ihrem Weg zurück zum Nest eine Pheromonspur (1). Des Weiteren folgen weitere Ameisen, die sich gegen Anfang noch relativ gleichmäßig auf allen möglichen Wegen verteilen und ebenfalls auf ihrem Weg eine Pheromonspur hinterlassen (2). Nach kürzester Zeit ist eine deutliche Präferenz der Ameisen für den kürzesten Weg zwischen Futterstelle und Nest zu erkennen (3).[130]

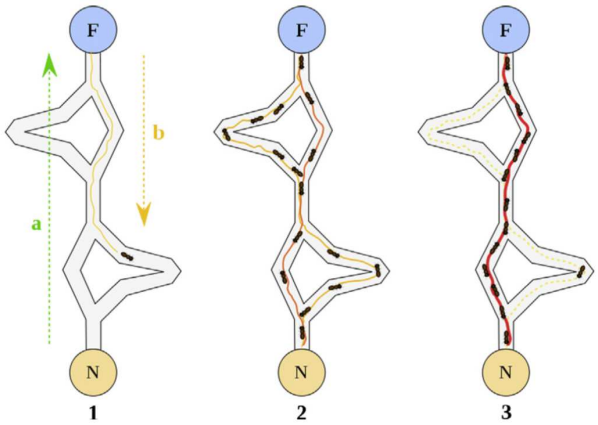

Abbildung 4.3: Wie Ameisen den kürzesten Weg finden (Toksari, 2016, S.778; angelehnt an Goss u. a., 1989, S.579)

Doch wie genau schaffen es die Ameisen, alleine durch die gelegte Pheromonspur den kürzesten Weg zu identifizieren? Die Konzentration der Pheromonspur erhöht die Wahr-

[128] vgl. Resende und Ribeiro, 2019, S.204
[129] vgl. Resende und Ribeiro, 2019, S.176f
[130] vgl. Goss u. a., 1989, S.579

scheinlichkeit, dass sich eine Ameise für einen bestimmten Weg entscheidet. Zu Beginn der Nahrungssuche ist noch keine Pheromonspur auf allen möglichen Wegen vorhanden, wodurch alle möglichen Wege die gleiche Wahrscheinlichkeit haben, weswegen auch eine gleichmäßige Verteilung in (2) stattfindet. Die Ameisen, die den kürzesten Weg wählen, erreichen die Nahrungsquelle als Erste. Sie treten ihren Rückweg früher wieder an als die Ameisen auf einem längeren Weg. Die Ameisen, die wiederum den kürzesten Weg am Rückweg zum Nest wählen, kommen ebenfalls früher am Nest an, als die Ameisen die am Rückweg einen längeren Weg wählen. Dadurch steigt die Pheromonspur am kürzesten Weg schneller an, als auf den längeren Wegen. Durch diese schnellere Erhöhung der Pheromonkonzentration am kürzesten Weg erhöht sich wiederum die Wahrscheinlichkeit, dass eine Ameise diesen Weg wählt, was erneut die Pheromonkonzentration am kürzesten Weg erhöht.[131]

Doch wie lässt sich dieses Prinzip für die Optimierung eines Problems nutzen? Dies lässt sich gut mit dem zu optimierenden Problem in dieser Diplomarbeit erklären. Vereinfach erklärt, lässt sich das Hauptproblem der Diplomarbeit auf die fortlaufende Einplanung von einzelnen Aufträgen auf jeweils einer bestimmten Menge an möglichen Maschinen zusammenfassen, bis daraus ein fertiger Produktionsplan entsteht. Jeder Auftrag hat von den insgesamt 17 Maschinen eine Teilmenge an möglichen Maschinen, auf denen dieser eingeplant werden kann. Mit ein bisschen Fantasie lässt sich dieser Einplanungsprozess, der bestimmt, auf welcher Maschine ein Auftrag eingeplant werden soll, mit dem Entscheidungsprozess einer Ameise für den kürzesten Weg vergleichen. Jede Ameise erstellt sozusagen ihren eigenen Produktionsplan, der sich durch die Einplanung aller Aufträgen ergibt, sowie auch jede Ameise ihren eigenen Weg zur Futterstelle sucht. Am Anfang des Optimierungsprozesses hat jede mögliche Maschine jedes Auftrags die gleiche Wahrscheinlichkeit. Die Länge des Weges zur Futterstelle kann verglichen werden mit den entstehenden Kosten eines fertigen Produktionsplans. So, wie es das Ziel der Ameisen ist, den kürzesten Weg zur Futterstelle zu finden, ist es auch das Ziel den Produktionsplan mit den geringsten entstehenden Kosten zu finden. Je geringer die Kosten eines fertigen Produktionsplans sind, umso höher wird die Wahrscheinlichkeit, dass weitere Ameisen diesen Weg gehen. Dementsprechend erhöhen sich auch die Wahrscheinlichkeiten der ausgewählten Maschinen für die einzelnen Aufträge eines fertigen Produktionsplans mit geringen Kosten.

ACO ist somit eine populationsbasierte Metaheuristik und kann als eine Vielzahl parallel laufender stochastischer Konstruktionsprozesse gesehen werden. Das Zentrale Element eines ACO-Algorithmus ist die künstliche Ameise als Teil einer Kolonie. Jede künstliche Ameise konstruiert schrittweise eine zulässige Lösung für das gegebene Problem. Dabei baut sie eine Lösung aus den möglichen Elementen (bzw. Variablen) einer Lösung zusammen.[132] Wie bereits erwähnt, ist die Entscheidungspolitik einer Ameise stochastischer

[131] vgl. Zäpfel und Braune, 2005, S.106
[132] vgl. Zäpfel und Braune, 2005, S.108

Natur und basiert hauptsächlich auf dem Prinzip der Pheromonspur aller Ameisen in einer Kolonie. Die Pheromonspur macht es möglich, im Laufe des Optimierungsprozesses einen Lernprozess zu fördern (welcher im GRASP-Algorithmus nicht vorhanden ist, weshalb sich dieses Prinzip des ACO-Algorithmus gut mit dem GRASP-Algorithmus kombinieren lässt). Des Weiteren lassen sich aber auch einfach problemspezifische Informationen nutzen, die die Auswahlwahrscheinlichkeiten zusätzlich beeinflussen. Durch die weniger deterministische Natur des ACO-Algorithmus lassen sich eine weitaus größere Anzahl an verschiedenen Lösungen erstellen als mit dem GRASP-Algorithmus, wodurch eine bessere Breitensuche möglich ist.[133] Um die Diversität der Suche länger aufrecht zu erhalten, wird der Mechanismus der Pheromonverdunstung benutzt, der ebenfalls in der Natur stattfindet. Dieser Mechanismus reduziert langsam die bereits gelegten Pheromonspuren während der Laufzeit des Verfahrens. Dies führt dazu, dass die Ameisenkolonie Teile des bisherigen Suchverlaufs vergessen und die Suche somit in neuen Regionen fortsetzen kann.[134] Da der ACO-Algorithmus eine eher schlechte Tiefensuche hat, wird dieser oft mit einer LS Heuristik kombiniert, die versucht, die fertig konstruierte Lösung einer Ameise zu verbessern.[135]

Die ACO Metaheuristik ist weitaus komplexer als die VNS und GRASP Metaheuristiken. Dieser Aspekt ist an der erhöhten Anzahl an Parametern zu erkennen, die üblicherweise folgende sind: Größe der Ameisenkolonie, Pheromonverdunstungsrate, ein Parameter für den Einfluss von problemspezifischen Informationen, ein Parameter für den Einfluss der Pheromonspur und ein Abbruchkriterium. Oft hat die Art und Weise, wie und wann die Pheromonspur letztendlich aktualisiert wird, ebenfalls einen großen Einfluss auf die Funktionsweise der ACO Metaheuristik.[136]

4.8.6.4 Tabu Search

Die Tabu Search (TS) Metaheuristik basiert wiederum auf einem vergleichsweise einfachen Prinzip. Die klassische TS Metaheuristik wird vom Gründer Fred Glover als Strategie beschrieben, um innere Heuristiken, die speziell an die Problemstruktur angepasst sind, zu steuert. Diese Heuristik ist meist eine LS Heuristik. Das Grundprinzip von TS ist es, dem LS Verfahren die Möglichkeit zu geben, die Suche nach der Begegnung eines lokalen Optimums fortzusetzen, indem auch schlechtere Lösungen akzeptiert werden. Damit sich in diesem Fall das Verfahren nicht im Kreis dreht und immer wieder bereits besuchte Lösungen im Suchraum betrachtet werden, wird eine sogenannte Tabu-Liste benutzt, die die kürzlich besuchten Lösungen speichert und als Tabu erklärt.[137] Diese Liste ist sozusagen das Gedächtnis des Suchverlaufs und stellt das zentrale Element des Verfahrens

[133] vgl. Dorigo und Stützle, 2019, S.315f
[134] vgl. Zäpfel und Braune, 2005, S.110
[135] vgl. Dorigo und Stützle, 2019, S.329
[136] vgl. Dorigo und Stützle, 2019, S.318f
[137] vgl. Gendreau und Potvin, 2019, S.40

dar. Entscheidend für die Performance von TS ist die Festlegung der Länge der Tabu-Liste, die im klassischen TS neben dem Abbruchkriterium den einzigen Parameter darstellt. Eine zu kurze Tabu-Liste kann zu Zyklen im Suchverlauf führen, während eine zu lange Liste die Suche im Lösungsraum zu stark einschränkt.[138]

Eines der Hauptprobleme von Metaheuristiken wie TS, die hauptsächlich auf LS Heuristiken basieren, ist die zu lokale Suche, die dazu führt, dass die Breitensuche vernachlässigt wird. Dementsprechend ist es für TS besonders wichtig, durch andere Prinzipien eine ebenso gute Breitensuche wie Tiefensuche umzusetzen. Aus diesem Grund ist z.B. die Kombination von TS und GA sehr beliebt.[139]

4.8.6.5 Simulated Annealing

Von den Metaheuristiken, die auf einer Metapher basieren, ist Simulated Annealing (SA) eine der am einfachsten anzuwendenden Metaheuristik. Inspiriert ist dieses Verfahren von der Physik bzw. dem Gebiet der Thermodynamik. Es werden die Auswirkungen der Temperaturverminderung auf die atomare Struktur eines Körpers betrachtet. Ziel eines solchen Abkühlungsprozesses ist es, ausgehend von einem Zustand hoher Energie einen Zustand minimaler Energie zu erreichen, in dem die Atome eine perfekte kristalline Struktur aufweisen. Dabei ist die Geschwindigkeit des Abkühlvorgangs entscheidend. Wird die Temperatur zu schnell verringert, können sich die Atome nicht optimal anordnen und es entsteht eine sogenannte polykristalline bzw. amorphe Struktur, die nicht dem Zustand minimaler Energie entspricht. Bei genügend langsamer Temperaturabsenkung, wie es bei natürlichen Abkühlvorgängen der Fall ist, sodass die Atome genügend Zeit haben, einen Gleichgewichtszustand für die jeweilige Temperatur zu erreichen, ist es möglich, den Zustand minimaler Energie zu erreichen. Die Temperatur sollte also möglichst langsam und in kleinen Schritten verringert werden. Ein wichtiges Merkmal dieser Vorgehensweise ist die Möglichkeit des Auftretens von zeitweise wieder erhöhter Energie während des Abkühlprozesses.[140] Das bedeutet, dass sich die Energie eines Körpers während eines Abkühlprozesses nicht stetig verringert, sondern die Atome ab und zu wieder einen Zustand erhöhter Energie einnehmen müssen, um letztendlich den zwischenzeitlichen Gleichgewichtszustand erreichen zu können. Es ist also nicht jede neue Anordnung der Atome zwingend mit einer Reduzierung der Energie verbunden.

Dieses Prinzip zwischenzeitlich wieder erhöhte Zustände der Energie zuzulassen, um letztendlich den Zustand der minimalen Energie erreichen zu können, eignet sich sehr gut für das Lösen von Optimierungsproblemen und ist als Metropolis Algorithmus (MA) bekannt. Dieser Grundsatz ermöglicht es während dem Optimierungsvorganges das schon vielseitig angesprochene Problem des Steckenbleibens in einem lokalen Optimum zu

[138] vgl. Zäpfel und Braune, 2005, S.89f
[139] vgl. Gendreau und Potvin, 2019, S.46ff
[140] vgl. Zäpfel und Braune, 2005, S.71

vermeiden. Kombiniert mit einer LS Heuristik entsteht die SA Metaheuristik. Die LS Heuristik ist, wie schon oft beschrieben, dafür zuständig, die lokale Nachbarschaft einer Lösung zu durchsuchen, wobei diese durch den MA gesteuert wird. Der MA steuert die LS Heuristik über das sogenannte Metropolis Kriterium, das die Wahrscheinlichkeit angibt, mit welcher eine schlechtere Lösung akzeptiert wird (bessere Lösungen werden immer akzeptiert). Diese Wahrscheinlichkeit ist abhängig von der Temperatur und nimmt mit der Reduktion dieser ebenfalls ab. Das bedeutet, dass zu Beginn des Optimierungsvorgangs bei hoher Temperatur es sehr wahrscheinlich ist, dass eine schlechtere Lösung akzeptiert wird. Gegen Ende des Optimierungsvorganges wird die Wahrscheinlichkeit zunehmend geringer, bis zu dem Zeitpunkt an dem ausschließlich bessere Lösungen akzeptiert werden.[141] Die Anfangsphase von SA ist somit vergleichbar mit einer zufälligen Durchsuchung des Lösungsraums, wohingegen die Endphase mit einer rein lokalen Suche gleichzusetzen ist.[142]

Der Optimierungsablauf mittels SA wird oft mit einer sogenannten Markov-Kette verglichen, die eine Folge von Zufallsversuchen darstellt, sodass der Ausgang jedes einzelnen Zufallsversuchs nur vom Ausgang des vorherigen Zufallsversuchs abhängig ist. Im Kontext der Optimierung bedeutet das, dass die zufällige Akzeptanz einer neuen Lösung nur von der davor betrachteten Lösung abhängt (und der gegebenen Temperatur). Weiter zurückliegende Lösungen haben also keinen Einfluss mehr auf die Akzeptanz der neuen Lösung. Der Vergleich von SA mit einer Markov-Kette deutet auf die Gedächtnislosigkeit der SA Metaheuristik hin. Die Länge einer Markov-Kette kann als die Anzahl der betrachteten Lösungen gesehen werden, bevor die nächste Temperaturreduzierung eingeleitet wird. Am Beispiel des Abkühlprozesses eines Körpers ist die Länge einer Markov-Kette vergleichbar mit der Zeit, die die Atome haben, um sich neu auszurichten und somit den Gleichgewichtszustand für die aktuelle Temperatur zu erreichen. SA lässt sich mit vier Parametern steuern: der Anfangstemperatur, der Verminderungsvorschrift für die Temperatur, der Länge einer Markov-Kette und einem Abbruchkriterium.[143]

SA ist besonders gut geeignet für Probleme, die eine simulationsbasierte Evaluation der Lösungen benötigen, wodurch oft eine erhebliche Menge an Speicherplatz erforderlich ist. In diesen Fällen sind populationsbasierte Verfahren weniger geeignet.[144] Zusammengefasst lässt sich sagen, dass SA ohne jegliche zusätzliche Prinzipien eine Mischung aus Breiten- und Tiefensuche umsetzt und durch das Metropolis Kriterium das Entkommen eines lokalem Optimums möglich ist. Das Prinzip, eine schlechtere Lösung mit einer gewissen Wahrscheinlichkeit zu akzeptieren, ist definitiv eine Schlüsselkomponente von SA. Ein Nachteil ist die Gedächtnislosigkeit des Verfahrens, die eine Kombination mit TS naheliegend macht.

[141] vgl. Delahaye u. a., 2019, S.3ff
[142] vgl. Zäpfel und Braune, 2005, S.73
[143] vgl. Zäpfel und Braune, 2005, S.75ff
[144] vgl. Delahaye u. a., 2019, S.19

4.8.6.6 Genetischer Algorithmus

Der klassische Genetische Algorithmus (GA), entwickelt von John Holland (1975)[145], basiert auf der darwinistischen Evolutionstheorie. Dieser Prozess der natürlichen Evolution hat sich als sehr leistungsstark erwiesen, vor allem in Hinblick auf die Diversität und Anpassung von Lebewesen auf Erden. In einem natürlichen System ist die Umwelt durch einen ständigen Veränderungsprozess geprägt, der Individuen dazu zwingt, sich an diese anzupassen.[146] Inspiriert von diesem Prinzip, zielt der GA darauf ab, den Prozess der natürlichen Evolution bei Optimierungsproblemen anzuwenden. Dafür ist es nötig, die Grundmechanismen durch sogenannte Operatoren (können auch als Heuristiken gesehen werden) abzubilden, nämlich Selektion, Mutation und Rekombination (auch oft *crossover* genannt). Diese Mechanismen werden iterativ auf Lösungskandidaten für ein gegebenes Problem angewendet, wobei immer eine größere Anzahl an unterschiedlichen Lösungskandidaten gleichzeitig betrachtet werden. Dementsprechend ist der GA auch eine populationsbasierte Metaheuristik. Die Lösungskandidaten einer Population werden in diesem Fall meistens als Individuen bezeichnet. In jeder Iteration des Verfahrens wird aus einer bestehenden Population, durch die bereits genannten Operatoren, eine neue Population erzeugt, wobei jede aufeinanderfolgende Population als Generation bezeichnet wird.[147] Der Optimierungsverlauf des GA kann, so wie die Evolution, als '*survival of the fittest*' angesehen werden. So setzt sich gutes Genmaterial bevorzugt durch und führt zu einer Optimierung der Individuen im Verlauf vieler Generationen. Der gesamte Aufbau jedes Lebewesens, dessen Organe, das Aussehen und sogar geistige Fähigkeiten sind durch das Genmaterial vorgegeben.[148]

Wie schon erwähnt, ist der GA eine populationsbasierte Metaheuristik. Jedes Individuum in der Population stellt eine zulässige Lösung (einen Produktionsplan) dar. Jede Lösung wird im GA durch Gene (Variablen) repräsentiert. Im Falle der Problemstellung dieser Diplomarbeit stellt jedes Gen einen Auftrag und dessen Maschine, auf der dieser eingeplant wurde, dar. Die Gesamtheit aller Gene eines Individuums stellen somit das Erbgut und den dazugehörigen Produktionsplan dar. So, wie sich in der Evolution gutes Erbmaterial bevorzugt durchsetzt, setzen sich im Laufe der Generationen überwiegend die Produktionspläne mit geringeren Kosten durch. Der GA startet mit einer initialen Population, die eine beliebige Anzahl an verschiedenen Individuen und somit Produktionsplänen beinhaltet. Jeder Produktionsplan wird durch eine Zielfunktion bewertet, die die entstehenden Kosten des Produktionsplans angibt. Die Operatoren kommen dann in folgender Reihenfolge zum Einsatz:

Selektion

[145] Holland, J. (1975): *Adaptation in Natural and Artificial Systems*, Cambridge, University of Michigan Press.
[146] vgl. Whitley, 2019, S.245
[147] vgl. Zäpfel und Braune, 2005, S.43
[148] vgl. Heistermann, 1994, S.11ff

Die Selektion ist dafür zuständig, für den Rekombinations- und Mutationsoperator Produktionspläne auszuwählen. Dementsprechend ist die Selektion ein Vorgang, bei dem jene Individuen aus der aktuellen Population ausgewählt werden, die einerseits als Elternteil für den Rekombinationsprozesses dienen und andererseits eine Mutation erfahren. Der Selektionsprozess ist, so wie in der Evolution, stochastischer Natur. So, wie sich nicht ausschließlich die besten Individuen miteinander paaren, erfahren auch nicht ausschließen die besten Individuen eine Mutation. Dementsprechend ist die Selektion vom Fitnesswert (den entstehenden Kosten eines Produktionsplans) der Individuen abhängig. Umso geringer die entstehenden Kosten des Produktionsplans sind, desto höher ist die Wahrscheinlichkeit, dass dieser vom Selektionsprozess ausgewählt wird und überlebt.[149]

Rekombination

Die Rekombination des GA stellt ein vielversprechendes Prinzip dar, das in vielen anderen Metaheuristiken nicht vorhanden ist. Zwei Individuen, die durch den Selektionsoperator ausgewählt wurden, werden durch die Rekombination miteinander gekreuzt. Diese Kreuzung ist dafür zuständig, das Erbmaterial (Teillösungen) der ausgewählten Individuen zu kombinieren, um neue und möglicherweise noch bessere Individuen zu erzeugen.[150] Dementsprechend werden Teillösungen von zwei Produktionsplänen miteinander kombiniert, in der Hoffnung einen Produktionsplan mit noch geringeren Kosten zu erstellen. Besteht ein Produktionsplan z.B. aus insgesamt 100 Aufträgen, können durch die Rekombination die ersten 30 Aufträge so eingeplant werden, wie sie beim ersten Produktionsplan (erster Elternteil) eingeplant wurden und die restlichen 70 Aufträge so eingeplant werden, wie sie beim zweiten Produktionsplan (zweiter Elternteil) eingeplant wurden. Ziel der Rekombination sollte es sein, vielversprechende Teile des Erbgutes von zwei verschiedenen Produktionsplänen zu kombinieren. Die größte Herausforderung besteht somit darin, diese vielversprechenden Teile zu identifizieren, was unter anderem durch das Nutzen von problemspezifischen Informationen erreicht werden kann.

Mutation

Der Mutationsoperator verändert das Erbgut eines, durch die Selektion ausgewählten, Individuums nur geringfügig an meist zufälligen Genen. Dadurch wird, wie auch in der Natur, eine Diversität in der Population erzielt.[151] Die Aufrechterhaltung der Diversität ist beim GA ein kritischer Punkt, da durch zu geringe Diversität die Rekombination nicht mehr effektiv funktionieren kann. Ist z.B. in allen Produktionsplänen einer Population ein Auftrag immer ausschließlich auf der gleichen Maschine eingeplant, ist es durch Rekombination nicht möglich, einen Produktionsplan zu erstellen, in dem der Auftrag auf einer anderen Maschine eingeplant ist. Dieses Problem kann durch Mutation gelöst werden, indem sie einen Produktionsplan so verändert, dass dieser Auftrag auf einer anderen Maschine

[149] vgl. Zäpfel und Braune, 2005, S.43f
[150] vgl. Zäpfel und Braune, 2005, S.45
[151] vgl. Zäpfel und Braune, 2005, S.45ff

eingeplant wird. Dieser Produktionsplan hat dann wiederum die Möglichkeit, sich in der Population durch Rekombination fortzupflanzen.

Jede neue Generation ergibt sich durch die, von den Operatoren erstellten, neuen Individuen. An diesem Punkt wird jedes Individuum der neuen Generation wieder durch die Zielfunktion bewertet und der Prozess beginnt von neuem mit dem Selektionsprozess. Damit die besten Individuen durch den stochastischen Prozess der Selektion nicht verloren gehen, wird oft die sogenannte Reproduktion (auch oft *elitisme* genannt) benutzt, die die besten Individuen ohne Veränderungen von einer Generation in die nächste übernimmt.[152] Die Wahl der Kodierung (wie eine Lösung in den Genen dargestellt wird) sowie die Definition der zugehörigen Operatoren für ein Problem ist ein entscheidender Faktor für die Performance des GA.[153] Dementsprechend ist es ausschlaggebend, wie bereits erwähnt, dass die Operatoren des GA sowohl problemspezifische Strukturen als auch Informationen effektiv nutzen.

Der GA hat üblicherweise folgende Parameter: Größe der Population, Rekombinationsrate (wie viel Individuen einer neuen Generation durch Rekombination erstellt werden) und ein Abbruchkriterium. Jedoch können die Eigenschaften der Operatoren oft auch als Parameter angesehen werden und spielen eine dementsprechend große Rolle in der Effektivität des GA.[154] Zusätzlich spielt die genaue Zusammenstellung der initialen Population eine nicht zu vernachlässigende Rolle. Die initiale Population sollte sowohl möglichst gute Individuen enthalten als auch eine hohe Diversität aufweisen.[155] Des Weiteren ist auch oft die Anzahl der Reproduktionen ein Parameter. Diese vielen Parameter bzw. Eigenschaften machen den GA zu einer komplexen Metaheuristik, die schwer nachzuvollziehende Dynamiken mit sich bringen kann.

Generell sind populationsbasierte Metaheuristiken um einiges komplizierter in der Anwendung und Entwicklung als Metaheuristiken, die nur eine Lösung auf einmal betrachten. Dies liegt meistens daran, dass die Dynamiken, die in einer Population entstehen, nur schwer einzuschätzen sind. Am kritischsten ist hierbei die Entwicklung einer effektiven Rekombination, da diese in populationsbasierten Metaheuristiken oft ein zentrales Element darstellt. Ist eine sinnvolle bzw. effektive Rekombination nicht möglich bzw. zu aufwändig, sollte vorzugsweise eine Metaheuristik benutzt werden, die nur eine Lösung auf einmal betrachtet.[156] Dennoch stellt der Prozess der Rekombination, wenn effektiv umgesetzt, ein sehr leistungsfähiges Prinzip dar.

Zusammengefasst ist das Prinzip der Rekombination definitiv ein Schlüsselelement des GA. Aber auch die Selektion, die die Wahrscheinlichkeit der Auswahl von Individuen abhängig von ihrem Fitnesswert gestaltet, ist ein interessantes Prinzip. Das Nutzen einer

[152] vgl. Falkenauer, 1998, S.40
[153] vgl. Zäpfel und Braune, 2005, S.46
[154] vgl. Zäpfel und Braune, 2005, S.51
[155] vgl. Mutingi und Mbohwa, 2017, S.51f
[156] vgl. Lourenço u. a., 2019, S.160

Population macht den GA zu einer Metaheuristik, die eine gute Breitensuche umsetzen kann. Die Tiefensuche zählt wiederum nicht zu den Stärken des GA. Dementsprechend ist die Kombination mit Prinzipen, die eine effektive Tiefensuche umsetzen, naheliegend.

4.9 Zusammenfassung und Auswahl der Metaheuristik

Nachdem einige vielversprechende Metaheuristiken betrachtet wurden, lassen sich folgende Punkte zusammenfassen:

- Jede Metaheuristik hat ihre Stärken und Schwächen, die sich durch die eingesetzten Prinzipien ergeben. Demzufolge ist es oft empfehlenswert, eine Kombination mehrerer Prinzipien von verschiedenen Metaheuristiken zu nutzen. Aussagen wie: "Diese Metaheuristik ist die beste", sind unschlüssig. Vielmehr sind Aussagen wie: "Diese Kombination von Prinzipien ist vielversprechend" sinnvoller.
- Der Schlüsselfaktor liegt also in der geeigneten Kombination von Prinzipien. Diese Kombination sollte sowohl eine effektive Breiten- sowie Tiefensuche umsetzen können.
- Jede noch so vielversprechende Kombination von Prinzipien ist unbrauchbar, wenn problemspezifische Strukturen und Informationen nicht effektiv genutzt werden (siehe auch Schlussfolgerung der NFL Theorie).

Die Frage, die sich nun stellt, ist, welche Metaheuristik bzw. Metaheuristiken für das Problem dieser Arbeit am besten geeignet sind? Wie bereits erwähnt, ist es nur schwer möglich, die Antwort darauf zu begründen. Trotzdem wird in dieser Arbeit in erster Linie die Anwendung des GA herangezogen. Dies liegt einerseits daran, dass der GA in der Literatur bereits auf einer Vielzahl von komplexen und großen Problemen erfolgreich angewendet wurde und z.B. auch bei SAP in der Praxis angewendet wird. Dieses Argument trifft jedoch auch auf andere Metaheuristiken zu.[157] Zusätzlich ist das nicht unbedingt, wie durch die NFL Theorie nahegelegt, ein erfolgsversprechendes Argument, das den Erfolg des GA für das Problem dieser Diplomarbeit garantiert. Deswegen wird der GA auch nur als Fundament benutzt, mit dem anschließend weitere Prinzipien kombiniert werden können.

Durch die letztendlich intensive Auseinandersetzung mit dem GA ist ein Vergleich mit einer zweiten Metaheuristik im Rahmen dieser Arbeit leider nicht möglich. Um hier einen fairen Vergleich aufstellen zu können, müsste annähernd so viel Zeit in die Entwicklung der zweiten Metaheuristik investiert werden, wie für den GA. Dennoch wird der endgültige GA mit einer zufälligen Durchsuchung des Lösungsraums und dem Einsatz einer alleinigen LS Heuristik verglichen, um zu untersuchen, ob die Anwendung einer Metaheuristik für die Problemstellung dieser Diplomarbeit tatsächlich bessere Ergebnisse liefert.

[157] z.B. wird ACO von Antoptima© in der Praxis ebenfalls erfolgreich eingesetzt (https://www.antoptima.com).

5 Kapitel 5

Anpassung der Metaheuristik an die Projektdaten

Im letzten Kapitel wurde anhand der Komplexitätstheorie erläutert, dass es sogenannte NP-schwere Probleme gibt, die so komplex sind, dass sie nicht exakt gelöst werden können. Dies liegt vor allem daran, dass die Anzahl der Lösungen im Lösungsraum mit der Größe des Inputs exponentiell zunimmt. Des Weiteren wurden Heuristiken und Metaheuristiken präsentiert, die Algorithmen zum Lösen dieser NP-schweren Problemen darstellen. In diesem Kapitel wird eine detaillierte Beschreibung und Übersicht der Problemstellung dieser Diplomarbeit präsentiert. Des Weiteren wird der Batchprozess sowie die Anpassungen der ausgewählten Metaheuristik an die Projektdaten durchgeführt.

5.1 Projektdaten

Das Produktionsproblem dieser Diplomarbeit ergibt sich aus den Aufträgen eines kunststoffverarbeitenden Unternehmen. Alle noch nicht verarbeiteten Aufträge und die dazu benötigten Informationen werden in einem Excel-File bereitgestellt.

5.1.1 Aufbau des Excel-Files

Die Informationen des Excel-Files sind in fünf Arbeitsblätter gegliedert, die folglich kurz beschrieben werden.

5.1.1.1 Bestellungen

Eine Bestellung wird durch sechs Informationen charakterisiert:

1. Die **Artikelnummer**, die zur Identifizierung des Artikels dient.
2. Die **Charge**, die die Farbe des bestellten Artikels darstellt.
3. Die bestellte **Menge** in Stück des jeweiligen Auftrags.
4. Das **Lieferdatum**, an dem der Auftrag versendet werden muss.
5. Die **Lieferzeit** am Lieferdatum. Diese ist für alle Aufträge um 6 Uhr in der Früh.

6. Die **Lieferzeit in Stunden**, die vom Optimierungsstart (erster Tag 0 Uhr) bis zum genauen Lieferzeitpunkt (6 Uhr am Lieferdatum) vorhanden sind. Diese Zeit repräsentiert nur die Zeit, die für die Produktion zur Verfügung steht. Dementsprechend nur von Montag bis Freitag jeden Tag 22.8 Stunden (5% jedes Tages (1.2 Stunden) sind für die Instandhaltung reserviert und somit nicht für die Produktion disponibel).

Für das Problem dieser Arbeit werden insgesamt 714 Bestellungen betrachtet, die sich aus 89 verschiedenen Artikelnummern zusammenstellen. Die erste Bestellung ist am 9.07.2018 und die letzte Bestellung am 30.05.2019 fällig. Das ergibt insgesamt einen Zeitraum von 233 Werktagen zwischen diesen beiden Lieferdaten.

5.1.1.2 Bestände

Die Lagerbestände zu Beginn der Optimierung sind durch folgende Punkte charakterisiert:

1. Die **Artikelnummer**.
2. Die Farbe des Artikels (**Charge**).
3. Die Menge in Stück, die auf Lager ist (**Bestand**).
4. Die Menge in Stück, die auf eine Palette passt (**Stück pro Palette**).
5. Die Menge in Stück, die in eine Lage auf einer Palette passt (**Runden auf**). Eine volle Palette besteht oft aus mehreren Lagen. Diese Kennzahl wird benötigt, da nur in ganzen Lagen geliefert wird.

5.1.1.3 Maschinenkosten

Es gibt insgesamt 17 Maschinen, eingeteilt in drei Maschinengruppen (2, 3 und 3SL). Das Arbeitsblatt ist folgendermaßen aufgebaut.

1. Die **Maschinennummer**.
2. Die **Maschinengruppe**, in der sich die jeweilige Maschine befindet.
3. Der **Stundensatz**, der die Betriebskosten jeder einzelnen Maschine angibt.

Die Maschinengruppe 2 ist mit einem Stundensatz von 3.5 die billigste Maschinengruppe und beinhaltet zehn Maschinen (Nr. 5-13 und 17). Danach folgt die Maschinengruppe 3 mit vier Maschinen (Nr. 1-3 und 15) und einem Stundensatz von 5. Die Maschinengruppe 3SL ist mit einem Stundensatz von 7 die teuerste Maschinengruppe und beinhaltet 3 Maschinen (Nr. 4, 14 und 16).

5.1.1.4 Maschinenliste

Die Maschinenliste gibt die möglichen Maschinen jedes Artikels an, auf denen dieser eingeplant werden kann. Da keine Dokumentation vorhanden ist, welche Artikel auf welchen Maschinen produziert werden können, besteht die Maschinenliste aus den Daten vergangener, bereits durchgeführter Aufträge und beinhaltet folgende Punkte:

1. Die **Artikelnummer**.
2. Die **Maschinennummern**, auf denen dieser Artikel schon einmal produziert wurde.
3. Die **Anzahl der Aufträge**, die von dem jeweiligen Artikel auf der jeweiligen Maschine schon produziert wurden.

5.1.1.5 Arbeitsplan

Der Arbeitsplan gibt an, mit welchen Werkzeugen die jeweiligen Artikel produziert werden können und besteht aus folgenden Punkten:

1. **Artikelnummer**.
2. Die **Nummer der möglichen Werkzeuge**, mit denen der jeweilige Artikel produziert werden kann.
3. Die **Rüstzeit** in Minuten des jeweiligen Werkzeuges.
4. Die **Zykluszeit** des jeweiligen Werkzeuges in Sekunden, die benötigt wird, um ein Stück des jeweiligen Artikels zu produzieren.
5. Ob das jeweilige Werkzeug die Fähigkeit besitzt, mehrere Artikel gleichzeitig zu produzieren (**Kombi-Werkzeug**). Wegen der in diesem Fall bereits angesprochenen Erhöhung der Zykluszeit (siehe Abschnitt 4.4.3) wird diese Eigenschaft nicht berücksichtigt.

Es gibt Werkzeuge die für mehrere Artikel eingesetzt werden können, so wie es auch Artikel gibt, die mit mehreren Werkzeugen produziert werden können.

5.1.2 Merkmale und Annahmen

Folgende Parameter stellen die Hauptparameter des Problems dar:

- **Tage = 300** : Dieser Parameter gibt die Dauer des betrachteten Planungshorizonts an. Alle Bestellungen, deren Lieferdatum sich innerhalb dieses Planungshorizonts befinden, werden in der Optimierung berücksichtigt.
- **AnzahlMaschinen = 17** : Gibt die Anzahl der Maschinen an, die zur Verfügung stehen.

- **Umrueststart = 6** : Gibt die Uhrzeit an, ab der ein Rüstvorgang beginnen darf. Dementsprechend kann an jedem Werktag erst ab 6 Uhr gerüstet werden.
- **Umruestende = 19** : Gibt die Uhrzeit an, bis zu welcher ein Rüstvorgang abgeschlossen sein muss. Folglich muss jeder Rüstvorgang bis 19 Uhr beendet sein.
- **AnzahlRuestMitarbeiter = 4** : Stellt die Anzahl der Rüstmitarbeiter dar, die für Rüstvorgänge zur Verfügung stehen.
- **StundenTag = 22.8** : Dieser Parameter stellt die Anzahl der Stunden dar, die jeden Werktag für die Produktion zur Verfügung stehen.
- **Batchrange = 10** : Dieser Parameter stellt die Anzahl der Tage dar, innerhalb welcher Bestellungen zusammengefasst werden dürfen. Um Rüstvorgänge zu vermeiden, kann es von Vorteil sein, Bestellungen, die mit dem gleichen Werkzeug produziert werden können, zusammenzufassen.
- **Ausschussquote = 5** : Gibt die Ausschussquote bei einem Farbwechsel an.

Des Weiteren gelten für die Optimierung folgende Annahmen:

- Alle 17 Maschinen sind am ersten Tag ab 6 Uhr einsatzbereit.
- Am ersten Tag muss für jeden Auftrag gerüstet werden, da nicht bekannt ist, welche Werkzeuge auf welchen Maschinen zum Startzeitpunkt gerüstet sind.
- Rüstmitarbeiter werden am ersten Tag nicht berücksichtigt.
- Es entstehen keine Lageraufträge, die den Lagerstand auf einem Minimum halten.
- Es wird nicht überproduziert, falls ein Auftrag außerhalb des Rüstintervalls fertig wird und für den nächsten Auftrag gerüstet werden muss. In diesem Fall steht die Maschine bis zum Umrüststart still.
- Einmal begonnen, darf ein Auftrag nicht mehr unterbrochen und zu einem späteren Zeitpunkt fortgesetzt werden.
- Es gibt keine Mindestproduktionsmenge.
- In der Praxis werden beim betrachteten Unternehmen die Produktionsmengen eines Auftrages auf ganze Lagen aufgerundet. Dies wird nicht berücksichtigt, da sonst Bestandsmengen, die nicht in ganzen Lagen sind, während des Optimierungsvorganges nicht berücksichtigt werden können.
- Kommissioniervorgänge werden nicht berücksichtigt. Dementsprechend ist ein Auftrag lieferbereit, sobald er fertig produziert worden ist.
- Sämtliche Wegzeiten werden vernachlässigt.
- Es gibt keine allgemeine Ausschussquote, nur die Ausschussquote, die bei einem Farbwechsel stattfindet.

- Die Lagerbestände werden am genauen Fertigstellungsdatum und Lieferdatum aktualisiert.
- Aufträge, die sich verspäten, werden sofort nach Fertigstellung geliefert. Dementsprechend erzeugen Produktionsmengen bei einer Verspätung keine Lagerkosten.
- Feiertage unter der Woche werde nicht berücksichtigt und sind somit ganz normale Werktage.

5.2 Initialisierungsphase vor der Optimierung

Bevor überhaupt mit der Optimierung begonnen werden kann, müssen die benötigten Informationen für die Produktion zusammengestellt werden.

5.2.1 Zusammenstellung der Produktionsdaten

In einem ersten Schritt werden die Produktions- und Bestandsdaten aller Aufträge und Artikel zusammengestellt.

5.2.1.1 Produktionsdaten

Um während des Optimierungsvorganges Suchvorgänge und Berechnungen auf ein Minimum zu begrenzen, werden bereits im Vorhinein alle benötigten Daten für jeden Auftrag zusammengestellt und berechnet. Des Weiteren wird überprüft, ob überhaupt eine Produktion für jede Bestellung notwendig ist. Dafür wird von der ersten Bestellung aus die Liefermenge jeder Bestellung mit den Beständen verglichen. Ist eine Bestellung über den Bestand abgedeckt, wird diese im Einplanungsprozess der Optimierung nicht mehr berücksichtigt. In diesem Fall wird aber durchaus der Verlauf des Bestandes am Lieferdatum reduziert, um die maximale Palettenanzahl während des Optimierungsvorganges richtig zu erfassen.

Von den insgesamt 714 Bestellungen im Excel-File ist nur für 540 Bestellungen eine Produktion nötig. Dementsprechend bleiben von den insgesamt 89 verschiedenen Artikeln der gesamten Bestellungen nur mehr 52 verschiedene Artikel übrig, für die eine Produktion stattfindet. Für diese 52 verschiedenen Artikel werden insgesamt von den 299 nur 41 verschiedene Werkzeuge benötigt.

5.2.1.2 Bestandsdaten

Der Bestand kann in drei verschiedene Arten von Artikel aufgeteilt werden:

1. Konstanter Bestand von Artikeln, die nicht in den Bestellungen vorkommen. Dieser Verlauf ist unabhängig von der Optimierung.

2. Fix-Variabler Bestand von Artikeln, die in den Bestellungen vorkommen, aber keine Produktion benötigen. Dieser Verlauf ist ebenfalls unabhängig von der Optimierung. Dieser Bestand ist fix, weil er unabhängig von der Optimierung ist, aber variabel, weil durch die Reduzierung an den Lieferdaten der Bestand nicht konstant ist.

3. Variabler Bestand von Artikeln, die in den Bestellungen vorkommen und zumindest eine Bestellung eine Produktion benötigt. Dieser Bestand kann wiederum in Bestellungen aufgeteilt werden, die:

   a) keine Produktion benötigen. Dieser Verlauf ist ebenfalls unabhängig von der Optimierung.

   b) eine Produktion benötigen. Dieser Verlauf ist vom Optimierungsprozess abhängig.

Dementsprechend ist nur der Verlauf des Bestandes 3.b) vom Optimierungsvorgang abhängig. Die Summe aller drei verschiedenen Artikel ergibt dann den Gesamtverlauf des Bestandes. Hier muss jedoch darauf geachtet werden, dass die Gesamtpaletten richtig berechnet werden. Da die Artikel auf den Paletten nicht durchgemischt werden dürfen, müssen die Gesamtpaletten jedes einzelnen Artikels in jeder Periode immer aufgerundet werden. In diesem Fall können der Verlauf der Gesamtpaletten der Artikel des 1. und 2. Teils des Bestandes schon vor der Optimierung aufgerundet werden. Der Verlauf der Gesamtpaletten der Artikel des 3. Teils des Bestandes kann jedoch erst nach dem Optimierungsprozesses aufgerundet werden, wenn der Verlauf des Bestandes 3.b) bekannt ist. Hier muss besonders darauf geachtet werden, dass der Bestand von Aufträgen, die nur teilweise über den Bestand abgedeckt sind, während des Optimierungsprozesses richtig berücksichtigt wird. Da einerseits ein initialer Lagerbestand vorhanden ist und andererseits eine produzierte Menge zu diesem initialen Lagerbestand dazu kommt.

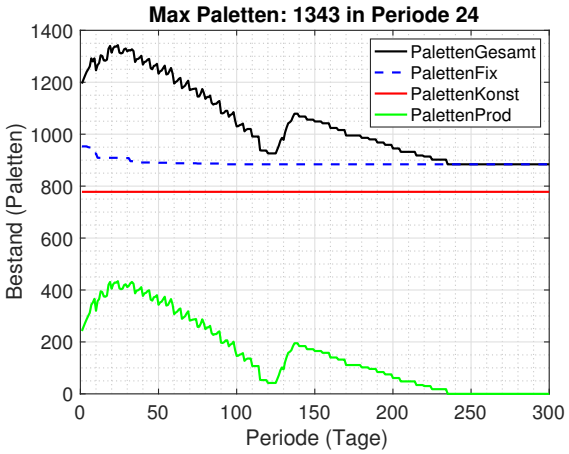

Abbildung 5.1: Beispiel Zusammensetzung des Bestandes

Abbildung 5.1 zeigt ein Beispiel, wie der Verlauf des Bestandes aussehen kann. In Rot ist der konstante Verlauf der Artikel ohne Bestellung zu sehen (1. Teil des Bestandes). In Blau

ist der gemeinsame Verlauf des 1. und 2. Teils des Bestandes zu sehen, die unabhängig von der Optimierung sind. Des Weiteren ist in Grün der alleinige Verlauf des Bestandes der Artikel mit Produktion zu sehen (3. Teil des Bestandes). Da keine Lageraufträge entstehen, verschwindet dieser Anteil, nachdem alle Lieferungen druchgeführt wurden. Schließlich ist in Schwarz der Verlauf des gesamten Bestandes zu sehen, wobei in diesem Beispiel der maximale Bestand in Periode 24 mit insgesamt 1343 Paletten vorzufinden ist. Der maximale Bestand ist für die Praxis eine wichtige Information, da oft nur eine maximale Anzahl an Palettenplätzen zur Verfügung steht.

5.2.2 Batchprozess

Durch den Batchprozess wird die Reihenfolge, in der die Aufträge den Maschinen zugeordnet werden, festgelegt. Das Zusammenlegen von Aufträgen (batchen), die mit dem gleichen Werkzeug produziert werden können, ist notwendig, weil die Rüstvorgänge eine nicht zu vernachlässigende Zeit in Anspruch nehmen. Außerdem wird durch den Batchprozess der Lösungsraum verkleinert (siehe Abschnitt 6.2.1). Die Zeit eines Rüstvorganges ist vom benutzten Werkzeug abhängig und kann für die 41 möglichen Werkzeuge 45 bis 180 Minuten dauern. Das betrachtete Problem dieser Diplomarbeit zeigt sehr gut, weshalb es in der Praxis oft nicht möglich ist, eine gleichmäßige Produktion der Artikel umzusetzen (siehe auch Absatz 2.5.1). Die Rüstvorgänge nehmen eine erhebliche Zeit in Anspruch, sodass eine gleichmäßige Produktion mit einer hohen Anzahl an Rüstvorgängen verbunden wäre, die zu großen Verschwendungen führen würde. In solchen Fällen ist das Batchen von Aufträgen üblich, wodurch zwar die Verschwendungen der Rüstvorgänge in Grenzen gehalten werden, jedoch unter anderem erhöhte Lagerkosten entstehen. Des Weiteren steigt durch den Batchprozesses auch die Wahrscheinlichkeit einer Verspätung.

Das in dieser Diplomarbeit betrachtete Unternehmen batcht in der Praxis Aufträge, die innerhalb einer Batchrange von bis zu 10 Tagen liegen. Das bedeutet, dass Aufträge, die mit dem gleichen Werkzeug produziert werden können und deren Lieferdatum nicht weiter als 10 Tage auseinander liegt, zusammengelegt werden, um Rüstvorgänge zu vermeiden. Diese Batchrange wird für den Anfang so übernommen. Werden von den 540 Aufträgen, die eine Produktion benötigen, durch einen Batchprozess alle möglichen Aufträge innerhalb der Batchrange zusammengelegt, so entstehen insgesamt 201 Batches. In diesem Fall wurde vom ersten Auftrag mit dem frühesten Lieferdatum ausgegangen und die Aufträge nach dem Prio-Werkzeug gebatcht, was das erste bzw. schnellere Werkzeug darstellt, falls es mehrere mögliche Werkzeuge zur Auswahl gibt.

Um die Rechenschritte während des Optimierungsprozesses zu reduzieren, werden alle notwendigen Informationen eines Batches, die für die Erstellung eines Produktionsplans benötigt werden, in einer *BatchKanban*-Matrix gespeichert. Die detaillierte Zusammenstellung jedes Batches wird in einer eigenen Batchtabelle gespeichert. Die Informationen der

Batchtabelle werden später benötigt, um die entstandenen Kosten der einzelnen Aufträge in einem Batch zu berechnen. In der *BatchKanban*-Matrix sind folgende Informationen enthalten:

- Nummer und Rüstzeit der möglichen Werkzeuge. Es werden nur die Werkzeuge angegeben, mit denen auch wirklich alle Aufträge in einem Batch produziert werden können.
- Gesamtproduktionszeit jedes Batches. Die Gesamtproduktionszeit berücksichtigt innerhalb eines Batches die Ausschussquote bei einem Farbwechsel zwischen zwei Aufträgen mit unterschiedlichen Farben. Für den ersten Auftrag im Batch wird immer von einer Ausschussquote ausgegangen. Falls für einen Batch zwei Werkzeuge möglich sind, werden auch zwei Gesamtproduktionszeiten gespeichert.

Des Weiteren wurde für jeden Batch die späteste Startzeit (mit Prio-Werkzeug) berechnet, sodass es bei keinem der Aufträge im Batch zu einer Verspätung kommt. Die Batches innerhalb der *BatchKanban*-Matrix wurden dann aufsteigend nach diesen spätesten Startzeiten sortiert. Ist ein langsameres Alternativwerkzeug für einen Batch möglich, wird eine alternative späteste Startzeit dokumentiert. Die Batches bleiben jedoch weiterhin in der *BatchKanban*-Matrix nach den spätesten Startzeiten mit dem Prio-Werkzeug sortiert.

5.2.3 Maschinenliste der Werkzeuge

Da die Maschinenlisten aus dem Excel-File (siehe Abschnitt 5.1.1.4) abhängig von den Artikeln sind, jedoch die Batches anhand eines gemeinsamen Werkzeuges erzeugt werden, können diese Maschinenlisten nicht mehr vollständig benutzt werden. Um den Einplanungsprozess nicht zu sehr einzuschränken, wird die Maschinenliste eines Werkzeuges unter folgenden Annahmen erstellt:

- Ein Artikel kann auf all jenen Maschinen produziert werden, auf denen ein mögliches Werkzeug gerüstet werden kann.
- Ein Werkzeug kann auf all jenen Maschinen gerüstet werden, die sich in den Maschinenlisten der Artikel befinden, die mit diesem Werkzeug produziert werden können.
- Da im Excel-File die Maschinenliste eines Artikels unabhängig vom Werkzeug ist, ist die Maschinenliste für Prio- und Alternativwerkzeug eines Artikels immer gleich.

Die Maschinenliste eines Werkzeuges ergibt sich somit aus allen Maschinenlisten von allen Artikeln, die mit diesem oder einem Alternativwerkzeug produziert werden können.

5.2.4 Zeit zwischen Optimierungsstart und erstem Lieferdatum

Der Zeitpunkt des Optimierungsstarts ist nicht vorgegeben, wobei dieser mindestens einen Tag vor dem ersten Liefertag des ersten Auftrags sein muss, für den eine Produktion not-

wendig ist. Es wird versucht, die Zeit zwischen Optimierungsstart und erstem Lieferdatum so zu wählen, dass es zumindest theoretisch möglich ist, alle Aufträge ohne Verspätung zu produzieren. Dies wird über den Abgleich zwischen vorhandenen Kapazitäten (Maschinen und Werkzeuge) und benötigten Kapazitäten erreicht. Es werden zuerst die Werkzeuge und Maschinen unabhängig voneinander betrachtet. Des Weiteren wird eine sinnvolle Kombination beider untersucht. Die Startzeit zwischen Optimierungsstart und erstem Lieferdatum ergibt sich dann aus der größten Verspätung (kann sowohl positiv als auch negativ sein) aller Batches und wird immer auf ganze Tage aufgerundet.

5.2.4.1 Kapazitätsabgleich der Maschinen

Beim Kapazitätsabgleich der Maschinen wird weder die Werkzeugbelegung noch das Rüstintervall berücksichtigt. Es wird vom *Worst-Case-Szenario* ausgegangen, weswegen für jeden Batch ein Rüstvorgang angenommen wird und die Produktionszeit des langsameren Werkzeugs benutzt wird, falls ein Alternativwerkzeug vorhanden ist. Es werden die Batches der Reihenfolge nach, wie sie in der *BatchKanban*-Matrix gespeichert sind, den Maschinen zugeordnet. Dabei wird jedes Mal von allen möglichen Maschinen diejenige ausgewählt, die am frühesten frei wird. Sind mehrere Maschinen gleich früh frei, wird die erste dieser Maschinen ausgewählt. Dieser Vorgang wird so oft wiederholt, bis alle Batches eingeplant sind. Die Startzeit jedes Batches wird dann mit der spätesten erlaubten Startzeit verglichen, um zu überprüfen, ob für einen Auftrag eine Verspätung entsteht. Das Kapazitätsangebot der Maschinen reicht aus, um den Optimierungsstart bereits einen Tag vor dem ersten Liefertag des ersten Auftrags mit Produktion ansetzen zu können, ohne dass zwingend eine Verspätung entsteht.

5.2.4.2 Kapazitätsabgleich der Werkzeuge

Beim Kapazitätsabgleich der Werkzeuge wird weder die Maschinenbelegung noch das Rüstintervall berücksichtigt. Alle Batches werden zunächst auf ihrem Prio-Werkzeug eingeplant (mit der spätesten Startzeit aufsteigend), wobei wieder für jeden Batch ein Rüstvorgang angenommen wird. Entstehen für ein Werkzeug Verspätungen, wird versucht, den ersten Batch mit einer Verspätung auf ein Alternativwerkzeug zu verschieben. Hat ein Batch mit einer Verspätung kein Alternativwerkzeug, wird versucht, Batches, die vor diesem Batch eingeplant sind, auf ein Alternativwerkzeug zu verschieben. Dieser Vorgang wird für jedes Werkzeug so oft wiederholt, bis entweder keine Verspätungen mehr auftreten, oder es nicht mehr möglich ist, Verspätungen durch das Verschieben von Batches zu reduzieren. Das Kapazitätsangebot der Werkzeuge reicht für eine Batchrange von 10 Tagen nicht aus, sodass der Optimierungsstart 188.4 Stunden vor dem Lieferdatum des ersten Auftrags mit Produktion angesetzt werden muss, um eine zwingende Verspätung zu vermeiden.

5.2.4.3 Kapazitätsabgleich aus der Kombination von Maschinen und Werkzeugen

Es wird ähnlich vorgegangen wie bei dem Kapazitätsabgleich der Maschinen, nur werden jetzt die Werkzeugbelegung und das Rüstintervall berücksichtigt. Es wird für alle Batches ein Rüstvorgang angenommen. Die Batches werden der Reihenfolge nach, so, wie sie in der *BatchKanban*-Matrix gespeichert sind, den Maschinen zugeordnet. Dabei wird jedoch jede möglich Maschine durchgegangen und jene Maschine mit dem Werkzeug gewählt, auf der keine Werkzeugwartezeit entsteht. Gibt es mehrere mögliche Maschinen, auf denen keine Werkzeugwartezeit entsteht, wird aus diesen Maschinen die mit der frühesten Startzeit gewählt. Gibt es mehrere Maschinen ohne Werkzeugwartezeit, die gleich früh frei sind, wird die erste dieser Liste gewählt. Ist auf dieser Maschine für Prio- und Alternativwerkzeug die Produktion ohne Werkzeugwartezeit möglich, wird das Prio-Werkzeug ausgewählt. Das Resultat dieses Abgleichs zeigt ebenfalls, dass der Optimierungsstart 188.4 Stunden vor dem Lieferdatum des ersten Auftrags mit Produktion angesetzt werden muss, um eine zwingend Verspätung zu vermeiden.

5.2.4.4 Anpassung des Batchprozesses und der Prio-Werkzeuge

Bevor die Zeit zwischen Optimierungsstart und dem Lieferdatum des ersten Produktionsauftrags auf 188.4 Stunden gesetzt wird, wird der Batchprozess optimiert. Für eine Batchrange von 10 Tagen entstehen Batches mit einer Gesamtproduktionszeit von über 500 Stunden sowie einige Batches mit einer Produktionszeit von unter drei Stunden. Um den Batchprozess zu optimieren, wird die Gesamtproduktionszeit eines Batches auf 250 Stunden begrenzt, was dazu führt, dass Aufträge, deren Lieferdatum innerhalb der 10 Tage liegen, in einem Batch nich mehr aufgenommen werden, wenn dadurch die Gesamtproduktionszeit die 250 Stunden überschreitet. Des Weiteren wird die Batchrange für Batches, die eine Gesamtproduktionszeit von unter 20 Stunden haben auf 20 Tage erweitert. Dies führt dazu, dass Aufträge, deren Lieferdatum innerhalb der erweiterten Batchrange von 20 Tagen liegen, in einem Batch aufgenommen werden können, wenn dadurch die Gesamtproduktionszeit 20 Stunden nicht überschreitet. Durch diese Einschränkung und Erweiterung des Batchprozesses entstehen insgesamt 186 Batches statt der zuvor genannten 201 Batches.

Nach dem Anpassen des Batchprozesses, werden erneut die drei verschiedenen Kapazitätsabgleiche berechnet. Das Resultat zeigt in allen drei Fällen, dass der Optimierungsstart einen Tag vor dem Liefertag des ersten Auftrags mit Produktion angesetzt werden kann, ohne dass zwingend eine Verspätung entsteht. Der erste Auftrag, für den eine Produktion notwendig ist, ist ebenfalls der erste Auftrag in der Liste der Bestellungen des Excel-Files. Das Lieferdatum dieses Auftrags ist der 9.07.2018 um 6 Uhr, wodurch der Optimierungsstart am 8.07.2018 um 0 Uhr angesetzt wird. Die Zeit bis zum ersten Lieferdatum beträgt

somit 28.8 (22.8 + 6) Stunden.

Des Weiteren werden für die Batches, die laut dem Kapazitätsabgleich der Werkzeuge mit dem Alternativwerkzeug produziert werden sollten, um eine zwingende Verspätung zu vermeiden, die Alternativwerkzeuge als Prio-Werkzeuge definiert. In diesem Fall wird für insgesamt sechs der 186 Batches das Alternativwerkzeug als Prio-Werkzeug definiert.

5.2.5 Übersicht der Maschinenbelegung

Durch die Maschinenlisten, die die Maschinenauswahl einschränken, ist es schwierig abzuschätzen, ob theoretisch überhaupt eine gleichmäßige Verteilung der Batches auf allen Maschinen möglich wäre. In Abbildung 5.2 sind die minimalen (in grün) und maximalen (in rot) Belegzeiten der einzelnen Maschinen gut ersichtlich, wobei in diesem Fall keine Restriktionen berücksichtigt wurden (außer den Maschinenlisten der Werkzeuge). Die Belegzeit jedes Batches ergibt sich aus der Produktionszeit mit dem Prio-Werkzeug und dessen Rüstzeit. Wie in Abbildung 5.2 gut zu erkennen ist, ist es durch die Maschinenlisten aus dem Excel-File nicht möglich, Aufträge auf der Maschine Nr. 8 einzuplanen. Die minimale Belegung einer Maschine unterscheidet sich dann von Null, wenn es Batches gibt, die nur auf dieser einen Maschine eingeplant werden können. Die maximale Belegung jeder Maschine ergibt sich aus der Belegung aller Batches, die auf dieser Maschine eingeplant werden können.

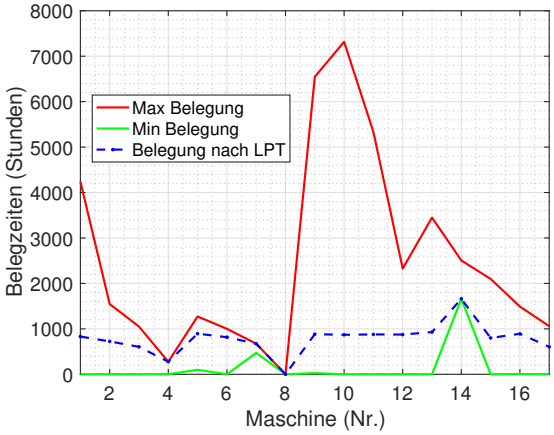

Abbildung 5.2: Übersicht der möglichen Maschinenbelegungen

Eine in der Fachliteratur gängige Heuristik, die für die gleichmäßige Verteilung von Aufträgen auf mehreren Maschinen eingesetzt wird, ist die *Longest Processing Time* (LPT) Heuristik. Diese Heuristik erzielt für das Problem der Diplomarbeit zwar keine optimale, aber durchaus zufriedenstellende Aufteilung ohne den Einsatz jeglicher Optimierungsansätze. Die Batches werden bei der LPT Heuristik nach den Produktionszeiten absteigend geordnet. Ausgehend vom Batch mit der höchsten Produktionszeit wird dann jeder Batch von den möglichen Maschinen auf der Maschine eingeplant, die am frühesten frei wird.

Sind mehrere Maschinen gleich früh frei, wird die Maschine ausgewählt, die die geringere maximal mögliche Belegung hat. Wie bereits erwähnt, werden in diesem Fall, außer der Maschinenliste der Werkzeuge, keine Restriktionen berücksichtigt. Das Ergebnis der LPT Heuristik ist in Abbildung 5.2 in blau zu sehen. Dementsprechend ist eine mehr oder weniger gleichmäßige Verteilung der Aufträge möglich, so gut es minimale und maximale Belegung zulassen. Die Frage ist, ob eine solch gleichmäßige Belegung auch unter der Berücksichtigung aller Restriktionen und Kosten einen zufriedenstellenden Produktionsplan ergibt.

Es sei hier generell erwähnt, dass das Kapazitätsangebot der Maschinen für die in dieser Arbeit betrachtete Auftragslage viel zu groß ist. Die gesamte Belegzeit (Produktionszeit + Rüstzeit) von allen Batches mit deren Prio-Werkzeug ergibt insgesamt 13.245 Stunden. Im Vergleich dazu umfasst das Kapazitätsangebot der 16 belegbaren Maschinen für den Zeitraum vom Optimierungsstart bis zum letzten Lieferdatum (233 Werktage, siehe Abschnitt 5.1.1.1) insgesamt $16 * 22,8 * 233 = 84.999,4$ Stunden. Dementsprechend sind bei einer gleichmäßigen Belegung der Maschinen hohe Lagerkosten zu erwarten, wenn Leerlaufzeiten, so gut es geht, vermieden werden sollen.

5.2.6 Größe des Lösungsraums

Anzahl der Batches	$n = 5$	$n = 10$	$n = 20$	$n = 50$	$n = 100$	$n = 186$
Größe des Lösungsraums	40	3200	9.22×10^7	6.01×10^{21}	1.76×10^{44}	6.28×10^{79}

Tabelle 5.1: Größe des Lösungsraums abhängig von der Anzahl der Batches.

Es wird davon ausgegangen, dass die Reihenfolge, in der die einzelnen Batches eingeplant werden, feststeht und während des Optimierungsprozesses nicht mehr verändert wird. Der Batchprozess übernimmt somit die Entscheidung der Reihenfolge, in der die einzelnen Aufträge eingeplant werden. Die Batches und deren Aufträge werden somit während der Optimierung immer in der Reihenfolge eingeplant, in der sie in der *BatchKanban*-Matrix gespeichert sind. Des Weiteren wird die Auswahl der Werkzeuge während dem Optimierungsprozess durch eine geeignete Heuristik umgesetzt und gilt somit auch nicht als Variable. Dementsprechend ergibt sich die Anzahl der Lösungen $N_{Lösungen}$ im Lösungsraum, aus den Kombinationen der verschiedenen Einplanungsprozesse der Batches auf deren möglichen Maschinen und wird durch folgende Gleichung dargestellt:

$$N_{Lösungen} = \prod_{i=1}^{n} M_{Bi}, \tag{5.1}$$

wobei n die Anzahl der Batches darstellt und M_{Bi} die Anzahl der möglichen Maschinen repräsentiert, auf denen der jeweilige Batch eingeplant werden kann. Die Größe des Lösungsraums ergibt sich somit aus der Produktreihe der Anzahl der möglichen Maschinen

aller 186 Batches. In diesem Fall ist der Lösungsraum in der Größenordnung von 10^{79} Lösungen (siehe auch Tabelle 5.1). Diese Zahl ist vergleichbar mit der Anzahl der Atome in unserem Universum, die sich in etwa in der Größenordnung von 10^{78} Atomen bewegt.[158]

Exakte Methoden, die jede einzelne Lösung im Lösungsraum betrachten, sind für diese unvorstellbare Menge an Lösungen unbrauchbar. Dementsprechend werden folglich Metaheuristiken an das Problem angepasst, um den Lösungsraum möglichst clever durchsuchen zu können.

5.3 Lösungsdarstellung und -decodierung

Wie bereits erwähnt, ist die Kodierung bzw. Darstellung einer Lösung ausschlaggebend für die Performance einer Metaheuristik. Die Darstellung einer Lösung sollte zu möglichst wenigen Rechenschritten der benutzten Operatoren führen. Für den GA wird zwischen Genotyp und Phänotyp unterschieden.[159] Der Genotyp stellt die Informationen dar, die in den Genen eines Individuums gespeichert sind. In der Problemstellung dieser Diplomarbeit sind das die Informationen, welcher Maschine jeder Batch zugeordnet wurde. Diese Informationen ergeben jedoch bei weitem noch keinen fertigen Produktionsplan. Es müssen die realen Start- und Endzeiten jedes Batches berechnet werden, unter der Berücksichtigung aller betrachteten Restriktionen. Der Genotyp muss dementsprechend decodiert werden, so wie auch die Gene eines Menschen nicht die fertige Person ausmachen. Die Gene jedes Menschen müssen erst durch den Organismus decodiert werden und ergeben erst dann das physikalische Erscheinungsbild, das auch Phänotyp genannt wird. Der fertige bzw. reale Produktionsplan, der durch die Berücksichtigung aller Informationen und Restriktionen entsteht, stellt somit den Phänotypen dar.

5.3.1 Lösungsdarstellung

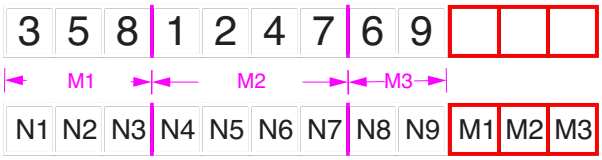

Abbildung 5.3: Lösungsdarstellung Für 9 Batches (**N**) und 3 Maschinen (**M**)

Für *parallel machine scheduling* Probleme wird in der Literatur oft folgende Darstellung einer Lösung benutzt: Es gibt sogenannte Auftrags- und Trenn-Gene. Jedes Auftrags-Gen repräsentiert einen Auftrag, wobei die Belegung der einzelnen Maschinen nacheinander

[158] Ganzenmüller, Andreas (2007): *Der Sinn des Lebens: was uns die Erwachsenen nicht beantworten können*, Gelnhausen, Wagner Verlag, S.116.
[159] vgl. Falkenauer, 1998, S.28f

in den Genen dokumentiert wird. Der Übergang zwischen zwei Maschinen wird durch ein Trenn-Gen mit beliebigem Zeichen dargestellt.[160] In dieser Arbeit wird eine leicht abgeänderte Version dieser Repräsentation benutzt, welche in Abbildung 5.3 zu sehen ist. Abbildung 5.3 zeigt ein Beispiel der Lösungsdarstellung von der Einplanung von insgesamt neun Batches auf drei Maschinen. Die Gene sind in Auftrags-Gene (N) ($N1$ bis $N9$ in rot) und Maschinen-Gene (M) ($M1$ bis $M3$ in blau) aufgeteilt. In den M-Genen wird die Information gespeichert, wie viele Aufträge (Batches) auf jeder Maschine eingeplant wurden. In den N-Genen wird die Belegung der einzelnen Maschinen durch die Reihenfolge der Auftragsnummern (in diesem Fall die Nummer und Reihenfolge der Batches) dargestellt. Die genauen Belegungen der einzelnen Maschinen in den N-Genen ergeben sich somit über die M-Gene. Im Beispiel in Abbildung 5.3 stellen z.B. die ersten drei Gene ($M1 = 3$) die Belegung der ersten Maschine dar. Die nächsten vier Gene ($M2 = 4$) stellen die Belegung der zweiten Maschine dar und die letzten zwei N-Gene ($M3 = 2$) stellen die Belegung der letzten Maschine dar. Eine Lösung der Problemstellung dieser Diplomarbeit wird somit durch insgesamt 203 Genen ($n + m$) dargestellt, nämlich 186 N-Gene ($n = 186$ Batches) und 17 M-Gene ($m = 17$ Maschinen).

5.3.2 Lösungsdecodierung

Für die Erstellung des realen Produktionsplans, der durch den Genotyp nur schematisch dargestellt wird, ist ein Decodierungsprozess notwendig. Dieser Decodierungsprozess wird durch einen sogenannten *Decoder* durchgeführt, der eine Heuristik darstellt. Der Decoder plant die Batches, so wie sie im Genotyp gespeichert sind, unter den realen Gegebenheiten auf den jeweiligen Maschinen ein. Obwohl durch den Genotyp die Belegungen der Maschinen festgelegt sind, ist die Reihenfolge, in der die Batches durch den Decoder eingeplant werden, ebenso entscheidend, insbesondere aufgrund der Berücksichtigung der Werkzeugbelegung. Überschneiden sich z.B. die geplanten Produktionszeiten von zwei Batches, die das gleiche Werkzeug benutzen und auf zwei unterschiedlichen Maschinen eingeplant sind, muss jener Batch, der als zweites eingeplant wird, auf die Fertigstellung des ersten eingeplanten Auftrags warten, bis das benötigte Werkzeug wieder frei ist. Für die Problemstellung dieser Arbeit, werden die Batches durch den Decoder immer in der Reihenfolge eingeplant, in der diese in der *BatchKanbab*-Matrix gespeichert sind, also aufsteigend nach der spätesten möglichen Startzeit. Die Batches sind folglich auch in dieser Reihenfolge nummeriert.

Der benutzte Decoder erstellt den realen Produktionsplan (Phänotyp) durch die Durchführung folgender Schritte für jeden Batch (siehe Algorithmus 5.1), wobei die Batches aufsteigend nach deren Nummer eingeplant werden. Der Algorithmus 5.1 stellt den vereinfachten Ablauf des Decoders dar, wobei für jeden Schritt in Klammer die berechneten Informationen stehen, die dann im weiteren Verlauf benötigt werden.

[160] vgl. Tavakkoli-Moghaddam u. a., 2009, S.3226

Algorithm 5.1 Pseudo-Ablauf des Decoders für die Einplanung eines Batches

1. Über die Position des Batches in den N-Genen und den M-Genen ergibt sich die Maschine, der der betrachtete Batch zugewiesen ist. (**Maschine**)
2. Der Batch wird vorerst auf dieser Maschine zu dem Zeitpunkt eingeplant, an dem diese frei wird.
3. Es wird überprüft, ob gerüstet werden muss (**Rüststatus**, **Rüstwartezeit**):
   a) Für den Fall, dass gerüstet werden muss, wird überprüft, ob der geplante Rüstvorgang innerhalb des Rüstintervalls (6-19 Uhr) stattfindet. Falls der geplante Rüstvorgang nicht vollständig während des Rüstintervalls stattfindet, entsteht eine Rüstwartezeit, in der die Maschine bis zum Umrüststart (6 Uhr) still steht.
   b) Für den Fall, dass nicht gerüstet werden muss, muss das Rüstintervall nicht überprüft werden und die geplante Startzeit bleibt unverändert.
4. Anschließend wird überprüft, ob das Prio-Werkzeug des betrachteten Batches für die gesamte Belegzeit (Rüstzeit und Produktionszeit) frei ist (**Werkzeugnummer, Werkzeugwartezeit**, **Rüstzeit, Produktionszeit**):
   a) Ist das Prio-Werkzeug für die geplante Belegzeit besetzt, wird überprüft, ob ein Alternativwerkzeug zur Verfügung steht:
      i. Ist kein Alternativwerkzeug vorhanden, entsteht eine Werkzeugwartezeit, in der die Maschine still steht, bis das Prio-Werkzeug wieder frei ist.
      ii. Ist ein Alternativwerkzeug vorhanden:
         A. das ebenfalls besetzt ist, wird das Werkzeug gewählt, welches früher wieder frei wird und es entsteht eine Werkzeugwartezeit, in der die Maschine still steht bis das ausgewählte Werkzeug wieder frei ist.
         B. das frei ist, wird dieses gewählt und es entsteht keine Werkzeugwartezeit.
   b) Ist das Prio-Werkzeug frei, wird dieses gewählt und es entsteht keine Werkzeugwartezeit.
5. Im Anschluss wird überprüft, ob für die geplante Rüstzeit ein Rüstmitarbeiter frei ist (**Rüstmitarbeiterwartezeit**).
   a) Ist kein Rüstmitarbeiter für diesen Zeitraum frei, entsteht eine Rüstmitarbeiterwartezeit, in der die Maschine still steht, bis einer der Rüstmitarbeiter wieder frei ist.
   b) Ist ein Rüstmitarbeiter frei, entsteht keine Wartezeit und es kann zur geplanten Zeit mit dem Rüstvorgang und folglich mit der Produktion begonnen werden.
6. Die entstehenden Werkzeug- und Rüstmitarbeiterbelegungen werden für den Einplanungsprozess der weiteren Batches gespeichert.
7. Des Weiteren wird der neue Zeitpunkt gespeichert, an dem die betrachtete Maschine wieder frei wird.
8. Abschließend werden alle Informationen gespeichert, die durch den Einplanungsprozess entstanden sind und für die Bewertung der entstehenden Kosten benötigt werden:
   a) Die **Maschine,** auf der der Batch in den Genen eingeplant ist.
   b) Den **Rüststatus** und die **Rüstwartezeit**.
   c) Die **Werkzeugnummer** des benutzten Werkzeuges und die damit verbundenen Details:
      i. **Rüstzeit**, **Produktionszeit** und **Werkzeugwartezeit**.
   d) Die **Rüstmitarbeiterwartezeit**.
   e) Die **Ist-Startzeit** des Batches, die sich durch den Einplanungsprozess ergeben hat.

Im Beispiel in Abbildung 5.3 würde der Decoder als erstes den Batch Nr. 1 auf der zweiten Maschine einplanen, anschließend den Batch Nr. 2 auf der gleichen Maschine, dann den Batch Nr. 3 auf der ersten Maschine, usw. bis alle Batches eingeplant sind.

Wie auch aus dem Ablauf des Decoders zu erkennen ist, sind folgende Restriktionen vorhanden, die die Ist-Startzeit jedes Batches beeinflussen:

- **Rüstintervall**: Wenn gerüstet werden muss, darf der Rüstvorgang nur innerhalb des Rüstintervalls (6 bis 19 Uhr) stattfinden. Alle Rüstvorgänge, die nicht vollständig innerhalb des Rüstintervalls stattfinden, verschieben sich auf den Umrüststart (6 Uhr), wodurch eine Rüstwartezeit entsteht.
- **Werkzeugbelegung**: Eine Produktion kann nur dann zur geplanten Zeit stattfinden, wenn eines der möglichen Werkzeuge für die gesamte Belegzeit (Rüstzeit + Produktionszeit) frei ist. Andernfalls entsteht eine Werkzeugwartezeit, bis ein mögliches Werkzeug wieder frei ist.
- **Rüstmitarbeiterbelegung**: Ein Werkzeug kann nur dann zur geplanten Zeit gerüstet werden, wenn ein Rüstmitarbeiter für den gesamten Rüstvorgang frei ist. Andernfalls entsteht eine Rüstmitarbeiterwartezeit, bis einer der Rüstmitarbeiter wieder frei ist.

Logischerweise müssen für eine Ist-Startzeit alle drei Restriktionen gleichzeitig erfüllt sein.

5.4 Zielfunktion

Nachdem durch den Decoder der realer Produktionsplan decodiert wurde, gilt es, die entstehenden Kosten des Produktionsplans zu bestimmen. Für die Problemstellung dieser Arbeit werden insgesamt fünf verschiedene Kostenarten berücksichtigt, nämlich die Lager-, Verspätungs-, Rüst-, Betriebs- und Wartezeitkosten. Über die Ist-Startzeiten der einzelnen Batches und die detaillierten Informationen der Zusammenstellung jedes Batches in der Batchtabelle lassen sich die Fertigstellungszeiten der einzelnen Aufträge in den Batches berechnen, womit dann die Lager- und Verspätungskosten der einzelnen Aufträge berechnet werden können. Um die Rechenzeit in Grenzen zu halten, wir der Decodierungs- und Kostenbewertungsprozess für die einzelnen Individuen der Population parallel berechnet.

5.4.1 Lagerkosten

Die Lagerkosten K_{Lager} jedes einzelnen Auftrags werden mit folgender Gleichung berechnet:

$$K_{Lager} = C_L \cdot [M_{IL} \cdot x_l + M_P \cdot (x_l - x_f)], \tag{5.2}$$

mit:

$C_L = \frac{LagerkostenProPaletteProTag}{StundenProTag \cdot StückProPalette} = \frac{LpPpT}{SpT \cdot SpP}$,

wobei:

$LpPpT = \frac{4}{20}$... die allgemeinen Lagerkosten einer Palette pro Tag sind.

$SpT = 22.8$... die Stunden darstellt, die jeden Tag für die Produktion zur Verfügung stehen.

SpP... die Menge in Stück darstellt, die vom jeweiligen Artikel auf eine Palette passen.

M_{IL}... die initiale Lagermenge in Stück darstellt, falls eine Bestellung teilweise über den Initialbestand abgedeckt ist.

M_P... die Produktionsmenge in Stück darstellt.

x_l... die Lieferzeit in Stunden darstellt (vom Optimierungsstart bis zum geplanten Lieferdatum des Auftrags um 6 Uhr).

x_f... die Fertigstellungszeit in Stunden darstellt (vom Optimierungsstart bis zur genauen Fertigstellung des Auftrages).

Da angenommen wurde, dass die Produktionsmengen von Aufträgen mit einer Verspätung keine Lagerkosten erzeugen, werden hier nur Aufträge betrachtet, die keine Verspätung haben. Es gilt somit $(x_l - x_f) \geq 0$. Jedoch müssen hier Aufträge, die sich verspäten und teilweise über den Initialbestand abgedeckt sind, ebenfalls berücksichtigt werden. In diesem Fall wird $x_f = x_l$ gesetzt, wodurch der Ausdruck $M_P \cdot (x_l - x_f) = 0$ ist.

Zusätzlich kann hier bereits der Bestandsverlauf des 3. Teils des Bestandes (siehe Abschnitt 5.2.1.2) ermittelt werden, wodurch die maximal auftretende Palettenanzahl im Planungshorizont berechnet werden kann. Für die betrachteten Artikel stehen laut Angaben des Unternehmens insgesamt 500 Palettenplätze zur Verfügung, wobei diese Restriktion nicht zwingend gilt, es stehen also im Notfall noch mehr Palettenplätze zur Verfügung. Da die Stammdaten im Excel-File leider fehlerhaft sind (siehe Abbildung 5.1, wo die Gesamtpaletten des 1. und 2. Teils des Bestandes bereits ca. 950 Paletten ergeben), wird die maximale Palettenanzahl dokumentiert, jedoch in den Kosten nicht berücksichtigt.

5.4.2 Verspätungskosten

Die Verspätungskosten K_{Versp} der Aufträge werden mit folgender Gleichung berechnet:

$$K_{Versp} = C_V \cdot L_M \cdot (x_f - x_l), \tag{5.3}$$

mit:

$C_V = \frac{VerspkostenProPaletteProTag}{StundenProTag \cdot StückProPalette} = \frac{VpPpT}{SpT \cdot SpP}$,

wobei:

$VpPpT = 100$... die allgemeinen Verspätungskosten einer Palette pro Tag sind.

$SpT = 22.8$... die Stunden darstellt, die jeden Tag für die Produktion zur Verfügung stehen.

SpP... die Menge in Stück darstellt, die vom jeweiligen Artikel auf eine Palette passen.

L_M... die Liefermenge in Stück darstellt.

x_l... die Lieferzeit in Stunden darstellt (vom Optimierungsstart bis zum geplanten Lieferdatum des Auftrags um 6 Uhr).

x_f... die Fertigstellungszeit in Stunden darstellt (vom Optimierungsstart bis zur genauen Fertigstellung des Auftrages).

Da in diesem Fall nur Aufträge mit einer tatsächlichen Verspätung betrachtet werden, gilt $(x_f - x_l) > 0$.

5.4.3 Rüstkosten

Die Rüstkosten $K_{Rüst}$ der Batches werden mit folgender Gleichung berechnet:

$$K_{Rüst} = C_R \cdot t_R, \tag{5.4}$$

wobei:

$C_R = 40$... den Stundensatz eines Rüstmitarbeiters darstellt.

t_R... die Rüstzeit in Stunden des jeweiligen Batches darstellt.

5.4.4 Betriebskosten

Die Betriebskosten K_{Betr}, die durch die Belegung eines Batches auf einer Maschine entstehen, werden mit folgender Gleichung berechnet:

$$K_{Betr} = C_B \cdot t_B - K_{unv}, \tag{5.5}$$

wobei:

C_B... den Stundensatz der Maschine darstellt, auf der der jeweilige Batch produziert wurde (siehe Absatz 5.1.1.3).

t_B... die Belegzeit in Stunden des jeweiligen Batches darstellt (Rüstzeit + Produktionszeit).

K_{unv}... die unvermeidbaren Betriebskosten des jeweiligen Batches darstellt.

Da die Betriebskosten bei Lösungen ohne große Verspätungen ca. 3/4 der Gesamtkosten einnehmen würden, werden für den Optimierungsvorgang nur jene Betriebskosten betrachtet, die auch wirklich optimierbar sind. Aus diesem Grund werden die unvermeidbaren Betriebskosten K_{unv} jedes einzelnen Batches abgezogen. K_{unv} repräsentiert für jeden Batch die Betriebskosten der Produktionszeit mit dem Prio-Werkzeug ohne Rüstvorgang auf der Maschine mit dem geringsten Stundensatz. Dies bedeutet, dass wenn ein Batch mit dem Prio-Werkzeug von den möglichen Maschinen auf derjenigen Maschine produziert wird, die den geringsten Stundensatz hat und auch kein Rüstvorgang stattfindet, keine Betriebskosten entstehen bzw. diese im Laufe dieser Arbeit nicht mehr weiter optimierbar sind. Demzufolge repräsentieren die Betriebskosten die optimierbaren Betriebskosten, wobei diese nicht ganz eliminiert werden können, da eine gewisse Anzahl an Rüstvorgängen notwendig ist.

5.4.5 Wartezeitkosten

Die Wartezeitkosten werden in die drei verschiedenen Arten der Wartezeiten aufgeteilt.

5.4.5.1 Rüstwartezeitkosten

Die Rüstwartezeitkosten $K_{RüstWZ}$ der Batches werden mit folgender Gleichung berechnet:

$$K_{RüstWZ} = C_{RWZ} \cdot t_{RWZ}, \tag{5.6}$$

wobei:

$C_{RWZ} = 1$... die allgemeinen Umrüststartwartezeitkosten pro Stunde darstellt.

t_{RWZ}... die Wartezeit auf den Umrüststart in Stunden des jeweiligen Batches darstellt.

5.4.5.2 Werkzeugwartezeitkosten

Die Werkzeugwartezeitkosten K_{WerkWZ} der Batches werd mit folgender Gleichung berechnet:

$$K_{WerkWZ} = C_{WWZ} \cdot t_{WWZ}, \tag{5.7}$$

wobei:

$C_{WWZ} = 1$... die allgemeinen Werkzeugwartezeitkosten pro Stunde darstellt.

t_{WWZ}... die Wartezeit auf ein Werkzeug in Stunden des jeweiligen Batches darstellt.

5.4.5.3 Rüstmitarbeiterwartezeitkosten

Die Rüstmitarbeiterwartezeitkosten $K_{RüstmaWZ}$ der Batches werden mit folgender Gleichung berechnet:

$$K_{RüstmaWZ} = C_{RmaWZ} \cdot t_{RmaWZ,} \tag{5.8}$$

wobei:

$C_{RmaWZ} = 1$... die allgemeinen Rüstmitarbeiterwartezeitkosten pro Stunde darstellt.

t_{RmaWZ}... die Wartezeit auf einen Rüstmitarbeiter in Stunden des jeweiligen Batches darstellt.

5.4.6 Gesamtkosten

Werden mehrere Kriterien in der Zielfunktion berücksichtigt, gibt es eine Vielzahl an Methoden, wie diese ausgewertet werden können.[161] In dieser Arbeit wird die einfache Summe der einzelnen Kosten gebildet, die möglich ist, weil hier vergleichbare Kostenarten miteinander verglichen werden. Es muss einem bewusst sein, dass es bei der Berücksichtigung von mehreren Kriterien oft kein eindeutiges globales Optimum gibt. In diesem Fall können eine Menge an verschiedenen Kombination der einzelnen Kosten zu den gleichen Gesamtkosten führen und somit auch zu mehreren globalen Optima.

Die Gesamtkosten eines Produktionsplans ergeben sich somit aus der Summer der einzelnen Kosten:

$$K_{Ges} = K_{Lager} + K_{Versp} + K_{Rüst} + K_{Betr} + K_{RüstWZ} + K_{WerkWZ} + K_{RüstmaWZ}. \tag{5.9}$$

5.5 Matlab Performance

Bevor der GA der *optimization toolbox* von Matlab an die Problemstellung angepasst wird, wird kurz auf die allgemeine Performance von Matlab eingegangen.

5.5.1 Vergleich Matlab/C++

Matlab ist vor allem sehr gut für die Erstellung eines Prototypen eines Programmes geeignet. Die Entwicklungszeit für ein Programm ist dementsprechend in Matlab um einiges kürzer als z.B. mit C++. Zusätzlich hat Matlab ein sehr umfangreiches Angebot an Toolboxen

[161] u.a. sind Methoden wie die Ermittlung der *Pareto Front* oder *Lexicographical goal programming* weit verbreitet.

und *build-in* Funktionen, die zur Reduzierung der Entwicklungszeit ebenfalls beitragen. Ein sehr großer Nachteil von Matlab ist jedoch, dass Matlab um einiges langsamer ist als C++. Der Geschwindigkeitsunterschied ist sehr stark vom eigentlichen Programm abhängig, wobei Matlab in einer Größenordnung von 10 bis 100 Mal langsamer sein kann als C++.[162] Für diese Diplomarbeit bedeutet das, dass die Verwendung von Matlab dazu führt, dass die Entwicklungszeit der Metaheuristik verringert wird, jedoch dann die Performance (Laufzeit) beeinträchtigt ist. Dies ermöglicht jedoch mehr Zeit in die Anpassung der Metaheuristik an die Problemstellung dieser Arbeit zu investieren. Dadurch, dass in dieser Arbeit die Entwicklung und Anpassung einer Metaheuristik im Vordergrund stehen, ist Matlab die geeignetere Wahl. Für die Anwendung in der Praxis sollte jedoch in Erwägung gezogen werden, das endgültige Programm z.B. in C++ zu implementieren.

5.5.2 Globale Variablen in Matlab

Globale Variablen sind Variablen, die in allen Funktionen aufgerufen werden können und somit die Entwicklung und Testläufe eines Programms um ein Vielfaches erleichtern können. Dies gilt vor allem für Parameter, deren Wert während der Entwicklung immer wieder angepasst werden muss. Jedoch leidet unter anderem die Laufzeit eines Programmes in Matlab darunter, weshalb oftmals von der Benutzung globaler Variablen abgeraten wird.[163] Um zu untersuchen, ob diese Aussage für diese Arbeit relevant ist, wird eine Funktion erstellt, in der zwei Variablen aufgerufen bzw. erstellt werden. Diese Funktion wird wiederum in einer Schleife 10^7 Mal aufgerufen. Für den ersten Durchlauf, erstellt die Funktion zwei lokale Variablen und für den zweiten Durchlauf ruft die Funktion zwei globale Variablen auf:

1. **Mit lokalen Variablen**: benötigen die 10^7 Aufrufe der Funktion ca. 0.52 Sekunden. Das ergibt im Durchschnitt für die Definition einer lokalen Variable $2.61 \cdot 10^{-8}$ Sekunden.
2. **Mit globalen Variablen**: benötigen die 10^7 Aufrufe der Funktion ca. 39.04 Sekunden. Das ergibt im Durchschnitt für den Aufruf einer globalen Variable $1.95 \cdot 10^{-6}$ Sekunden.

Dementsprechend ist in Matlab der Aufruf einer globalen Variable in einer Funktion mehr als 70 Mal langsamer als die Definition einer lokalen Variable. Es sollte also darauf geachtet werden, dass in Funktionen, die sehr oft aufgerufen werden, keine globalen Variablen benutzt werden. Zum Beispiel werden Funktionen, die für jeden einzelnen Batch eines Individuums aufgerufen werden, für eine Population von 100 Individuen über 100 Generationen insgesamt $186 \cdot 100 \cdot 100 = 1.86 \cdot 10^6$ Mal aufgerufen. Für solche Funktionen

[162] https://stackoverflow.com/questions/20513071/performance-tradeoff-when-is-matlab-better-slower-than-c-c (besucht am 18.03.19)

[163] siehe z.B.: https://de.mathworks.com/matlabcentral/answers/99537-which-type-of-function-call-provides-better-performance-in-matlab (besucht am 18.03.19)

sollte der Einsatz von globalen Variablen vermieden werden, da sich sonst die Laufzeit der Metaheuristik erheblich verlängern kann. Wohingegen der Einsatz von globalen Variablen in Funktionen, die nur für jedes Individuum aufgerufen werden (in diesem Fall nur 10^4 Aufrufe), keinen so kritischen Unterschied erzielt.

5.6 Anpassung des GA an die Problemstellung

Die Lösungsdarstellung, -decodierung und die Zielfunktion werden für den GA so übernommen, wie sie in den Abschnitten 5.3 und 5.4 beschrieben wurden. Für 186 Batches und 17 Maschinen besteht ein Individuum somit aus insgesamt 203 Genen.

5.6.1 Initiale Population

Wie bereits erwähnt, sollte die initiale Population eine möglichst hohe Diversität aufweisen und bereits möglichst gute Individuen beinhalten. Um dies zu erreichen, bietet sich unter anderem die Konstruktionsphase der GRASP Metaheuristik sehr gut an (siehe Abschnitt 4.8.6.2). Für diese Arbeit wird eine Kombination aus insgesamt acht verschiedenen initialen Populationen in Betracht gezogen, wobei zwei davon zufällig erstellte Populationen sind und sechs davon durch eine *greedy* Heuristik erstellt werden. Für weitere Diskussionen werden die verschiedenen initialen Populationen der Reihe nach von 1 bis 8 durchnummeriert. Im Weiteren wird nur auf die Zusammenstellung der N-Gene eingegangen. Die M-Gene jedes Individuums ergeben sich durch den durchgeführten Einplanungsprozess der Batches.

5.6.1.1 Zufällige Populationen

Eine zufällig erstellte Population erhöht meist die Diversität einer Population, da in diesem Fall keine möglichen Kombinationen ausgeschlossen werden und die Individuen mehr oder weniger gleichmäßig im Lösungsraum verteilt sind. Jedoch sind zufällig erstellte Individuen oft mit einem höheren Fitnesswert verbunden.

Zufällige Einplanung der einzelnen Batches (1)

Für diese initiale Population werden für die Einplanung der Batches auf den Maschinen keine Restriktionen berücksichtigt. Die Batches werden nach ihrer Nummer aufsteigend den Maschinen zugeordnet. Alle Maschinen einer Maschinenliste haben in diesem Fall die gleiche Wahrscheinlichkeit. Für die Einplanung jedes Batches wird dementsprechend immer zufällig eine der möglichen Maschinen ausgewählt, wobei eben jede Maschine die gleiche Wahrscheinlichkeit hat, ausgewählt zu werden. Dieser Vorgang wird so oft wiederholt, bis alle Batches einer Maschine zugeteilt wurden. Die Belegung der einzelnen Maschinen wird dann in den N-Genen abgespeichert.

Zufällige Einplanung ganzer Werkzeuge (2)

Für diese initiale Population werden ebenfalls keine Restriktionen berücksichtig. Der Unterschied zur vorigen initialen Population besteht darin, dass den Maschinen nicht einzelne Batches zufällig zugeordnet werden, sondern ganze Werkzeuge. Die Belegung eines Werkzeuges ergibt sich aus den Batches, für die das Werkzeug als Prio-Werkzeug definiert ist. Dementsprechend werden alle Batches, die ein gemeinsames Prio-Werkzeug aufweisen, immer gemeinsam auf einer Maschine eingeplant, wobei jede mögliche Maschine wieder die gleiche Wahrscheinlichkeit hat, ausgewählt zu werden. Dieser Vorgang wird so lange wiederholt, bis alle Prio-Werkzeuge und somit auch alle Batches eingeplant wurden. Letztendlich werden die Batches für jede Maschine wieder nach deren Nummer in aufsteigender Reihenfolge geordnet und in den N-Genen abgespeichert.

5.6.1.2 Gleichmäßige Population

Die bereits vorgestellte LPT Heuristik (siehe Abschnitt 5.2.5) erstellt eine zufriedenstellende gleichmäßige Verteilung der Batches auf den Maschinen, wobei diese Heuristik vollständig deterministisch ist und somit nur eine mögliche Lösung erstellen kann. Dementsprechend wird dieser Heuristik eine stochastische Komponente hinzugefügt, um eine Vielzahl an verschiedenen Lösungen erstellen zu können, die eine mehr oder weniger gleichmäßige Verteilung der Batches auf den Maschinen als Ziel haben.

Gleichmäßige Aufteilung der einzelnen Batches (3)

Für die gleichmäßige initiale Population werden außer den Maschinenlisten keine Restriktionen berücksichtigt. Die Belegzeit jedes Batches ergibt sich aus der Rüst- und Produktionszeit mit dem jeweiligen Prio-Werkzeug. Für jedes Individuum werden die Batches zufällig durchgemischt. Die Batches werden dann in ihrer neuen Reihenfolge den möglichen Maschinen zugeordnet, wobei, wie bei der LPT Heuristik, immer die am frühesten frei werdende Maschine ausgewählt wird. Sind mehrere Maschinen gleichfrüh frei wird zufällig (mit gleicher Wahrscheinlichkeit) eine dieser Maschinen ausgewählt. Dieser Vorgang wird so oft wiederholt, bis alle Batches einer Maschine zugeordnet wurden. Für jede Maschine wird dann die Reihenfolge der Batches wieder nach den ursprünglichen Batchnummern sortiert. Die fertige Belegung der einzelnen Maschinen wird dann in den N-Genen abgespeichert.

5.6.1.3 Population mit möglichst geringen Einzelkosten

Die restlichen fünf initialen Populationen werden mit einer *greedy* Heuristik erstellt, die sowohl Werkzeugbelegung als auch Rüstintervall berücksichtigt. Es werden bei diesen initialen Populationen die Rüstmitarbeiter nicht berücksichtig. Jede dieser initialen Populationen betrachtet eine der folgenden Kosten: Lager-, Verspätungs-, Rüst-, Betriebs- oder

Werkzeugwartezeitkosten. Das Ziel jeder initialen Population ist es, die Batches so einzuplanen, dass die jeweiligen betrachteten Kosten möglichst gering ausfallen. Der allgemeine Einplanungsprozess ist für die restlichen Populationen gleich und läuft folgendermaßen ab:

Die Batches werden aufsteigend nach ihrer Nummer den Maschinen zugeordnet. Jeder Batch wird auf allen möglichen Maschinen mit allen möglichen Werkzeugen eingeplant. Die Anpassung der Startzeit an die Restriktionen läuft gleich wie im Decoder ab, allerdings werden keine Rüstmitarbeiter berücksichtigt. Die Ergebnisse der Einplanung (Ist-Startzeit, Rüstzeit, Produktionszeit, Werkzeugwartezeit, Rüstwartezeit) eines Batches auf jeder der möglichen Maschinen mit jedem mögliche Werkzeug werden zwischengespeichert. Die Werkzeugauswahl auf jeder Maschine muss gleichermaßen wie im Decoder entschieden werden. In diesem Fall wird immer für jede Maschine die Kombination mit dem Werkzeug entfernt, die die größere Werkzeugwartezeit besitzt. Sind Prio- und Alternativwerkzeug gleichfrüh frei, wird die Kombination mit dem Alternativwerkzeug entfernt. Diese vorgegebene Auswahl des Werkzeuges durch den Decoder macht es jedoch für die einzelnen initialen Populationen nicht möglich, immer die idealen Kombinationen für die betrachteten Kosten zur Auswahl zu haben. Übrig bleibt somit nur mehr die Auswahl der Maschinen. Je nachdem welche Kosten betrachtet werden, wird dann folgendermaßen vorgegangen:

Lagerkosten (4)

Werden die Lagerkosten betrachtet, werden für einen Batch aus allen übrig bleibenden Kombinationen jene entfernt, die eine Verspätung haben. Aus den restlichen Möglichkeiten werden die, mit der geringsten Differenz zwischen spätester Startzeit und Ist-Startzeit ausgewählt. Ist keine Kombination ohne Verspätung möglich, wird die ausgewählt, die die geringste Verspätung hat. Bleibt nur mehr eine mögliche Maschine übrig, steht die Maschine fest. Bleiben jedoch mehrere mögliche Maschinen übrig, wird zufällig (mit gleicher Wahrscheinlichkeit) eine der restlichen Maschinen ausgewählt. Die ausgewählte Kombination aus Maschine und Werkzeug wird dann mit all den entstehenden Informationen übernommen. Dieser Vorgang wird so lange wiederholt, bis alle Batches einer Maschine zugeordnet wurden. Die Reihenfolge der Batches auf den einzelnen Maschinen wird dann in den N-Genen abgespeichert.

Verspätungskosten (5)

Werden die Verspätungskosten betrachtet, werden für einen Batch aus all den übrig bleibenden Kombinationen die entfernt, die eine Verspätung haben. Aus den bleibenden Möglichkeiten wird dann zufällig (mit gleicher Wahrscheinlichkeit) eine der Maschinen ausgewählt. Ist keine der übrig bleibenden Kombinationen ohne Verspätung möglich, wird die ausgewählt, die die geringste Verspätung hat. Bleibt nur mehr eine Maschine übrig, steht die Maschine fest, ansonsten wird zufällig eine der Maschinen ausgewählt.

Dieser Vorgang wird so oft wiederholt, bis alle Batches einer Maschine zugeordnet wurden. Schließlich wird das Resultat wieder in den N-Genen gespeichert.

Rüstkosten (6)

Für die initiale Population, die die Rüstkosten betrachtet, werden für einen Batch aus den übrig bleibenden Kombinationen bevorzugt die ausgewählt, die keinen Rüstvorgang haben. Gibt es nur eine Maschine ohne Rüstvorgang, steht die Maschine fest, ansonsten wird zufällig (mit gleicher Wahrscheinlichkeit) eine ausgewählt. Gibt es keine Maschine ohne Rüstvorgang, wird die Maschine ausgewählt, die die geringste Wartezeit auf Rüststart hat. Stehen in diesem Fall wieder mehrere Maschinen zur Auswahl, wird wieder zufällig eine davon ausgewählt. Dieser Vorgang wird wiederum so oft wiederholt, bis alle Batches einer Maschine zugeordnet wurden. Abschließend wird das Resultat wieder in den N-Genen gespeichert.

Betriebskosten (7)

Werden die Betriebskosten betrachtet, werden für einen Batch aus all den übrig bleibenden Kombinationen bevorzugt die ausgewählt, die die geringsten Betriebskosten haben. Betriebskosten sind vom eingesetzten Werkzeug, ob ein Rüstvorgang stattfindet und der Maschine abhängig. Gibt es mehrere Kombinationen mit gleichgeringen Betriebskosten, wird zufällig (mit gleicher Wahrscheinlichkeit) eine ausgewählt. Dieser Vorgang wird so oft wiederholt, bis alle Batches eingeplant sind und schließlich das Ergebnis in den N-Genen gespeichert wird.

Werkzeugwartezeitkosten (8)

Für die initiale Population, die die Werkzeugwartezeiten betrachtet, werden für einen Batch bevorzugt aus den übrigen Kombinationen die ausgewählt, die keine Werkzeugwartezeit haben. Für jedes Werkzeug gibt es immer mindestens eine Maschine, auf der es zu keiner Werkzeugwartezeit kommt, nämlich der Maschine, auf welcher das jeweilige Werkzeug das letzte Mal benutzt wurde. Gibt es wiederum mehrere mögliche Maschinen ohne Werkzeugwartezeiten, wird zufällig (mit gleicher Wahrscheinlichkeit) eine ausgewählt. Dieser Vorgang wird wieder so oft wiederholt, bis alle Batches einer Maschine zugeordnet wurden. Die Reihenfolge der Batches auf den einzelnen Maschinen wird wiederum in den N-Genen gespeichert.

5.6.2 Bewertung und Diskussion der verschiedenen initialen Populationen

In diesem Abschnitt werden die Kosten der acht zuvor präsentierten initialen Populationen verglichen, mit dem Ziel zu untersuchen, ob diese wirklich die Resultate liefern, die erwartet werden. Des Weiteren werden auch die Distanzen zwischen den erstellten Individuen untersucht (siehe genaue Definition von Distanz in Abschnitt 5.6.2.2).

5.6.2.1 Bewertung der Kosten

Für eine detaillierte Diskussion der Kostenbewertung der verschiedenen initialen Population siehe Anhang 8.2. Die Erwartungen der verschiedenen initialen Populationen mit möglichst geringen Einzelkosten wurden weitgehend bestätigt. Dadurch kann zusammengefasst werden, dass jede initiale Population ihre Stärken und Schwächen hat, wodurch auch jede ihren Teil zur genetischen Vielfalt beiträgt.

5.6.2.2 Bewertung der Distanzen

Um sicherzustellen, dass die erstellten Individuen aus den initialen Populationen auch wirklich eine hohe Diversität aufweisen, wird die Distanz zwischen den Individuen berechnet. Für eine binäre Darstellung einer Lösung spricht man auch von der *Hamming-Distanz*, die angibt, an wie vielen Stellen sich zwei binäre Zahlen unterscheiden.[164] Dieses Distanzmaß muss jedoch für die gewählte Lösungsdarstellung in dieser Arbeit angepasst werden.

Für die ausgewählte Lösungsdarstellung sollte das Distanzmaß zwischen zwei Individuen angeben, wie viele Batches auf unterschiedlichen Maschinen eingeplant wurden. Dafür müssen alle Maschinen der betrachteten Individuen einzeln verglichen und die Schnittmengen der Batches berechnet werden. Die Distanz D_{xy} zwischen zwei Individuen x und y ergibt sich somit aus der gesamten Anzahl der Batches n minus der Summe aller Schnittmengen S_i, die die Anzahl der gleich eingeplanten Batches aller m Maschinen darstellt:

$$D_{xy} = n - \sum_{i=1}^{m} S_i \tag{5.10}$$

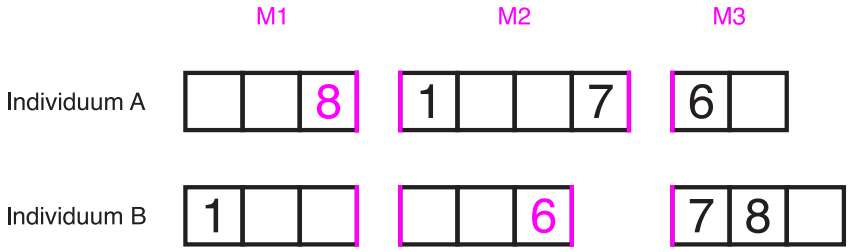

Abbildung 5.4: Beispiel von zwei Individuen für die Distanzberechnung

Für das Beispiel in Abbildung 5.4 sind die Batches von Individuum A und B in rot gekennzeichnet, wenn diese auf der gleichen Maschine eingeplant sind. Für die insgesamt neun Batches ergibt sich zwischen Individuum A und B eine Distanz von :

$D_{AB} = 9 - (2 + 2 + 1) = 4.$

Für die Bewertung der Distanzen werden von jeder initialen Population 1000 Individuen erstellt und die Distanz von jedem Individuum zu allen anderen Individuen der betrachteten

[164] vgl. Zäpfel und Braune, 2005, S.126

initialen Population berechnet. Dies wird für jede einzelne initiale Population insgesamt 10 Mal wiederholt, wobei jedes Mal die minimale, maximale und durchschnittliche Distanz sowie die Standardabweichung berechnet werden. Letztendlich wird dann der Durchschnitt von allen vier Werten für jede initiale Population berechnet. Die gerundeten Ergebnisse sind in Tabelle 5.2 sichtbar, wobei die Nummern der initialen Populationen mit der Reihenfolge, wie diese in Abschnitt 5.6.1 vorgestellt wurden, übereinstimmen. Da für die 186 Batches nur 156 auf mehreren Maschinen produziert werden können, ist die größte maximale Distanz zwischen zwei Individuen 156.

	Initiale Population (Nr.)							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Minimum	78	32	47	1	77	11	53	70
Maximum	130	155	105	134	131	131	126	132
Durchschnitt	105	105	76	83	105	94	93	103
Std. Abweichung	6	15	6	19	6	11	8	7

Tabelle 5.2: Distanzen zwischen 1000 Individuen von jeder Population

Die initiale Population mit möglichst geringen Lagerkosten (Nr. 4) besitzt die geringste Mindestdistanz von 1, was bedeutet, dass hier leicht Individuen entstehen können, die identisch sind, da der Wert aufgerundet wurde. Dies lässt sich dadurch erklären, dass die Auswahlmöglichkeiten der Maschinen sehr eingeschränkt sind, da immer die Maschine gewählt wird, auf der die geringsten Lagerkosten entstehen bzw. die mit der geringsten Verspätung, falls eine Verspätung unvermeidbar ist. Diese Auswahlmöglichkeiten macht diese initiale Population sehr deterministisch, wodurch die Anzahl der erstellbaren Individuen sehr beschränkt ist. Ganz im Gegenteil sind z.B. für die initiale Population mit möglichst geringen Verspätungen (Nr. 5) die Auswahlmöglichkeiten der Maschinen um einiges mehr stochastisch, da oft für einen Batch mehrere mögliche Maschinen ohne Verspätung zur Auswahl stehen. Dementsprechend ist hier die minimale Distanz mit 77 um einiges höher. Des Weiteren bestätigt sich auch die Vermutung aus der Kostenbewertung, dass die Maschinenauswahl der initialen Population Nr. 5 in den meisten Fällen identisch mit der zufälligen Einplanung der Batches von der initialen Population Nr. 1 ist, da hier alle vier Werte der Distanz fast identisch sind. Die geringste maximale und durchschnittliche Distanz entsteht in der initialen Population mit einer möglichst gleichmäßigen Verteilung (Nr. 3), was dadurch zu erklären ist, dass Maschinen weder leer noch überfüllt sein dürfen, wodurch die maximale Distanz zwischen Individuen eingeschränkter ist, was sich ebenfalls auf die durchschnittliche Distanz auswirkt. Nichtsdestotrotz zeigt die durchschnittliche Distanz und die Standardabweichung aller initialen Populationen, dass die Individuen ausreichend gut im Lösungsraum verteilt sind.

5.6.3 Anpassung der Operatoren

Ein wichtiger Aspekt bei der Anpassung der Operatoren einer Metaheuristik ist das effektive Nutzen der Struktur des betrachteten Problems. Für das Problem dieser Arbeit bietet sich sehr gut die Logik des sogenannten *Grouping Genetic Algorithm* (GGA) an. Der GGA wurde ursprünglich von Emanuel Falkenauer (1992) entwickelt und ist eine Abwandlung des klassischen GA, um eben effektiv die Struktur von Problemen zu erfassen, in denen es darum geht, Objekte in Gruppen einzuteilen. In der Praxis gibt es eine Vielzahl von Problemen, die sich als Gruppierungsproblem zusammenfassen lassen, unter anderem klassische Probleme wie: assembly line balancing, job shop scheduling, container loading, pickup and delivery, team formation und viele andere Probleme.[165] Das Problem dieser Diplomarbeit lässt sich ebenfalls als Gruppierungsproblem zusammenfassen, nämlich der Gruppierung von Batches auf Maschinen.

5.6.3.1 Crossover

Im GGA arbeitet der Crossover mit den Gruppen und nicht mit den einzelnen Genen. Es werden dementsprechend nur ganze Gruppen von einem Elternteil zum anderen übergeben. Der generelle Ablauf des Crossovers des GGA kann in folgenden Schritten zusammengefasst werden:[166]

1. Es wird eine zufällige Anzahl an Maschinen (mindestens eine bis $m-1$) ausgewählt, die vom Elternteil A dem Elternteil B übergeben werden.
2. Die ausgewählten Maschinen werden dann im Elternteil B gelöscht und durch die Maschinen des Elternteils A ersetzt.
3. Dadurch können doppelte Einträge von Batches entstehen, die auf mehreren Maschinen eingeplant sind. Diese doppelten Einträge werden von den nicht eingefügten Maschinen gelöscht, dementsprechend von den Maschinen des Elternteils B.
4. Des Weiteren kann es dazu kommen, dass Batches verloren gehen, und somit in den Genen nicht mehr vorkommen. Diese fehlenden Batches können durch eine Heuristik auf einer der möglichen Maschinen neu eingeplant werden.

In Abbildung 5.5 ist der Ablauf des Crossovers des GGA dargestellt. In diesem Beispiel wird die Maschine Nr. 2 vom Elternteil A dem Elternteil B übergeben. Nach dieser Kreuzung sind Batch Nr. 1 und 7 doppelt und Batch Nr. 6 fehlt. Dementsprechend werden Batch Nr. 1 und 7 auf den Maschinen vom Elternteil B gelöscht und Batch Nr. 6 wird mit einer Heuristik einer Maschine zugeordnet, in diesem Fall der Maschine Nr. 1. Diese Heuristik kann den Batch zufällig einer möglichen Maschine zuordnen oder gezielt problemspezifische Informationen nutzen.

[165] vgl. Mutingi und Mbohwa, 2017, S.4ff
[166] vgl. Falkenauer, 1998, S.100f

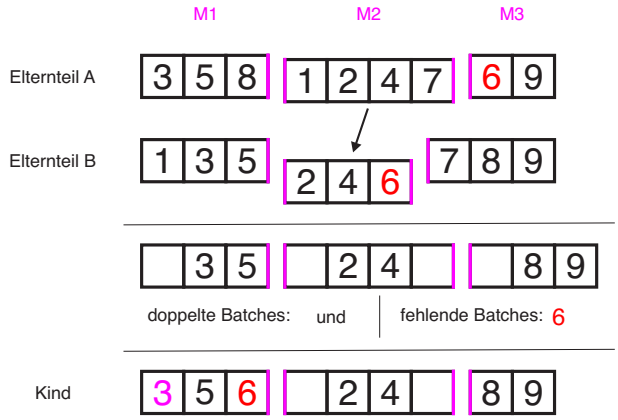

Abbildung 5.5: GGA Crossover

Dieser Ablauf des Crossovers wird vorerst für die Experimentserien so übernommen, wobei fehlende Batches zufällig auf einer möglichen Maschine eingeplant werden.

5.6.3.2 Mutation

So wie der Crossover sollte die Mutation ebenfalls die Struktur des Problems nutzen und mit den Gruppen arbeiten. Je nach Problemstellung kommen für die Mutation laut Falkenauer drei verschiedene Variationen in Frage:[167]

1. das Erstellen neuer Gruppen,
2. das Löschen einer bereits existierenden Gruppe,
3. oder das Verschieben einzelner Elementen zwischen den Gruppen.

Für das Problem dieser Arbeit kommt vor allem das Verschieben einzelner Batches zwischen den Maschinen in Frage. Für jede Mutation wird nur ein Batch verschoben, wobei problemspezifische Informationen aus der Bewertung des zu mutierenden Individuums benutzt werden. Diese Informationen beinhalten unter anderem die genaue Start- und Endzeit, das benutzte Werkzeug, den Rüststatus, sowie einzelne Kosten, wie Lager-, Werkzeugwartezeit- oder Rüstkosten jedes einzelnen Batches.

Es zeigt sich, dass die Strategie, sich immer auf die höchsten Kosten zu konzentrieren, nicht zu den besten Ergebnissen führt. Das Ineinandergreifen der einzelnen Ziele bzw. Kosten macht ebenfalls eine gezielte Auswahl von mittelmäßigen oder auch gut erreichten Ziele notwendig.[168] Dazu kommt, dass die Reduktion der Gesamtkosten oft nicht nur durch das bewusste Reduzieren eines Kriteriums erzielt wird, sondern durch einen zufälligen Ausgleich zwischen den einzelnen Kriterien.[169] Dementsprechend ist es nicht immer zielführend, in einer Mutation den Batch mit den höchsten Kosten zu betrachten. Der Mutationsoperator sollte also sowohl deterministische als auch stochastische Eigenschaften besitzen.

[167] vgl. Falkenauer, 1998, S.102
[168] vgl. Fritzsche, 2009, S.216f
[169] vgl. Fritzsche, 2009, S.103

Demnach wird der Ablauf der Mutation folgendermaßen vorgenommen, wobei alle einzelnen Kosten außer der Wartezeit auf Rüststart betrachtet werden:

1. Es wird eine zufällige Reihenfolge der einzelnen Kosten definiert, die dann der Reihe nach betrachtet werden.
2. Für jede Kostenart werden nur die Batches betrachtet, für die die Kosten auch wirklich auftreten und die auch verschiebbar sind, also auf mehr als nur einer Maschine eingeplant werden können.
3. Die übrig bleibenden Batches werden in einer zufälligen Reihenfolge abgearbeitet.
4. Je nach Kostenart sind bestimmte Kriterien zu erfüllen, damit die Mutation eines Batches durchgeführt wird. Wird ein passender Batch gefunden, beginnt der Vorgang wieder beim 1. Punkt mit dem nächsten Individuum.
5. Treten die betrachteten Kosten nicht auf bzw. erfüllen keine der übrig bleibenden Batches die Kriterien, werden die nächsten Kosten betrachtet.

Im Mutationsoperator werden die Werkzeugbelegung, das Rüstintervall und die Rüstmitarbeiter nicht berücksichtigt. Damit die Mutation eines Batches stattfindet, werden im 4. Punkt für die einzelnen Kosten folgende Kriterien beachtet:

Verspätungskosten

Für jeden Batch mit einer Verspätung wird die Plan-Startzeit auf den anderen möglichen Maschinen berechnet. Die Plan-Startzeit auf jeder Maschine ergibt sich aus der Endzeit des Batches mit der nächstkleineren Nummer, da die aufsteigende Reihenfolge der Batchnummern weiterhin berücksichtigt werden soll. Die Maschinen werden in der Reihenfolge betrachtet, in der sie in der Maschinenliste des Batches abgespeichert sind. Ergibt sich auf einer Maschine eine frühere Plan-Startzeit als die aktuelle Ist-Startzeit, wodurch die Verspätung reduziert werden kann, wird die Maschine ausgewählt und die restlichen Maschinen werden nicht mehr betrachtet. Ist auf allen möglichen Maschinen die Plan-Startzeit später als auf der aktuellen Maschine, wird der nächste verschiebbare Batch mit Verspätungskosten betrachtet.

Lagerkosten

Für jeden Batch mit Lagerkosten werden analog, wie für die Verspätungskosten, die Plan-Startzeiten auf den anderen möglichen Maschinen berechnet. Wobei für die Lagerkosten eine Maschine erst dann ausgewählt wird, wenn die Plan-Startzeit später als die aktuelle Ist-Startzeit ist, jedoch gleichzeitig ebenfalls früher als die späteste Startzeit (siehe Abschnitt 5.2.2 letzter Absatz), sodass noch keine Verspätung auftritt. Wird auf allen möglichen Maschinen das Kriterium nicht erfüllt, wird der nächste verschiebbare Batch mit Lagerkosten betrachtet.

Rüstkosten

Für jeden Batch, für den ein Rüstvorgang stattfindet, wird auf den anderen möglichen Maschinen das benutzte Werkzeug des Batches mit der nächstkleineren Nummer gesucht. Ist dieses Identisch mit dem Prio-Werkzeug des betrachteten Batches, wird die Maschine ausgewählt und die restlichen Maschinen werden nicht mehr betrachtet. Ist auf allen anderen Maschinen ebenfalls ein Rüstvorgang notwendig, wird der nächste verschiebbare Batch mit Rüstkosten betrachtet.

Betriebskosten

Zur Erinnerung: es entstehen keine Betriebskosten, wenn ein Batch ohne Rüstvorgang mit dem Prio-Werkzeug von den möglichen Maschinen auf derjenigen Maschine mit dem geringsten Stundensatz produziert wird. Für jeden Batch mit Betriebskosten wird unter den möglichen Maschinen eine Maschine gesucht, die einen geringeren Stundensatz besitzt, als die Maschine, auf der der Batch aktuell eingeplant ist. Aus allen Maschinen, die dieses Kriterium erfüllen, wird zufällig eine ausgewählt. Erfüllt keine der Maschinen dieses Kriterium, wird der nächste Batch betrachtet.

Werkzeugwartezeiten

Da die Werkzeugbelegung im Mutationsoperator nicht berücksichtigt wird, wird zufällig ein verschiebbarer Batch mit einer Werkzeugwartezeit ausgewählt und zufällig auf eine der anderen möglichen Maschinen verschoben.

Dadurch, dass im Mutationsoperator die Werkzeugbelegung, das Rüstintervall und die Rüstmitarbeiter nicht berücksichtigt werden, ist nicht sichergestellt, dass die Mutation eines Batches auch wirklich zu einer Reduktion der betrachteten Kosten für den Batch führt. Jedoch wird dadurch einerseits viel Rechenzeit gespart und andererseits eine gute Mischung aus deterministischen und stochastischen Eigenschaften des Mutationsoperators erzielt.

5.6.3.3 Selection

Hier wird die *Tournament Selection* benutzt. Für diese wird eine festgelegte Anzahl an Individuen (*tournament size*) aus der Population ausgewählt und das Individuum mit dem besten Fitnesswert dient dann als Individuum für die Mutation bzw. den Crossover.[170] Für diesen Operator sind im GGA keine bestimmten Anpassungen vorgesehen, weshalb für die Experimentserien die Tournament Selection von Matlab ohne Änderungen übernommen werden kann.

[170] vgl. Falkenauer, 1998, S.50f

5.7 Untersuchung der Rüstmitarbeiter

Bevor mit den Experimentserien begonnen wird, wird die Relevanz der Rüstmitarbeiter untersucht. Insgesamt stehen vier Rüstmitarbeiter zur Verfügung, dementsprechend entsteht eine Wartezeit auf einen Rüstmitarbeiter, sobald sich mehr als vier Rüstvorgänge überschneiden. Das Ziel ist es herauszufinden, ob die Rüstmitarbeiter vernachlässigt werden können, da diese für die Bewertung der Individuen viel Zeit in Anspruch nehmen. Dies wird durch einen Kostenvergleich von Individuen mit und ohne Berücksichtigung der Rüstmitarbeiter erreicht. Dafür werden für jede der acht präsentierten initialen Populationen aus dem Abschnitt 5.6.1 100 Individuen erstellt und mit der Zielfunktion einmal mit und einmal ohne Berücksichtigung der Rüstmitarbeiter bewertet. Die Kostendifferenz der Kosten mit Berücksichtigung der Rüstmitarbeiter minus der Kosten ohne Berücksichtigung der Rüstmitarbeiter ist in Abbildung 5.6 zu sehen, wobei auf der x-Achse die insgesamt 800 Individuen der respektiven 8 initialen Populationen aufgetragen sind und auf der y-Achse die Differenz der betrachteten Kosten zu sehen ist. Da die Rüstmitarbeiter-Wartezeiten die Rüst- und Betriebskosten nicht beeinflussen, sind diese in Abbildung 5.6 nicht abgebildet.

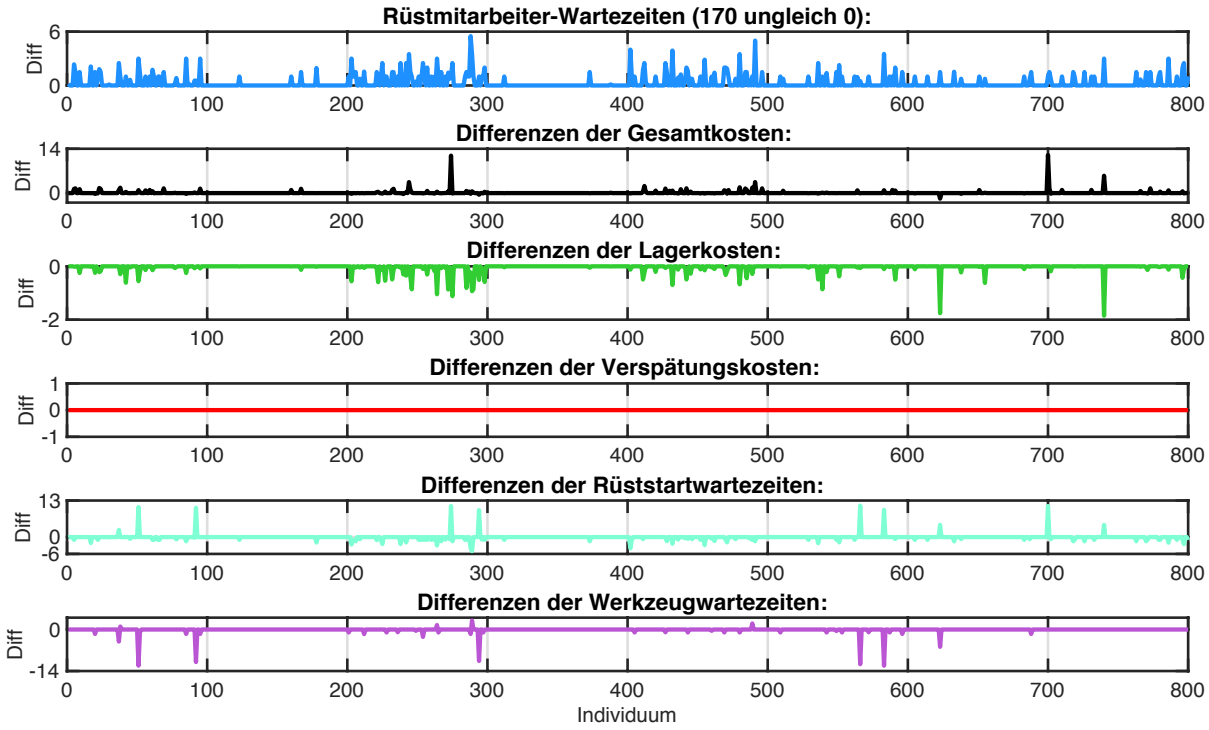

Abbildung 5.6: Kostendifferenz mit und ohne Berücksichtigung der Rüstmitarbeiter

Der Vergleich der 800 Individuen mit und ohne Berücksichtigung der Rüstmitarbeiter zeigt, dass die Wartezeiten auf Rüstmitarbeiter einerseits nur in ca. 20% der Fällen auftritt und sich nur sehr gering auf die Gesamtkosten auswirken. Dementsprechend werden im weiteren Verlauf die Rüstmitarbeiter vernachlässigt.

6 Kapitel 6

Experimentserien und Erweiterte Auswertungen

Im letzten Kapitel wurden die detaillierten Projektdaten sowie der Aufbau und Ablauf des in dieser Arbeit benutzten GA vorgestellt. In diesem Kapitel werden Experimentserien zur Bestimmung der Parameter des GA durchgeführt. Zusätzlich werden erweiterte Auswertungen vorgenommen, wie z.B. die Untersuchung der Batchrange auf die Ergebnisse des GA, sowie die Hybridisierung des GA mit einem LS. Letztendlich wird auch untersucht, inwiefern der GA wirklich besser ist als zufällig erstellte Individuen bzw. eine Kombination dieser mit einem LS.

6.1 Parameter Tuning Genetischer Algorithmus

Für die Bestimmung der Parameter werden diese während Testläufen variiert und deren Ergebnisse verglichen. Da die Ergebnisse von Metaheuristiken mehr oder weniger stark variieren, muss für jede Parameterkombination eine ausreichend große Anzahl an Testläufen durchgeführt werden, um aussagekräftige Ergebnisse zu erzielen. Für den GA werden folgende Ausgangsparameter benutzt:

- **Elite count** = 10: Definiert die Anzahl der besten Individuen (mit dem geringsten Fitnesswert) einer Population, die ohne Veränderung in die nächste Generation übertragen werden. In diesem Fall werden aus jeder Generation die 10 besten Individuen ausgewählt.
- **Crossover rate** = 0.8: Definiert in jeder neuen Generation den Anteil der Individuen (ohne die Elite-Individuen), die durch den Crossover erstellt werden. Die restlichen Individuen werden durch die Mutation erstellt. Dementsprechend besteht jede neue Generation aus 10 Elite Individuen und die restlichen 80% der Population werden durch den Crossover und 20% durch die Mutation erzeugt.
- **Tournament size** = 5: Wird im Selektionsoperator benutzt, und legt die Anzahl der Individuen fest, die zufällig aus der Population ausgewählt werden und gegeneinander in einem "Turnier" antreten (siehe auch Abschnitt 5.6.3.3).
- **Population size**: Gibt die Anzahl der Individuen in der Population an. Dies ist der erste Parameter, für den Testläufe durchgeführt werden und ist somit für den Anfang nicht definiert.

- **Number of generations** $= \frac{AnzahlBewertungen}{PopulationsGroesse}$: Definiert die Anzahl der Generationen, die vom GA durchlaufen werden. Um für die ersten Testläufe der variierenden Populationsgröße immer die gleiche Anzahl an bewerteten Individuen zu erzielen, wird diese auf 60.000 fixiert. Eine Bewertung findet dann statt, wenn durch die Zielfunktion der Fitnesswert eines Individuums ermittelt wird. Für eine Populationsgröße von z.B. 100 Individuen ergibt sich eine Anzahl von $\frac{60.000}{100} = 600$ Generationen.

6.1.1 Ermittlung der Populationsgröße

Es werden insgesamt sieben Populationsgrößen (40, 80, 160, 320 560, 800 und 1200) getestet, wobei aus jeder initialen Population aus Abschnitt 5.6.1 gleich viele Individuen vorhanden sind. Für jede Populationsgröße werden insgesamt 10 Populationen erstellt, für die jeweils 10 Testläufe durchgeführt werden. Dies ergibt insgesamt 100 Testläufe pro Populationsgröße.

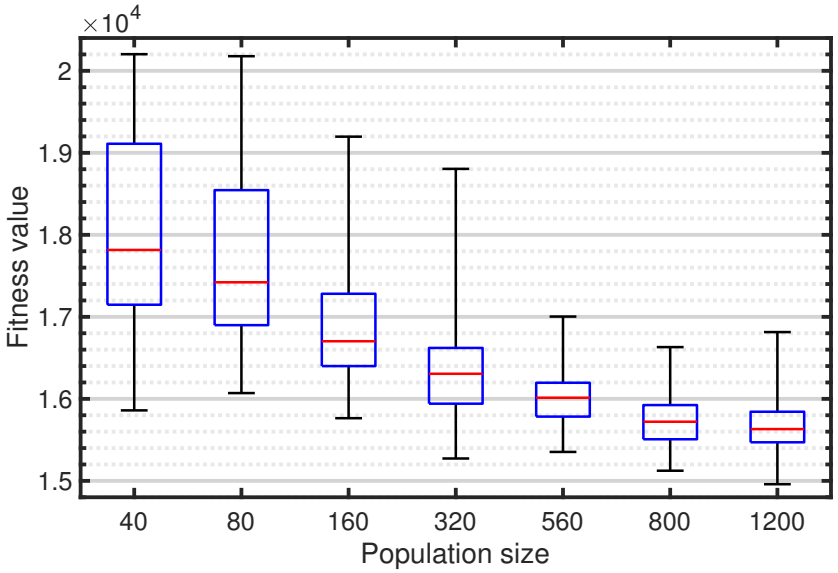

Abbildung 6.1: Ergebnisse der Testläufe für die Ermittlung der Populationsgröße

Die Ergebnisse der Testläufe sind in Abbildung 6.1 zu sehen (für eine Erklärung eines solchen Box-Plot Diagramms siehe Anhang 8.3). Auf der x-Achse befinden sich die Populationsgrößen und auf der y-Achse die Fitnesswerte der Ergebnisse der Testläufe. Sehr gut zu sehen ist, dass sowohl der Medianwert als auch die Varianz mit der Populationsgröße stetig abnimmt. Da sich der Medianwert zwischen den beiden Populationsgrößen 800 und 1200 nicht mehr viel verändert, wird für die weiteren Testläufe eine Populationsgröße von 800 gewählt, wobei hier jene Population verwendet wird, die den geringsten Durchschnittswert in den 10 Testläufen erzielt hat. Das beste Individuum hat in dieser initialen Population einen Fitnesswert von 19851. Des Weiteren hat sich für diese Populationsgröße gezeigt, dass der GA nach 40 Generationen nur sehr selten noch bessere Individuen findet, weshalb

die Anzahl der Generationen für die weiteren Testläufe auf 50 begrenzt wird, wodurch sich eine Anzahl von nur mehr 40.000 Bewertungen ergibt.

6.1.2 Ermittlung der Crossover rate

Für die Testläufe dieses Parameters werden insgesamt 11 Crossover rates (von 0 in 0.1 Schritten bis 1) betrachtet, wobei für jeden Wert insgesamt 30 Testläufe absolviert werden. Eine Crossover rate von 0 bedeutet, dass nur die Mutation zum Einsatz kommt und eine Crossover rate von 1 bedeutet, dass nur der Crossover eingesetzt wird.

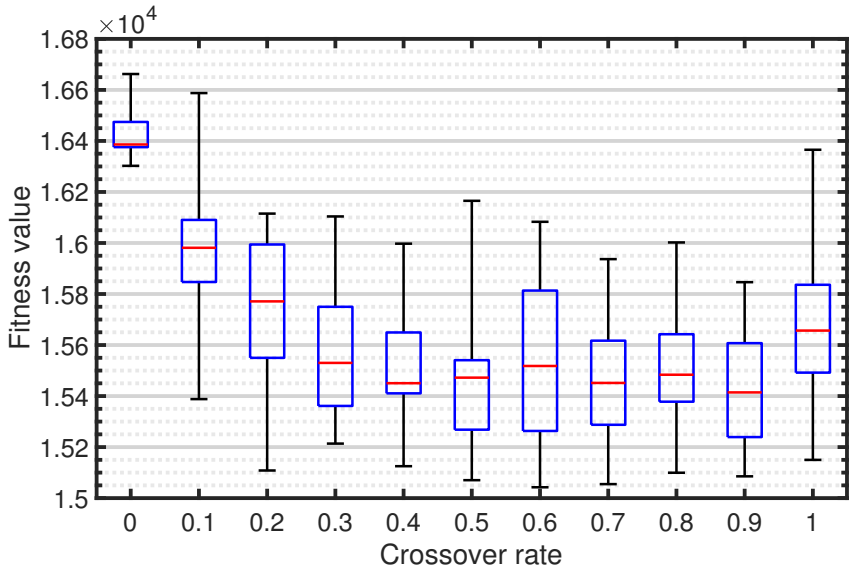

Abbildung 6.2: Ergebnisse der Testläufe für die Ermittlung der Crossover rate

Die schlechtesten Ergebnisse werden durch die Testläufe erzielt, bei denen nur die Mutation zum Einsatz kommt (siehe Crossover rate 0 in Abbildung 6.2). Dies lässt sich dadurch erklären, dass die Mutation immer nur einen Batch verschiebt und somit sehr leicht in einem lokalen Minimum stecken bleiben kann bzw. das Ergebnis nach 50 Generationen nicht weiter als eine Distanz von 50 zu einem Individuum der initialen Population entfernet sein kann. Nichtsdestotrotz liefert der GA in diesem Fall Ergebnisse mit der geringsten Varianz. Kommt nur der Crossover zum Einsatz, entsteht zwar eine viel größere Varianz, jedoch werden weitaus bessere Ergebnisse erzielt, als nur mit Mutation. Das spricht für das Prinzip des GA, bei dem der Crossover meist die wichtigste Rolle einnimmt. Zusätzlich ist dieses Ergebnis umso überraschender, da der benutzte Crossover rein stochastischer Natur ist und somit keine deterministische Komponente besitzt. Das Ergebnis aller Testläufe fällt weniger eindeutig aus, als für die Populationsgröße. Die Crossover rate von 0.9 weist im Vergleich zu den anderen Werten eine gute Verteilung des oberen und unteren Quartils, sowie den geringsten Medianwert auf, weshalb diese für die weiteren Testläufe übernommen wird.

6.1.3 Ermittlung der Elite-Anzahl

Um die Elite-Anzahl zu bestimmen, werden jeweils 30 Testläufe mit 0, 1, 10, 20, 40, 80, 160, 320 und 400 Elite-Individuen betrachtet. Die Ergebnisse sind in Abbildung 6.3 zu sehen.

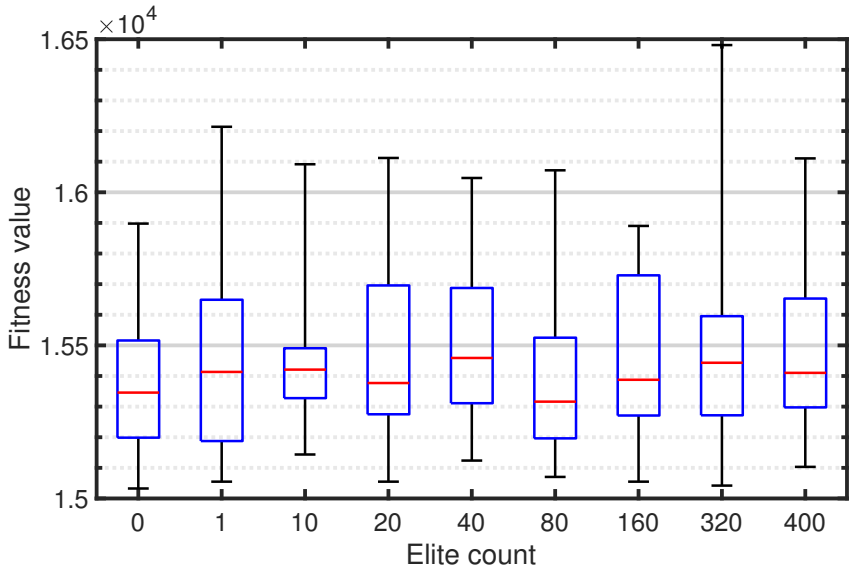

Abbildung 6.3: Ergebnisse der Testläufe für die Ermittlung der Elite-Anzahl

Überraschend fällt das Ergebnis der Testläufe mit 0 Elite-Individuen aus, da hier auch Verschlechterungen des besten Fitnesswertes von einer Generation zur nächsten möglich sind. Die Medianwerte aller Testläufe variieren nur mehr sehr wenig, wodurch nicht eindeutig festzustellen ist, welche Elite-Anzahl das größte Potential besitzt. Die Testläufe mit der Elite-Anzahl von 80 liefert den besten Medianwert, dicht gefolgt von den Testläufen mit der Eliteanzahl von 0. Da in diesen beiden Fällen die Varianz auch sehr ähnlich ausfällt, werden für den nächsten Parameter Testläufe sowohl mit einer Elite-Anzahl von 0 und 80 durchgeführt.

6.1.4 Ermittlung der Turniergröße

Für die Testläufe dieses Parameters werden insgesamt 9 Turniergrößen (1, 2, 5, 10, 20, 40, 80, 160 und 240) betrachtet, mit jeweils 30 Testläufen pro Turniergröße. In Abbildung 6.4 sind die Ergebnisse der Testläufe für die Elite-Anzahl von 0 zu sehen.

Wenig überraschend fällt das Ergebnis für eine Turniergröße von 1 aus, was mit einer vollkommen zufälligen Auswahl der Individuen zu vergleichen ist. Dadurch, dass die besten Individuen auch nicht gespeichert werden, sind diese Testläufe mit einer zufälligen Durchsuchung des Lösungsraums vergleichbar. In diesem Fall hat sich das beste Individuum im Vergleich zur initialen Population, wo der geringste Fitnesswert 19851 ausmacht, immer verschlechtert. Bereits eine Turniergröße von 2 erzielt eine ausschlaggebende Verbesserung

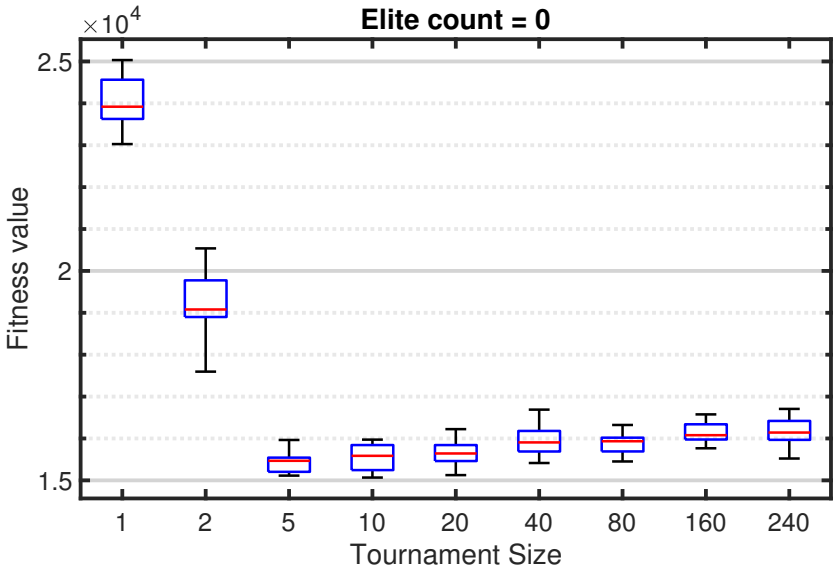

Abbildung 6.4: Ergebnisse der Testläufe für die Ermittlung der Tournament size (Elite count = 0)

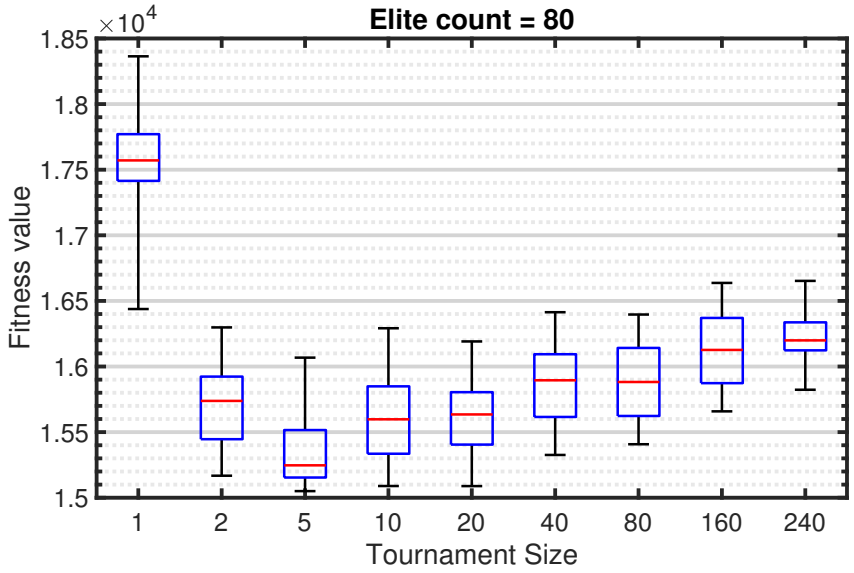

Abbildung 6.5: Ergebnisse der Testläufe für die Ermittlung der Tournament size (Elite count = 80)

der Ergebnisse, wobei diese noch immer im Vergleich zur initialen Population teilweise geringe Verbesserungen aber auch Verschlechterungen erzielt. Die besten Ergebnisse werden mit einer Turniergröße von 5 erzielt, wobei danach der Medianwert stetig ansteigt.

In Abbildung 6.5 sind die Ergebnisse der Testläufe mit einer Elite-Anzahl von 80 zu sehen. Durch das Speichern der besten Individuen, werden sogar mit einer zufälligen Selektion (tournament size 1) stetig bessere Individuen gefunden, als in der initialen Population. Ähnlich wie für die Testläufe ohne Elite-Individuen erzielt bereits eine Turniergröße von 2 eine deutliche Verbesserung der Ergebnisse. Die Turniergröße von 5 liefert hier ebenfalls die besten Ergebnisse, wobei diese wieder für größere Turniergrößen stetig schlechter werden.

Erweiterte Testläufe haben gezeigt, dass für die Turniergröße 3 und 4 im Durchschnitt schlechtere Ergebnisse gefunden werden als für die Turniergröße 5.

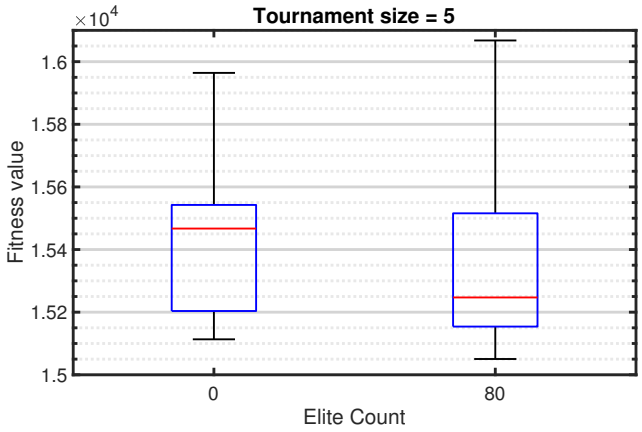

Abbildung 6.6: Vergleich der Ergebnisse für die Turniergröße 5 mit Elite count 0 und 80

In Abbildung 6.6 ist der Vergleich der Ergebnisse für die Turniergröße von 5 aus den beiden vorhergehenden Abbildungen zu sehen. Die Elite-Anzahl von 80 liefert, wie auch in den Testläufen für die Elite-Anzahl, einen besseren Medianwert. Dementsprechend wird die Elite-Anzahl von 80 übernommen.

6.1.5 Detaillierte Auswertung mit getunten Parametern

Die Testläufe für den GA führen zu folgenden Parametern:

- **Elite count** = 80**,**
- **Crossover rate** = 0.9,
- **Tournament size** = 5,
- **Population size** = 800**,**
- **Number of generations** = 50.

Für diese Parameter wurden 100 Testläufe absolviert, wobei für jeden Testlauf die detaillierten Einzelkosten gespeichert wurden. Ein Testlauf dauert im Durchschnitt 49 Sekunden. Für den Testlauf mit den geringsten Gesamtkosten ist der Optimierungsverlauf des besten Individuums in Abbildung 6.7 zu sehen. Die Achsenbeschriftung der Gesamtkosten gilt ebenfalls für die Einzelkosten. Auf der x-Achse sind die Generationen und auf der y-Achse die Kosten aufgetragen. Im Titel des jeweiligen Plots stehen die betrachteten Kosten mit dem erzielten Wert des besten Individuums in der letzten Generation. In Klammer befinden sich Zusatzinformationen. Für die Lagerkosten bezieht sich der erste Wert in der Klammer (575) auf die maximal auftretende Palettenanzahl (ohne dem konstanten Anteil der Paletten, siehe rote Linie im Beispiel in Abbildung 5.1) und der zweite Wert (19) auf die Periode, in der dieser auftritt. Die maximal auftretenden Paletten übersteigen somit die

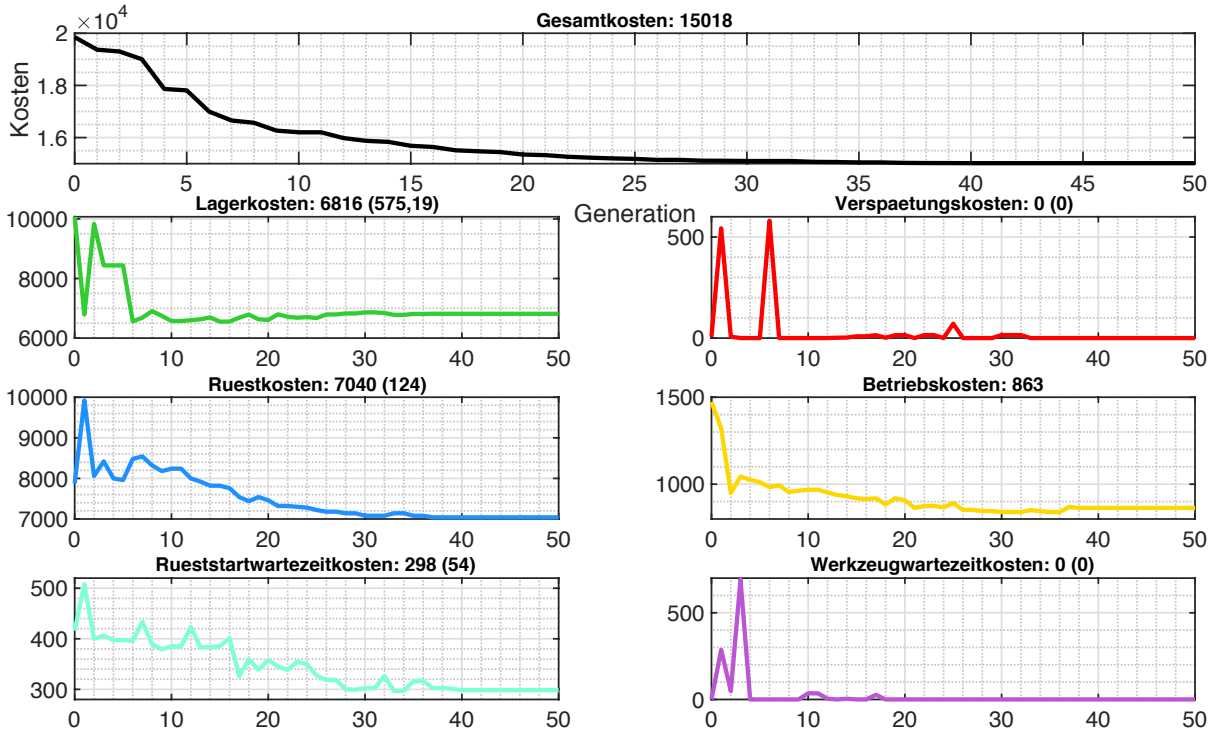

Abbildung 6.7: Optimierungsverlauf der besten Lösung der 100 Testläufe

Vorgabe der 500 Paletten, wobei leider nicht klar ist, wie viel fehlerhafte Stammdaten aus dem Excel-File dazu beitragen. Für die restlichen Kosten gibt der Wert in der Klammer die Anzahl an. Dementsprechend treten z.B. insgesamt 124 Rüstvorgänge auf, von denen insgesamt 54 eine Wartezeit auf Rüststart aufweisen. Der Verlauf der einzelnen Kosten zeigt, dass der GA es schafft, Individuen ohne Verspätungen und Werkzeugwartezeiten zu finden. Zusätzlich konnten im Vergleich zum besten Individuum in der initialen Population alle weiteren Kosten reduziert werden.

	GesK	**LagerK**	**MaxPal**	**Per**	**VerspK**	**RüstK**	**BetrK**	**RüstsWzK**	**WerkWzK**
Max	15.992	7.461	601	31	264	8.080	1.004	400	265
Mean	15.457	6.773	564	21	22	7.435	894	316	16
Min	15.018	6.280	524	19	0	7.040	847	229	0
Stdev	251	260	15	3	34	208	28	33	46

Tabelle 6.1: Übersicht der einzelnen Kosten aus den Ergebnissen der 100 Testläufe

Die gerundeten Maximal-, Minimal- und Durchschnittswerte sowie die Standardabweichung der einzelnen Kosten der 100 Testläufe sind in Tabelle 6.1 zu sehen. Zusätzlich sind diese Werte ebenfalls für die maximale Palettenanzahl und die Periode, in der diese auftritt, angegeben. In Tabelle 6.1 ist gut ersichtlich, dass zwar nicht alle Testläufe Individuen ohne Verspätungen und Werkzeugwartezeiten erzielen, jedoch ist der Durchschnittswert für diese beiden Kosten sehr gering. Folglich fallen diese Kosten, falls sie überhaupt anfallen, sehr gering aus. Die Lager- und Rüstkosten variieren am stärksten und beeinflussen dementsprechend die Ergebnisse auch am meisten. Die Rüststartwartezeit- und Betriebskosten

variieren am wenigsten. Die maximale Palettenanzahl aller Testläufe liegt über den 500 vorgegebenen Paletten, wobei diese immer in sehr frühen Perioden auftritt.

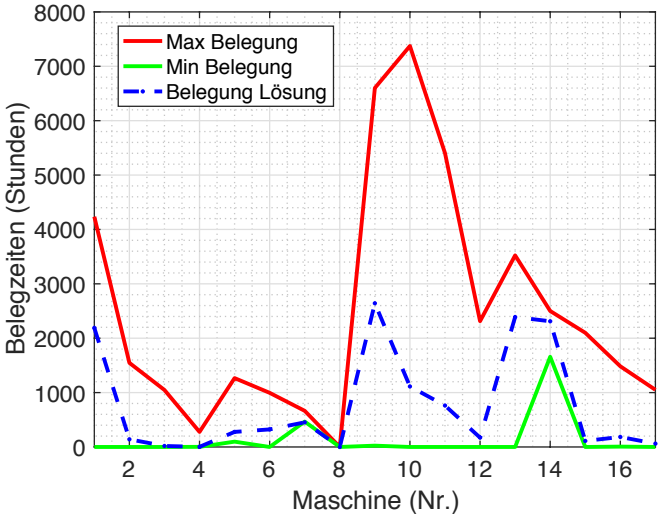

Abbildung 6.8: Maschinen Belegungszeiten der besten Lösung der 100 Testläufe

In Abbildung 6.8 sind in Blau die Belegzeiten der einzelnen Maschinen bezogen auf die Lösung mit den geringsten Gesamtkosten zu sehen. Die geringsten Gesamtkosten sind dementsprechend mit einer sehr ungleichmäßigen Belegung der Maschinen verbunden.

6.2 Erweiterte Auswertungen

Um zu untersuchen, inwiefern der GA mit den getunten Parametern noch verbessert werden kann, werden in diesem Abschnitt erweiterte Auswertungen durchgeführt. Zusätzlich werden die Auswirkungen des Batchprozesses auf die Ergebnisse des GA untersucht. Letztendlich wird der beste GA mit nur zufälligen und in Kombination mit einem LS erstellten Individuen verglichen. Die weiteren Testläufe des GA werden mit 75 statt den zuvor benutzten 50 Generationen durchgeführt.

6.2.1 Untersuchung der Batchrange

Der Batchprozess ist ein in der Praxis oft eingesetztes Mittel, um die Anzahl der Rüstvorgänge und die damit verbundenen Rüstkosten zu reduzieren. Dementsprechend ist es wichtig, die Auswirkungen des Batchprozesses auf die Ergebnisse des GA zu untersuchen und inwiefern die restlichen Kosten durch den Batchprozess beeinflusst werden.

In erster Linie ist es wichtig zu erwähnen, dass durch den Batchprozess der Lösungsraum, den der GA durchsuchen muss, deutlich verkleinert wird. Je mehr Aufträge in Batches zusammengefasst werden, umso weniger Batches gibt es und dementsprechend reduziert sich die Anzahl der Variablen und die damit verbundene Größe des Lösungsraums. Auf

der anderen Seite wird durch den Batchprozess der Lösungsraum eingeschränkt, was dazu führt, dass gewisse Lösungen ausgeschlossen werden und es somit passieren kann, dass das globale Optimum gar nicht dargestellt werden kann.

Batchrange	**0A**	**0B**	**4**	**6**	**10A**	**10B**
Anzahl der Batches	540	439	407	267	201	186
Lösungsraum	1.09×10^{222}	3.72×10^{188}	5.71×10^{172}	1.79×10^{111}	6.40×10^{81}	6.28×10^{79}

Tabelle 6.2: Auswirkungen der Batchrange auf die Größe des Lösungsraums

In Tabelle 6.2 sind die Auswirkungen der Batchrange auf die Größe des Lösungsraums ersichtlich. Für die Batchrange 0A werden keine Aufträge gebatcht, wodurch genauso viele Batches wie Aufträge entstehen. Bei der Batchrange 0B werden alle Aufträge, die gebatcht werden können und das gleiche Lieferdatum haben, in einem Batch zusammengefasst. Für die Batchrange 10A werden alle Aufträge, die gebatcht werden können und deren Lieferdatum sich innerhalb der Batchrange von 10 Tagen befindet, uneingeschränkt in einem Batch zusammengefasst. Wohingegen für die Batchrange 10B zusätzliche Einschränkungen berücksichtigt werden (siehe Abschnitt 5.2.4.4), wie z.B. eine maximale Gesamtproduktionszeit der Batches von 250 Stunden, sowie eine erweiterte Batchrange von 20 Tagen für Batches, deren Gesamtproduktionszeit unter 20 Stunden liegt.

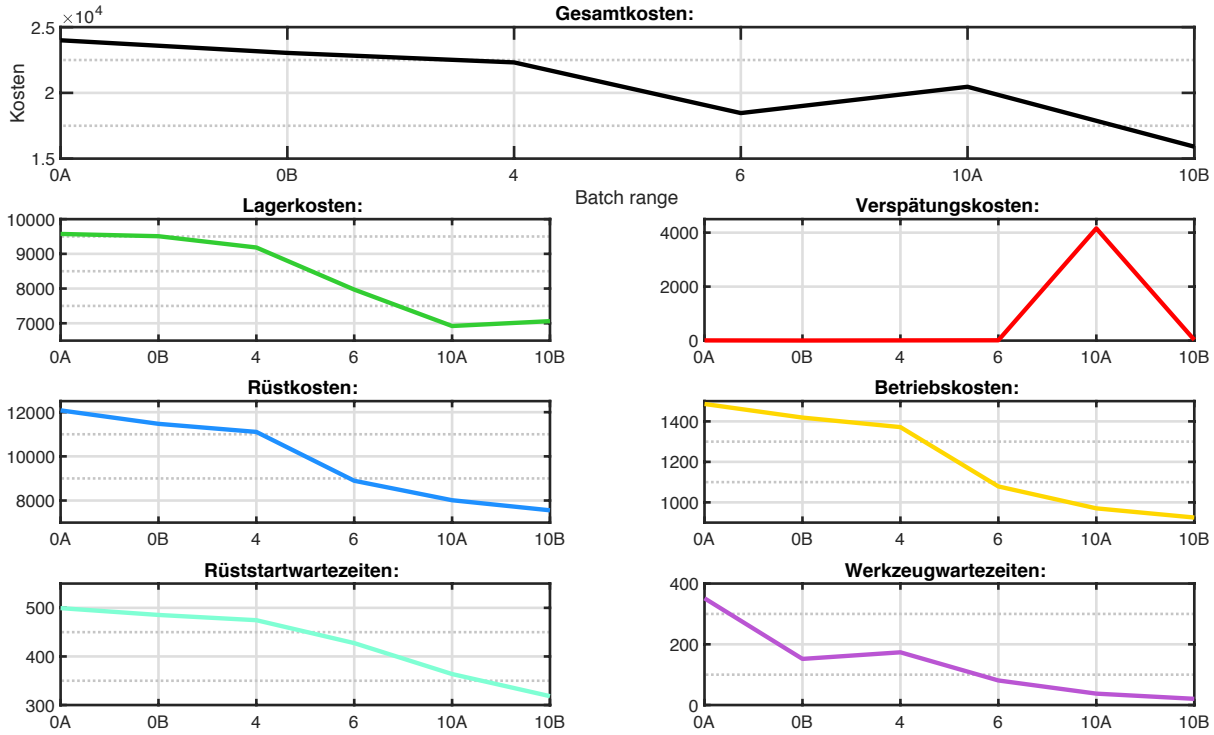

Abbildung 6.9: Auswirkungen der Batchrange auf die einzelnen Kosten

Um die Auswirkungen der Batchrange auf die einzelnen Kosten zu untersuchen, wurden für jede Batchrange aus Tabelle 6.2 10 initiale Populationen erstellt und jeweils 10 Testläufe absolviert. Die durchschnittlichen Kosten aller 100 Testläufe für jede Batchrange sind in

Abbildung 6.9 zu sehen. Erwartungsgemäß nehmen die Rüstkosten mit einer größeren Batchrange ab. Gut zu erkennen ist ebenfalls, dass für alle Batchranges (außer 10A) die Testläufe Lösungen mit sehr geringen bis keinen Verspätungen liefern. Dies zeigt, dass für die Batchrange 10A Batches mit einer zu großen Gesamtproduktionszeit entstehen, wodurch Verspätungen unvermeidbar werden (siehe auch Ergebnis des Kapazitätsabgleichs in Abschnitt 5.2.4.2 und 5.2.4.3). Jedoch verhindert eine sinnvoll gewählte Einschränkung des Batchprozesses, wie sie für die Batchrange 10B durchgeführt wurde, das Problem der unvermeidbaren Verspätungen und führt zu einer weiteren Reduzierung der Gesamtkosten. Überraschend ist, dass neben den Rüstkosten ebenfalls alle weiteren Kosten mit einer Vergrößerung der Batchrange abnehmen. Vor allem für die Lagerkosten wäre eigentlich eine Erhöhung dieser zu erwarten, da durch den Batchprozess Aufträge vorgezogen werden, wodurch diese früher produziert werden und somit höhere Lagerkosten entstehen. Eine mögliche Erklärung ist, dass der Batchprozess und die damit verbundene Reduktion des Lösungsraums dazu führen, dass der GA leichter gute Lösungen findet, bei denen nicht nur die Rüstkosten sondern alle Kosten reduziert werden können.

Die Untersuchung der Batchrange zeigt, dass eine sinnvolle Reduzierung des Lösungsraums zu erheblich besseren Lösungen führt, auch wenn dadurch gegebenenfalls sehr gute Lösungen ausgeschlossen werden.

6.2.2 Erweiterte initiale Population

Die initiale Population des GA sollte Individuen mit einer hohen Diversität und gutem (geringem) Fitnesswert aufweisen. Um die initiale Population aus 5.6.1 unter diesen beiden Aspekten weiter zu verbessern, werden folgende Änderungen vorgenommen:

Die Erstellung der acht einzelnen initialen Populationen bleibt unverändert, jedoch werden von jeder initialen Population 1000 Individuen erstellt (anstatt der 100 für eine gesamte initiale Population von 800). Die Individuen werden dann mit der Zielfunktion bewertet und nach den Gesamtkosten aufsteigend sortiert. Anschließend wird abwechselnd von jeder initialen Population, beginnend mit dem Individuum mit den geringsten Gesamtkosten, ein Individuum in die gesamte Population übergeben. Ein Individuum wird jedoch nur dann in die gesamte initiale Population aufgenommen, wenn dieses eine Mindestdistanz von 40 zu allen bereits aufgenommenen Individuen nicht unterschreitet. Wird die Mindestdistanz zu einem der bereits aufgenommenen Individuen unterschritten, wird das nächst schlechtere Individuum der betrachteten initialen Population untersucht. Dieser Vorgang wird so lange wiederholt, bis die 800 Individuen der gesamten initialen Population erreicht wurden. Das Resultat ist eine gesamte initiale Population mit besseren Individuen, wobei gleichzeitig sichergestellt wird, das diese möglichst gleichmäßig im Lösungsraum verteilt sind und dementsprechend eine hohe Diversität aufweisen.

Für die Untersuchung werden jeweils für die normale und die in diesem Abschnitt be-

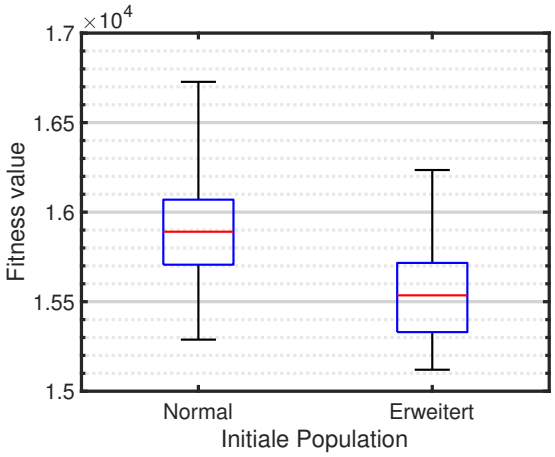

Abbildung 6.10: Vergleich normale und erweiterte initiale Population

schriebene erweiterte initiale Population 10 initiale Populationen erstellt und jeweils 10 Testläufe durchgeführt. Die Ergebnisse der jeweils 100 Testläufe sind in Abbildung 6.10 zu sehen. Die Testläufe des GA mit der erweiterten initiale Population liefern im Durchschnitt deutlich bessere Lösungen als die bisher benutzte initiale Population. Deshalb wird diese für die weiteren Auswertungen übernommen, wobei jene initiale Population verwendet wird, die den besten Durchschnitt in diesen Testläufen erzielt hat.

6.2.3 Erweiterte Selektion und Crossover

Der Crossover kann noch effektiver gestaltet werden, indem die Distanz zwischen zwei Elternteilen berücksichtigt wird. Haben zwei durch die Selektion ausgewählten Elternteile eine zu geringe Distanz bzw. sind sogar identisch, ist der Crossover ineffektiv. Um diesem Problem entgegenzuwirken, wird sowohl in der Selektion als auch für den Crossover die Distanz berücksichtigt.

Die Selektion der Individuen für die Mutation bleibt unverändert, wohingegen für den Crossover aktiv die Distanz zwischen den Elternteilen berücksichtigt wird. In einem ersten Selektionsprozess wird Elternteil A noch normal ohne Distanzberücksichtigung festgelegt. Für Elternteil B wird aktiv die Distanz zum Elternteil A berücksichtigt. Solange die Turniergröße nicht erreicht wurde, wird zufällig ein Individuum aus der Population ausgewählt. Haltet das Individuum eine Mindestdistanz von 10 zum Elternteil A ein, so wird dieses in das Turnier aufgenommen, ansonsten nicht. Ist das Turnier vollständig, wird das beste Individuum als Elternteil B ausgewählt.

Im Crossover wurde bis jetzt eine zufällige Anzahl an Maschinen ausgewählt, die vom Elternteil A dem Elternteil B übergeben werden (siehe Abschnitt 5.6.3.1). Ist die Belegung der ausgewählten Maschinen der beiden Elternteile identisch, ist der Crossover wiederum ineffektiv. Dementsprechend wird die Auswahl der Maschinen erweitert, indem nur die Maschinen für die Übergabe zur Auswahl stehen, die in beiden Elternteilen nicht identisch

belegt sind.

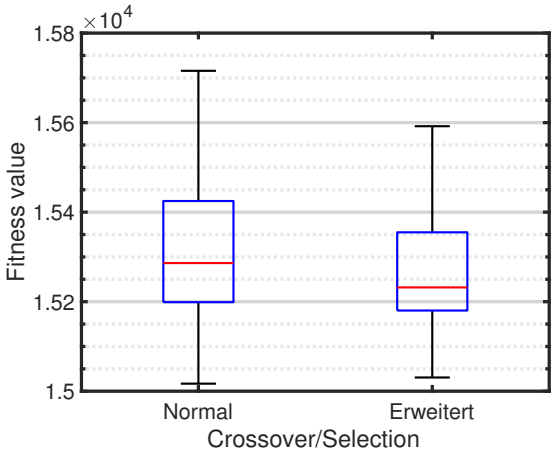

Abbildung 6.11: Vergleich Crossover/Selektion normal und erweitert

Es werden sowohl für den normalen und erweiterten Crossover- als auch Selektionsoperator 100 Testläufe durchgeführt. Die Ergebnisse sind in Abbildung 6.11 zu sehen. Die erweiterten Operatoren zeigen einen leicht besseren Medianwert und eine geringere Varianz. Dieses Ergebnis wird ebenfalls durch die durchschnittliche Periode, in der die letzte Verbesserung stattfindet, bestätigt, die für die erweiterten Operatoren in Periode 43 stattfindet (verglichen mit Periode 36 ohne Distanzberücksichtigung). Folglich kann durch die Distanzberücksichtigung der Lösungsraum effektiver durchsucht werden, jedoch bleibt das beste gefundene Individuum in beiden Fällen ca. gleich. Ein Nachteil aus der Berücksichtigung der Distanz ergibt sich durch die erhöhte Rechenzeit eines Testlaufes, die sich im Durchschnitt von 70 Sekunden für die normalen auf 90 Sekunden für die erweiterten Operatoren erhöht. Dennoch wird die Distanzberücksichtigung für die weiteren Testläufe übernommen, da sonst vor allem mit geringer werdender Diversität viele unnötige Crossover entstehen, die keine neuen Lösungen generieren.

6.2.4 Hybridisierung GA mit LS

Um die Tiefensuche des GA zu verbessern, wird dieser mit einem LS kombiniert. Es wird ein *first improvement* LS benutzt (siehe Definition S.50), wobei die Reihenfolge, in der die Batches betrachtet werden, zufällig am Anfang jedes LS definiert wird. Die betrachtete Nachbarschaft ist die kleinstmögliche und somit die Verschiebung von einem Batch. Der LS beginnt mit dem ersten Batch der zufällig definierten Reihenfolge. Jeder Batch wird auf den restlichen möglichen Maschinen eingeplant, solange keine bessere Lösung gefunden wird. Wird eine bessere Lösung gefunden, wird diese übernommen und der nächste Batch wird betrachtet, auch wenn noch weitere mögliche Maschinen übrig bleiben. Wurden alle Batches einmal betrachtet, beginnt der Vorgang wieder beim ersten Batch der zufällig definierten Reihenfolge. Ein lokales Minimum wird dann erreicht, wenn der LS alle Batches einmal durchlaufen hat, ohne eine Verbesserung gefunden zu haben.

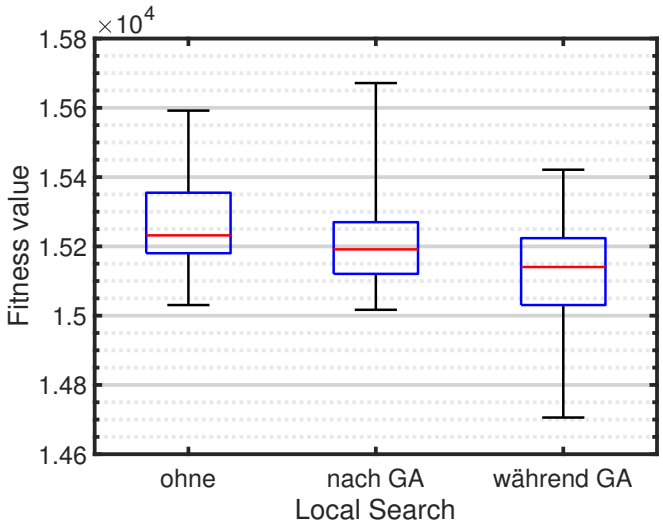

Abbildung 6.12: Vergleich Testläufe ohne und mit LS

Im ersten Szenario wird untersucht, wie weit die Ergebnisse des GA mit anschließendem LS noch verbessert werden können. Im zweiten Szenario wird der LS innerhalb des GA als Mutation verwendet. Es hat sich herausgestellt, dass ein LS Vorgang in jeder Generation zu zeitintensiv ist und der GA dadurch zu schnell konvergiert, weshalb der LS nur in vorgegebenen Generationen eingesetzt wird. Um in diesem Fall die Rechenzeit in Grenzen zu halten, werden die einzelnen LS-Vorgänge parallel berechnet. Der LS wird während dem GA, beginnend bei der ersten Generation, nur in jeder zweiten Generation eingesetzt, wobei für die ersten 6 Generationen der LS auf 10 Verbesserungen, dann bis zur 12. Generation auf 5 Verbesserungen und für die restlichen Generationen auf 2 Verbesserungen begrenzt wird. Für beide Szenarien werden 100 Testläufe absolviert, deren Ergebnisse in Abbildung 6.12 zu sehen sind. Im ersten Szenario hat der GA in 45 von den 100 Testläufen ein lokales Optimum erzielt. Für die restlichen Ergebnisse konnte der LS nur mehr geringe Verbesserungen erzielen, bis ein lokales Optimum gefunden wurde. Dementsprechend sind auch die Ergebnisse des GA mit anschließendem LS nur leicht besser als ohne, wobei insgesamt das beste gefundene Individuum ca. gleich bleibt. Erst durch den Einsatz vom LS während dem GA kann eine deutliche Verbesserung erzielt werden. Insgesamt erzielen in diesem Szenario 20 Lösungen einen Fitnesswert von unter 15.000. Ein Nachteil ist wiederum, dass sich die Rechenzeit für einen Testlauf deutlich von durchschnittlich 90 Sekunden ohne LS auf 170 Sekunden mit LS während dem GA erhöht.

6.2.5 Vergleich Hybridisierung GA+LS mit zwei anderen Methoden

Es werden in diesem Abschnitt drei Methoden zur Durchsuchung des Lösungsraums miteinander verglichen. Um einen fairen Vergleich zu erzielen, werden in diesem Fall nicht die Anzahl der Bewertungen, sondern die Rechenzeit benutzt. Ein Testlauf vom GA mit LS als Mutation (GA+LS) dauert im Durchschnitt 170 Sekunden, was als Basis für

den Vergleich mit den beiden anderen Methoden benutzt wird. Für die erste Methode (Random) werden 170 Sekunden lang zufällige Individuen erstellt und mit der Zielfunktion bewertet. Nach 170 Sekunden wird der Fitnesswert des besten Individuums gespeichert und der Vorgang beginnt wieder von neuem. Dies wird insgesamt 100 Mal wiederholt, um die Ergebnisse mit den 100 Testläufen des GA vergleichen zu können. Für die zweite Methode (Random+LS) werden zufällig Individuen erstellt und anschließend mit dem LS bis zu einem lokalen Optimum verbessert. Nach 170 Sekunden wird wieder der beste Fitnesswert gespeichert. Dieser Vorgang wird wiederum 100 Mal wiederholt.

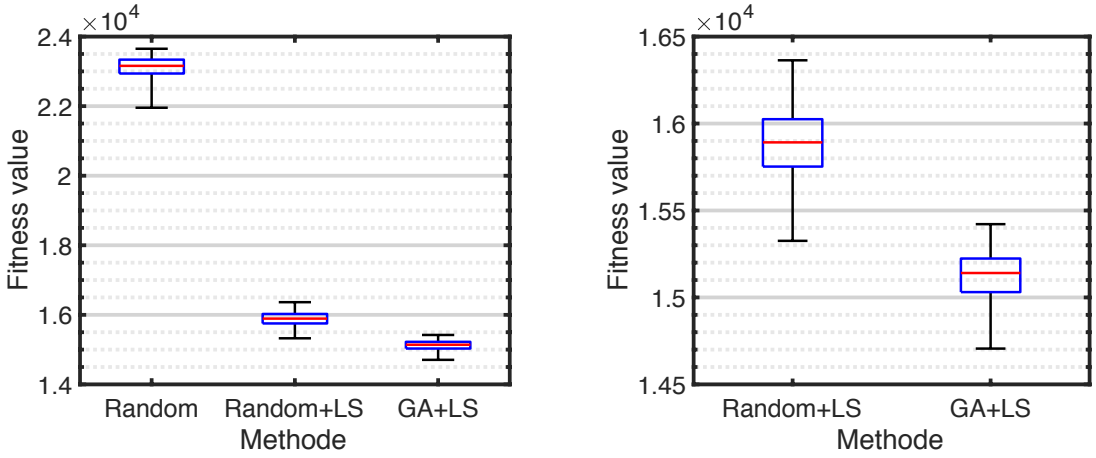

Abbildung 6.13: Vergleich von drei Durchsuchungsmethoden des Lösungsraums

Die Ergebnisse der jeweils 100 Testläufe der drei Methoden sind links in Abbildung 6.13 zu sehen. Wenig überraschend und deutlich zu erkennen ist, dass die zufällige Durchsuchung des Lösungsraums keine zufriedenstellenden Ergebnisse liefert. Wohingegen die zweite Methode (Random+LS) bereits deutlich bessere Ergebnisse erzielt, die mit dem GA mithalten können. Rechts in Abbildung 6.13 ist für eine bessere Lesbarkeit der Vergleich zwischen Random+LS und GA+LS ersichtlich. Auch wenn der alleinige Einsatz von LS bereits gute Ergebnisse liefert, erzielt die Kombination aus GA und LS nochmal eindeutig bessere Ergebnisse.

6.3 Zusammenfassung des endgültigen Algorithmus dieser Arbeit

Durch den Batchprozess und das vorgestellte Grundgerüst des GA in Kapitel 5 und die in diesem Kapitel durchgenommenen Erweiterungen, ergibt sich folgender Pseudo-Code des endgültigen Algorithmus dieser Arbeit (siehe Algorithmus 6.1). Die Reihenfolge in der die einzelnen Aufträge den Maschinen zugeordnet werden, wird über den Batchprozess festgelegt. Danach wird der GA für die Lösung der Maschinenbelegungsplanung eingesetzt, sprich welche Batches auf welchen Maschinen eingeplant werden sollen.

1: Vor dem GA wird ein Batchprozess durchgeführt, der einzelnen Aufträge in Batches zusammenfasst (siehe Abschnitt 5.2.2 und 5.2.4.4). Die Reihenfolge, in der die Batches

Algorithm 6.1 Pseudo-Code des Algorithmus dieser Arbeit

1: *Batchprocess* //Durchführung des Batchprozesses (siehe Abschnitt 5.2.2 und 5.2.4.4)
2: $Gen \leftarrow 0$
3: $initialize(Pop(Gen))$ //Erstellen der initialen Population (siehe Abschnitt 5.6.1 und 6.2.2)
4: $decode(Pop(Gen))$ //Decodieren der initialen Population (siehe Abschnitt 5.3.2)
5: $evaluate(Pop(Gen))$ //Bewertung der initialen Population (siehe Abschnitt 5.4)
6: **while** $Gen < MaxGen$
7: $Gen \leftarrow Gen + 1$
8: $Indiv_S \leftarrow selection(Pop(Gen-1))$ //Selektion für Mutation und Crossover (siehe Abschnitt 5.6.3.3 und 6.2.3)
9: $Indiv_{crossover} \leftarrow crossover(Indiv_S(crossover))$ //Durchführung des Crossovers (siehe Abschnitt 5.6.3.1 und 6.2.3)
10: **if** $Gen \in Gen_{LS}$
11: $Indiv_{mutation} \leftarrow localsearch(Indiv_S(mutation))$ //Durchführung des LS (siehe Abschnitt 6.2.4)
12: **else** //sonst normale Mutation durchführen
13: $Indiv_{mutation} \leftarrow mutation(Indiv_S(mutation))$ //Durchführung der Mutation (siehe Abschnitt 5.6.3.2)
14: **end if**
15: $Indiv_{elite} \leftarrow elite(Pop(Gen-1))$ //Definieren der elite-Individuen, die ohne Veränderung übernommen werden
16: $Pop(Gen) \leftarrow Indiv_{crossover} + Indiv_{mutation} + Indiv_{elite}$ //Zusammenstellen der neuen Population
17: $decode(Pop(Gen))$ //Decodieren der neuen Population (siehe Abschnitt 5.3.2)
18: $evaluate(Pop(Gen))$ //Bewertung der neuen Population (siehe Abschnitt 5.4)
19: **end while**

den Maschinen zugeordnet werden, wird dann fixiert, wodurch ebenfalls die Reihenfolge der einzelnen Aufträge feststeht.

3: Zu Beginn des GA wird die initiale Population erstellt. Diese setzt sich aus den acht in Abschnitt 5.6.1 beschriebenen Populationen zusammen. Darunter sind Populationen mit der zufälligen Verteilung von einzelnen Batches und ganzen Werkzeugen, eine Population mit einer möglichst gleichmäßigen Verteilung der Batches und Populationen, die mit einer *greedy* Heuristik erstellt werden und sich auf einzelne Kosten konzentrieren. Um die initiale Population in Hinsicht auf die Fitnesswerte und Diversität der Individuen zu verbessern, wurden die in Abschnitt 6.2.2 erläuterten Erweiterungen übernommen. Es werden von jeder initialen Population zehn Mal mehr Individuen erstellt als benötigt und nur die besten 10% kommen in die endgültige initiale Population. Jedoch wird ebenfalls darauf geachtet, dass eine gewisse Diversität sichergestellt ist, weshalb ein Individuum nur dann in die endgültige initiale Population aufgenommen wird, wenn es eine gewisse Mindestdistanz zu allen bereits aufgenommenen Individuen der endgültigen initialen Population einhält.

4: Bevor die Individuen einer Population bewertet werden können, müssen diese decodiert werden. Dafür wird der in Abschnitt 5.3.2 beschriebene Decodierungsprozess durchgeführt, der sicherstellt, dass alle Restriktionen berücksichtigt werden und so jedem Individuum einen zulässigen Produktionsplan zuordnet. Um die Laufzeit des GA möglichst gering zu halten, wird der Decodierungsprozess der einzelnen Individuen mit der in Matlab vorhandenen *parfor-Schleife* parallel berechnet.

5: Danach können die Individuen mit der Zielfunktion aus Abschnitt 5.4 bewertet werden, die die Summe der einzelnen Kosten berechnet. Die Bewertung wird, wie der Decodierungsprozess, ebenfalls parallel durchgeführt.

8: Die Selektion der Individuen für die Mutation und den Crossover wird durch eine *Tournament Selection* umgesetzt (siehe Abschnitt 5.6.3.3). Um unwirksame Crossover zu vermeiden, wird durch die erweiterte Selektion in Abschnitt 6.2.3 sichergestellt, dass eine gewisse Mindestdistanz zwischen zwei Elternteilen besteht.

9: Der benutzte Crossover basiert auf dem Prinzip des GGA, der nur ganze Gruppen (Maschinen) von einem Elternteile zum anderen übergibt (siehe Abschnitt 5.6.3.1). Um hier wiederum unwirksame Crossover zu vermeiden, stehen nur die Maschinen für die Übergabe zur Auswahl, die sich in den beiden Elternteilen auch wirklich unterscheiden (siehe Abschnitt 6.2.3).

11: Es hat sich herausgestellt, dass ein LS Vorgang in jeder Generation zu zeitintensiv ist und der GA dadurch zu schnell konvergiert. Aus diesem Grund findet nur in vorgesehenen Generationen ein LS Vorgang statt, wie er in Abschnitt 6.2.4 beschrieben ist. Da die LS Vorgänge auch in diesem Fall sehr zeitintensiv sind, werden diese parallel berechnet.

13: Findet kein LS Vorgang statt, wird eine klassische Mutation des GGA benutzt, die einzelne Batches auf andere Maschinen verschiebt. Dabei werden aktiv Informationen aus dem Decodierungsprozess genutzt, um gezielt Kosten von einzelnen Batches zu reduzieren (siehe Abschnitt 5.6.3.2).

15-18: Die besten Individuen werden ohne Veränderung in die nächste Generationen übernommen. Danach ist die neue Population vollständig, wodurch erneut der Decodierungs- und Bewertungsprozess aus Zeile 3 und 4 durchgeführt werden kann.

7 Kapitel 7

Schlussfolgerungen und Ausblick

7.1 Schlussfolgerungen

Das Hauptziel dieser Arbeit war es, eine Metaheuristik an die konkreten Anforderungen eines kunststoffverarbeitenden Unternehmens anzupassen und alle relevanten Kosten und Restriktionen, die in der Produktion auftreten, zu berücksichtigen. Dies war durch den Einsatz eines hybridisierten GA mit einem LS gut umsetzbar. Die Reihenfolgenplanung konnte über einen sinnvollen Batchprozess vor dem Einsatz der Metaheuristik definiert werden. In der Literatur gilt das Berücksichtigen von problemspezifischen Strukturen und Informationen als ausschlaggebend für eine effektive Metaheuristik. Diese beiden Aspekte konnten im Crossover und der Mutation des GA umgesetzt werden. Des Weiteren konnte eine ausgewogene Breiten- sowie Tiefensuche durch eine intensive Auseinandersetzung mit der initialen Population (starten mit bereits guten Individuen, die eine hohe Diversität aufweisen) sowie dem Einsatz eines LS erreicht werden. Durch die Berücksichtigung der Distanzen zwischen den Individuen in der initialen Population, sowie während dem GA, konnte ebenfalls eine effektivere Durchsuchung des Lösungsraums erzielt werden. Mit Hilfe eines zeitintensiven aber notwendigen Decodierungsprozesses war eine Berücksichtigung aller relevanten Restriktionen möglich. Zusätzlich konnte mittels einer in Matlab möglichen Parallelisierung durch *parfor-Schleifen* die Rechenzeit der Decodierungs- und Kostenbewertungsprozesse sowie der LS Vorgänge reduziert werden, wodurch die Laufzeit der Metaheuristik ausreichend in Grenzen gehalten werden konnte. Dieser Vorteil kann in Metaheuristiken, die nur eine Lösung gleichzeitig betrachten, nicht genutzt werden.

Die Testläufe haben gezeigt, dass alle Einzelkosten der Zielfunktion effektiv reduziert werden konnten. Außerdem weisen die Ergebnisse Produktionspläne mit keinen bis nur sehr geringen Verspätungs- und Werkzeugwartezeitkosten auf. Die maximal auftretende Palettenanzahl im Lager zeigt in allen Ergebnissen eine Überschreitung der 500 vorgesehenen Paletten, wobei leider nicht klar ist, wie viel davon durch fehlerhafte Stammdaten aus dem Excel File zustande kommt. Da für die betrachteten Aufträge kein realer Produktionsplan vom Unternehmen zur Verfügung steht, konnte kein Kostenvergleich durchgeführt werden. Außerdem ist es leider nicht möglich zu beurteilen, wie weit das beste Ergebnis des GA von den minimal möglichen Gesamtkosten entfernt ist, da diese nicht bekannt sind.

Durch die nicht vernachlässigbaren Rüstvorgänge ist unter anderem eine klassische Produktionsglättung, wie sie in Kapitel 2 beschrieben wurde, nicht möglich. Dennoch hat sich gezeigt, dass durch einen sinnvollen Batchprozess und der damit verbundenen Reduktion des Lösungsraums nicht nur die Rüstkosten sondern alle Einzelkosten deutlich reduziert werden konnten. Demnach ist eine angemessene Reduktion des Lösungsraums für praktische Probleme zielführend, auch wenn dadurch gegebenenfalls sehr gute Lösungen ausgeschlossen werden.

7.2 Ausblick

Die Resultate dieser Diplomarbeit zeigen vielversprechende Ergebnisse, jedoch darf nicht vergessen werden, dass hier *deterministisches Scheduling* durchgeführt wurde und somit Unsicherheiten nicht berücksichtigt wurden. Da Unsicherheiten in der Praxis zum Alltag gehören, kann das zu unbrauchbaren Produktionsplänen führen. Deshalb wäre der nächste wichtige Schritt zu überprüfen, wie zuverlässig die in dieser Arbeit erstellten Produktionspläne in der Praxis umgesetzt werden können und wie auftretende Unsicherheiten letztendlich im Optimierungsprozess berücksichtigt werden können. Dennoch ist deterministisches Scheduling beim Auftreten von Unsicherheiten nicht völlig unbrauchbar. Je nachdem wieviel Zeit zur Verfügung steht, wäre es denkbar, beim Auftreten einer ausschlaggebenden Unsicherheit einen erneuten Optimierungsprozess einzuleiten, der die Auswirkungen der aufgetretenen Unsicherheit berücksichtigt.

Ist eine ausreichende Sicherheit in der Umsetzbarkeit der Produktionspläne in der Realität gegeben, könnte der GA dann weiter optimiert werden. Ein großes Verbesserungspotential besteht im effektiveren Einsatz des LS während dem GA. Hier wäre der Einsatz einer angepassten Selektion für den LS von Vorteil, die unter bestimmten Kriterien entscheidet, welche Individuen das größte Potential für einen intensiven LS Vorgang aufweisen. Im Laufe dieser Arbeit hat sich gezeigt, dass durch intensive LS Vorgänge die Diversität in der Population sehr schnell verloren geht, weshalb ein intensiver LS nur in Kombination mit einer aktiven Aufrechterhaltung der Diversität zielführend wäre. Dies wäre unter anderem durch eine Variation der Parameter während eines GA-Laufs möglich. Letztendlich stellt die aktive Suche der Pareto-Front, wie sie in NSGA (Non-dominated Sorting Genetic Algorithm) II und III umgesetzt wird, ein vielversprechendes Potential dar, das im Rahmen dieser Arbeit leider nicht mehr untersucht werden konnte.

8 Kapitel 8

Anhang

8.1 Notation und Framework von Scheduling Problemen

Das folgende Framework basiert auf dem vorgestellten Framework von Pinedo (2016), das eine sehr weit verbreitete Notation von Scheduling Modellen darstellt.[171]

Spezifikationen des α-Feldes

Folgende Spezifikationen sind im α-Feld in der Literatur üblich (in der Klammer befindet sich die Abkürzung, die dann in den jeweiligen Feldern angegeben wird). Da die Benennungen in der Literatur hauptsächlich auf Englisch vorzufinden sind, werden diese in Folge auch auf Englisch gelassen.

Single machine (1): Wird nur eine Maschine betrachtet ($m = 1$), spricht man von einem single machine Problem. Das single machine Problem ist das einfachste von allen Maschinenumgebungen.

Identical machines in parallel (Pm): In diesem Fall werden m identische parallel laufende Maschinen betrachtet. Der Job j muss auf nur einer der m Maschinen bearbeitet werden bzw. auf einer Maschine, die zu einer bestimmten Maschinengruppe gehört (in diesem Fall steht im β-Feld der Eintrag M_j).

Machines in parallel with different speeds (Qm): In diesem Fall werden noch immer parallel laufende Maschinen betrachtet, jedoch haben die m Maschinen verschiedene Bearbeitungsgeschwindigkeiten. Die Geschwindigkeit von Maschine i wird mit v_i gekennzeichnet. Die Zeit p_{ij}, die ein Job j auf Maschine i verbringt wird als p_j/v_i gekennzeichnet. Diese Maschinenumgebung wird in der Literatur auch oft als *uniform machines* bezeichnet. Wenn alle Maschinen die gleiche Bearbeitungsgeschwindigkeit haben ($v_i = 1$ für alle i) dann ist $p_{ij} = p_j$ und die Umgebung ist gleich mit der Pm Umgebung.

Unrelated machines in parallel (Rm): Diese Maschinenumgebung ist eine Spezialfall der Qm Umgebung. Es laufen immer noch m verschiedene Maschinen parallel, jedoch hängt die Bearbeitungsgeschwindigkeit einer Maschine i vom Job j ab, das heißt dass

[171] Ursprünglich entstand diese *Three-Field-Notation* in: R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics, 287–326, 1979.

Maschine i den Job j mit der Geschwindigkeit v_{ij} bearbeitet. Die Zeit p_{ij}, die ein Job j auf der Maschine i verbringt ist somit gleich p_j/v_{ij}. Wenn die Geschwindigkeit der Maschinen unabhängig von den Jobs ist ($v_{ij} = v_i$ für alle i und j) dann ist diese Umgebung gleich der Qm Umgebung.

Flow Shop (Fm): Es befinden sich m Maschinen in Serie und jeder Job muss auf allen m Maschinen bearbeitet werden. Alle Jobs müssen somit die selbe Route durchlaufen. Wenn ein Job auf einer Maschine fertig ist, kommt er in die Warteschlange für die Bearbeitung der nächsten Maschine. Die Warteschlangen zwischen den Maschinen verfolgen üblicherweise das *First In First Out* (FIFO) Prinzip. In diesem Fall wird der flow shop auch als *permutation* flow shop bezeichnet und im β-Feld steht der Eintrag *prmu*.

Flexible flow Shop (FFc): Die FFc Umgebung ist eine Mischung aus der Fm und der Pm Umgebung. Anstatt m parallel laufender Maschinen in Serie werden c Stationen in Serie betrachtet, die jede für sich identische parallel laufende Maschinen beinhalten. Jeder Job muss in jeder Station bearbeitet werden, wobei jede der Maschinen in einer Station zur Bearbeitung möglich ist. Diese Maschinenumgebung wird in der Literatur auch als *hybrid flow shop* oder *multi-processor flow shop* bezeichnet.

Job Shop (Jm): In einem Job Shop mit m Maschinen hat jeder Job eine festgelegte Route. Es wird unterschieden zwischen Job Shops, in denen jeder Job jede Maschine maximal ein Mal durchläuft und Job Shops in denen ein Job eine Maschine mehrmals durchlaufen kann. In diesem Fall spricht man von *recirculation*, was durch den Eintrag *rcrc* im β-Feld wiedergegeben wird.

Felxible Job Shop (FJc): Die FJc Umgebung ist ein Mischung aus der Jm und Pm Umgebung. Es gibt insgesamt c Stationen, wobei jede Station eine bestimmte Anzahl an identisch parallel laufenden Maschinen beinhaltet. Jeder Job hat eine festgelegte Route, die durch die c Stationen führt, wobei jeder Job in jeder Station auf jeder der Maschinen bearbeitet werden kann. Wenn ein Job in seiner Route eine Station mehrmals durchlaufen kann, steht im β-Feld der Eintrag *rcrc* (recirculation).

Spezifikationen des β-Feldes

Im β-Feld können folgende Spezifikationen für die durchzuführenden Jobs angegeben werden:

Processing time (p_{ij}): p_{ij} gibt die Bearbeitungszeit von Job j auf Maschine i an. Der Index i wird weggelassen (p_j), wenn die Bearbeitungszeit von einem Job nicht von der Maschine abhängt bzw. die Jobs jeweils nur auf einer bestimmten Maschine bearbeitet werden können.

Release dates (r_j): Wenn dieser Ausdruck im β-Feld vorkommt, bedeutet das, dass ein Job j nicht vor einem bestimmten Zeitpunkt r_j begonnen werden kann. Für den Fall, dass r_j nicht angegeben ist, kann jeder Job jederzeit begonnen werden. Im Gegensatz werden Liefertermine (due dates d_j) in diesem Feld oft nicht spezifiziert, da anhand der zu

optimierenden Kriterien im γ Feld eindeutig ist, ob Liefertermine berücksichtigt werden oder nicht.

Preemptions (*prmp*): Preemption bedeutet, dass ein Job jeder Zeit unterbrochen werden kann und zu einem späteren Zeitpunkt auf der selben Maschine oder auf einer anderen (im Falle von parallel laufenden Maschinen) beendet werden kann. Muss ein Job beendet werden, sobald er begonnen hat, wird *prmp* im β-Feld nicht angegeben.

Precedence constraints (*prec*): Precedence constraints treten dann auf, wenn ein oder mehrere Jobs durchgeführt werden müssen, bevor ein anderer begonnen werden kann. Konkret bedeutet das, dass z.B. ein Fahrradreifen fertig produziert werden muss, bevor er am Fahrrad montiert werden kann. Erscheint *prec* im β-Feld nicht, muss keine bestimmte Reihenfolge der Jobs beachtet werden.

Sequence dependent setup times (s_{jk}): In diesem Fall sind die Rüstzeiten von der Reihenfolge der jobs abhängig. s_{jk} repräsentiert die Rüstzeit zwischen Job j und k. Ist die Rüstzeit zusätzlich von der Maschine abhängig, wird der Index i hinzugefügt (s_{ijk}).

Job families ($fmls$): In diesem Fall gehören die n Jobs zu F verschiedenen Job Familien. Jobs der gleichen Familie können verschiedene Bearbeitungszeiten haben, benötigen jedoch keine Rüstvorgänge zwischen den einzelnen Jobs. Rüstvorgänge finden somit nur statt, wenn von einer Job Familie auf eine andere gewechselt wird.

Batch Processing ($batch(b)$): Kann eine Maschine mehrere (b) Jobs gleichzeitig bearbeiten, spricht man von batchen. Da Jobs oft verschiedene Bearbeitungszeiten haben, dauert ein batch oft so lange, wie der längste Job im batch. Für $b = 1$ ist das Problem wie ein übliches Problem ohne batchen.

Breakdowns ($brkdwn$): Maschinenausfälle führen dazu, dass Maschinen nicht durchgehend einsatzbereit sind. In diesem Fall sind die Zeiten, in denen die Maschinen nicht einsatzbereit sind, fixiert (nicht zu verwechseln mit einem stochastischen Maschinenausfall). Zum Beispiel können das fixierte Instandhaltungszeiten sein. Die Maschinenverfügbarkeit kann also als eine Funktion $m(t)$ gesehen werden, die abhängig von der Zeit ist und zu jedem Zeitpunkt t die genaue Anzahl an verfügbaren Maschinen bekannt ist. Deswegen ist in der Literatur diese Spezifikation auch oft als *machine availability constraints* bekannt.

Machine eligibility restriction (M_j): Diese Spezifikation tritt im Maschinenumfeld mit m parallel laufenden Maschinen auf, wenn nicht jede Maschine jeden Job bearbeiten kann. M_j gibt die Maschinen an, die den Job j bearbeiten können. Wird M_j im β-Feld nicht angegeben, kann jeder Job auf jeder Maschine abgewickelt werden.

Permutation ($prmu$): Permutation wird im flow Shop Maschinenumfeld angegeben, wenn die Warteschlangen vor jeder Maschine nach dem FIFO-Prinzip ablaufen. Das heißt, dass die Reihenfolge, mit der die Jobs durch die erste Maschine gehen, in Folge auf allen weiteren Maschinen beibehalten wird.

Blocking (*block*): Blocking wird ebenfalls im flow Shop Maschinenumfeld angegeben, wenn zwischen den Maschinen nur ein begrenzter Puffer vorhanden ist. Ist dieser Puffer voll, kann die vorangehende Maschine keine weiteren Jobs mehr abwickeln und ist somit blockiert. Blocking wird oft dann berücksichtigt, wenn überhaupt kein Puffer zwischen den Maschinen vorhanden ist. In diesem Fall sollte die Abwicklung zwischen den Maschinen nach dem JIT-Prinzip verlaufen (sonst entsteht eine Blockade), weswegen die Spezifikation *block* in der Literatur oft das FIFO-Prinzip impliziert, also immer in Verbindung mit *prmu* betrachtet wird.

No-wait (*nwt*): Die No-wait Bedingung tritt ebenfalls meistens in flow Shop Umgebungen auf. In diesem Fall dürfen Jobs zwischen zwei Maschinen nicht warten. Das heißt, die Startzeit der ersten Maschine muss so gewählt werden, dass es auf den folgenden Maschinen zu keiner Wartezeit kommt. Diese Bedingung impliziert ebenfalls das FIFO-Prinzip und keinen Puffer zwischen den Maschinen.

Recirculation (*rcrc*): Recirculation tritt im Job Shop bzw. flexible Job Shop Maschinenumfeld auf, wenn ein Job eine Station bzw. eine Maschine öfter als ein Mal aufsuchen kann.

Spezifikationen des γ-Feldes

Folgende zu optimierende Kriterien werden üblicherweise im γ- Feld angegeben:

Makespan (C_{max}): C_{max} ist definiert als $max(C_1, ..., C_n)$, wobei C_j die Fertigstellungszeit (completion time) der einzelnen Jobs darstellt. C_{max} ist somit die Fertigstellungszeit des letzten Jobs, der das System verlässt.

Maximum Lateness (L_{max}): In diesem Fall gilt es, die maximale Verspätung $max(L_1, ..., L_n)$ zu reduzieren. Die Verspätung $L_j = C_j - d_j$ von einem Job j wird durch die Abweichung des geplanten Liefertermins (due date d_j) definiert und kann somit auch negativ sein, wenn der Job vor dem geplanten Liefertermin fertiggestellt wird (negative Verspätung).

Total weighted completion time ($\sum \omega_j C_j$): Die Summe der gewichteten Fertigstellungszeiten $\sum_{j=1}^{n} \omega_j C_j$ der n Jobs ist oft ein Indiz für die gesamten Bestandskosten innerhalb des Systems. Für alle $\omega_j = 1$ sind die einzelnen Jobs nicht gewichtet und es wird somit nur die Summe der Fertigstellungszeiten $\sum_{j=1}^{n} C_j$ optimiert. Die Summe der Fertigstellungszeiten wird in der Literatur oft als *flow time* und die gewichtete Summe der Fertigstellungszeiten als *weighted flow time* bezeichnet.

Total weighted Tardiness ($\sum \omega_j T_j$): Diese Spezifikation optimiert die Summe aller gewichteten (positiven) Verspätungen $\sum_{j=1}^{n} \omega_j T_j$ der n Jobs und ist definiert als $T_j = max(C_j - d_j, 0)$ wodurch im Vergleich zu L_j, T_j nicht negativ sein kann. Haben alle Jobs die gleiche Priorität ($\omega_j = 1$ für alle j), wird nur die Summer der Verspätungen $\sum_{j=1}^{n} T_j$ optimiert.

Weighted number of tardy Jobs ($\sum \omega_j U_j$): Hier wird nicht die Dauer der einzelnen Verspätungen betrachtet, sondern nur die Anzahl der Verspätungen. Ob ein Job eine

Verspätung von einer Woche oder einem Monat hat, spielt somit in diesem Fall keine Rolle. Haben alle Jobs die gleiche Priorität ($\omega_j = 1$ für alle j), wird nur die Summer der Anzahl an Verspätungen $\sum_{j=1}^{n} U_j$ optimiert. U_j wird als *unit penalty* bezeichnet und ist folgendermaßen definiert:

$$U_j = \begin{cases} 1 & wenn\, C_j > d_j \\ 0 & sonst \end{cases}$$

Total weighted Earliness ($\sum \omega_j E_j$): In diesem Fall wird nur die negative Verspätung betrachtet, also alle Jobs mit Fertigstellungszeiten vor dem geplanten Liefertermin. Die negative Verspätung ist definiert als $E_j = max(d_j - C_j, 0)$. Es ist in diesem Fall natürlich wieder möglich, nur die Anzahl der negativen Verspätungen zu betrachten.

Total setup/changeover cost[172] (TSC): Diese Spezifikation hat als Ziel die Reduktion der gesamten Rüstkosten aller Maschinen. Werden die Rüstzeiten optimiert, wird TST (*Total setup time*) im γ-Feld angegeben.

Konkrete Beispiele des Frameworks

$FJ3 \mid r_j, s_{ijk}, rcrc \mid \sum \omega_j T_j$: Dieses Beispiel beschreibt einen *flexible job shop* mit drei Stationen ($FJ3$). Die Jobs haben verschiedene release dates (r_j). Die Rüstzeiten sind von der Reihenfolge der Jobs und den jeweiligen Maschinen abhängig (s_{ijk}), wobei es möglich ist, dass ein Job eine Station mehrmals durchläuft ($rcrc$). Ziel ist es, die Summe aller gewichteten (positiven) Verspätungen zu minimieren ($\sum \omega_j T_j$).

$P4 \mid p_{ij} = p_j, prmp \mid \sum C_j$: Dieses Beispiel betrachtet eine Maschinenumgebung von vier identischen Maschinen, die parallel laufen ($P4$). Die Bearbeitungszeit eines Jobs j ist auf allen Maschinen gleich ($p_{ij} = p_j$) und es ist erlaubt einen Job zu unterbrechen und ihn zu einem späteren Zeitpunkt fortzusetzen ($prmp$). Das Ziel ist es, eine Reihenfolge aller n Jobs zu finden, die die Summe aller Fertigstellungszeiten minimiert ($\sum C_j$).

Das hier vorgestellte Framework deckt keines Falls alle möglichen Scheduling Modelle, die in der Praxis auftreten können. Es soll viel eher vermitteln, wie die verschiedenen Scheduling Modelle in der Literatur definiert werden. Denn nur durch eine einheitliche Definition der Modelle können Ergebnisse und Algorithmen für die verschiedenen Modelle verglichen bzw. weiter ergänzt werden.

8.2 Detaillierte Kostenbewertung der verschiedenen initialen Populationen

Um die Kosten der einzelnen initialen Populationen vergleichen zu können, werden von jeder initialen Population 1000 Individuen erstellt und mit der Zielfunktion aus Abschnitt

[172] vgl. Allahverdi, 2015, S.347

5.4 bewertet, wobei die Rüstmitarbeiter nicht berücksichtigt werden. Für jede initiale Population werden dann die minimalen, maximalen und Durchschnittskosten der Gesamtkosten sowie die der einzelnen Kostenarten berechnet (siehe respektive Abbildung 8.1, 8.3 und 8.2). In den anschließenden Diagrammen befinden sich die initialen Populationen auf der x-Achse und sind folgendermaßen durchnummeriert:

1 Initiale Population mit der zufälligen Einplanung einzelner Batches.

2 Initiale Population mit der zufälligen Einplanung ganzer Werkzeuge.

3 Initiale Population mit einer möglichst gleichmäßigen Aufteilung der Batches.

4 Initiale Population mit möglichst geringen Lagerkosten.

5 Initiale Population mit möglichst geringen Verspätungen.

6 Initiale Population mit möglichst geringen Rüstkosten.

7 Initiale Population mit möglichst geringen Betriebskosten.

8 Initiale Population ohne Werkzeugwartezeiten.

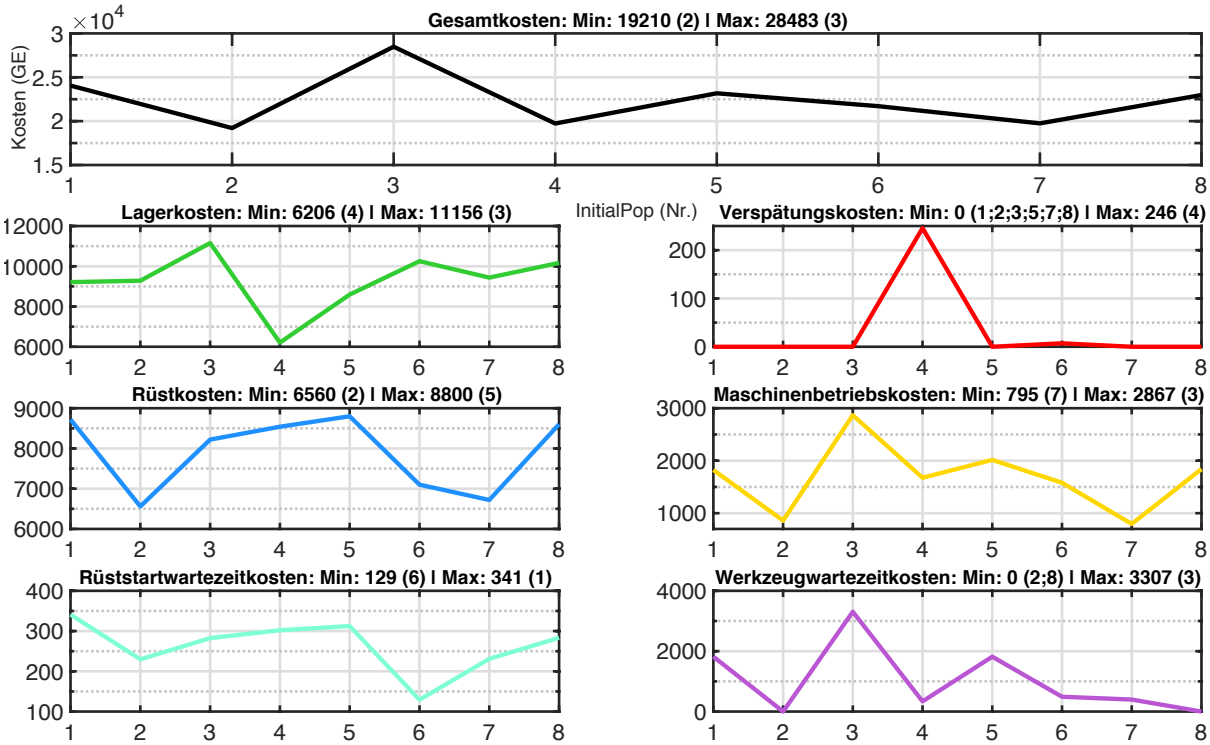

Abbildung 8.1: Minimale Kosten der initialen Populationen

Des Weiteren stehen für alle Kosten im Titel des jeweiligen Diagramms die minimalen und maximalen Werte. Zusätzlich befindet sich in Klammer die Nummer der initialen Population, die den minimalen bzw. maximalen Wert einnimmt. Für eine bessere Übersicht wurden nur die Achsen der Gesamtkosten beschriftet, wobei diese Achsenbeschriftung auch für die einzelnen Kosten gilt. Für eine bessere Lesbarkeit der maximalen und durchschnittlichen Verspätungskosten wurde in Abbildung 8.3 und 8.2 eine logarithmische Skala für

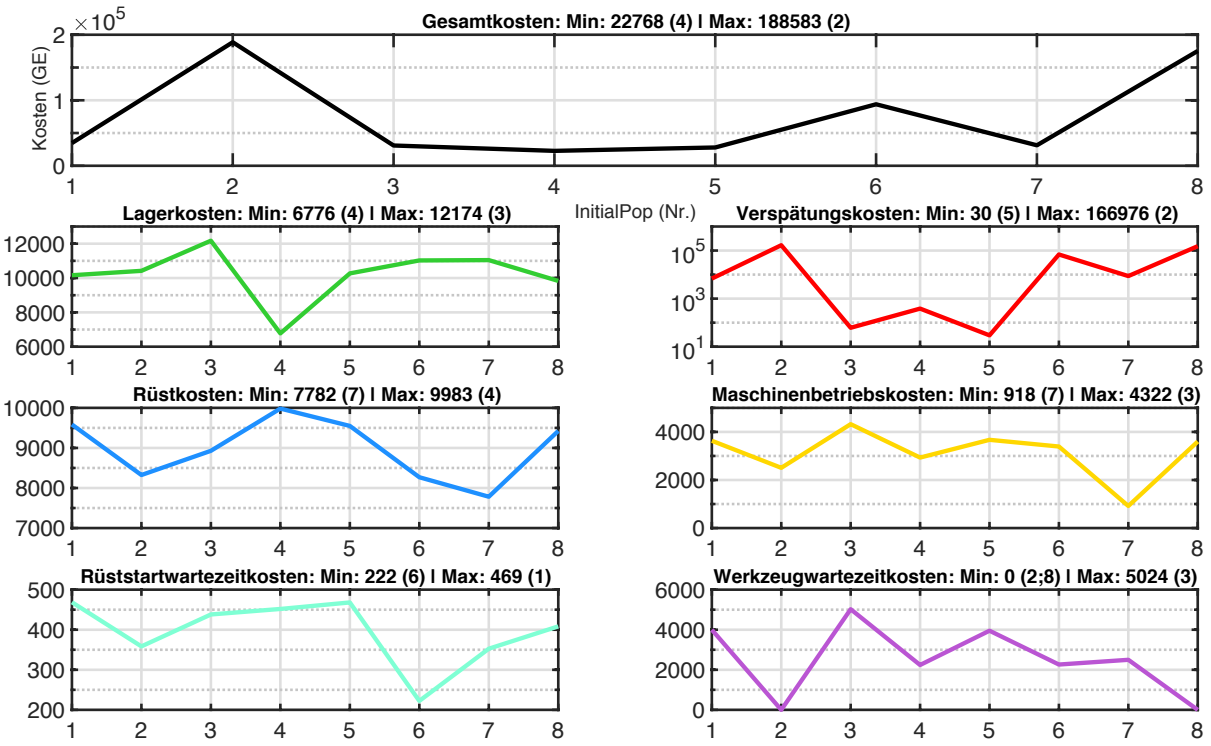

Abbildung 8.2: Durchschnittliche Kosten der initialen Populationen

die y-Achse benutzt. Zusätzlich wurden für die minimalen Lagerkosten in Abbildung 8.1 die Individuen ausgeschlossen, die eine Verspätung haben, da sonst Individuen mit einer großen Anzahl an Verspätungen die geringsten Lagerkosten erzielen würden. Gibt es keine Individuen ohne Verspätung (wie z.B. für die initiale Population Nr. 4), stehen nur die Individuen zur Auswahl, die eine minimale Verspätung haben.

(1) Die zufällige Aufteilung der einzelnen Batches liefert Individuen mit überdurchschnittlichen Kosten, was zu erwarten war, da keine Kosten gezielt bekämpft werden.

(2) Die zufällige Aufteilung von ganzen Werkzeugen liefert Individuen ohne Werkzeugwartezeiten, was in allen drei Abbildungen gut zu sehen ist. Des Weiteren sind geringere Rüstkosten zu erwarten, was ebenfalls in allen drei Abbildungen gut zu erkennen ist. Interessant ist, dass diese initiale Population sowohl das beste, als auch das schlechteste Individuum beinhaltet. Das Zustandekommen des schlechtesten Individuums lässt sich dadurch erklären, dass durch die geringe Anzahl an Werkzeugen (41) im Vergleich zur Anzahl an Batches (186), es viel wahrscheinlicher ist, dass durch eine zufällige Aufteilung der Werkzeuge die Maschinen überfüllt sind, was dazu führt, dass sehr hohe Verspätungskosten entstehen, die auch zu sehr hohen Gesamtkosten führen. Auf der anderen Seite ist es anscheinend durch die Anzahl der Werkzeuge auch mit einer annehmbaren Wahrscheinlichkeit möglich, eine gute Aufteilung zu erzielen, was dazu führt, dass sowohl geringe Betriebskosten als auch keine Verspätungen möglich sind. Kombiniert mit geringen

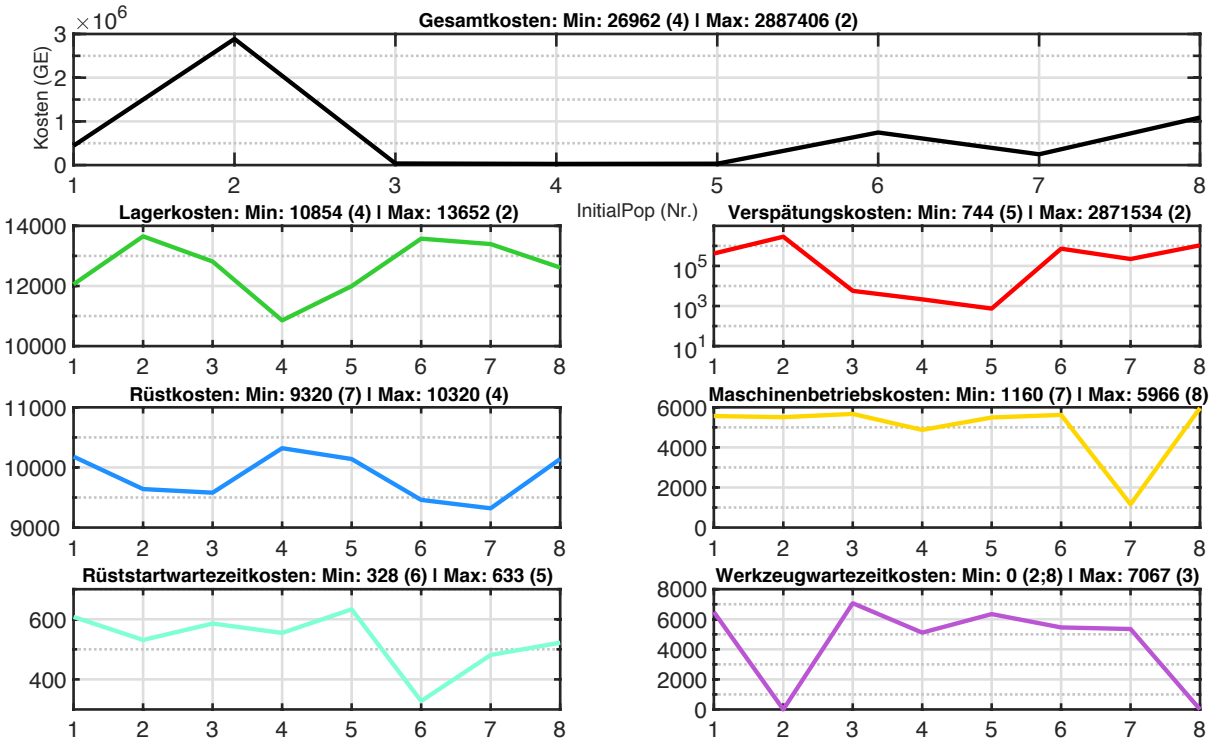

Abbildung 8.3: Maximale Kosten der initialen Populationen

Rüstkosten und keinen Werkzeugwartezeiten sind dementsprechend sehr gute Individuen möglich, wenn auch eher selten.

(3) Eine möglichst gleichmäßige Aufteilung der Batches führt zu Individuen mit sehr geringen Verspätungen, was dazu führt, dass die durchschnittlichen und maximalen Gesamtkosten relativ gut ausfallen. Jedoch weist diese initiale Population das Individuum mit den höchsten minimalen Gesamtkosten auf, was durch die relativ hohen restlichen Einzelkosten zu erklären ist. Die Lagerkosten sind sehr hoch, was vor allem auch durch die Kombination mit einem zu hohen Angebot an Maschinenkapazitäten im Vergleich zur Auftragslage entsteht (siehe auch letzter Absatz in Abschnitt 5.2.5). Die Werkzeugwartezeiten sind ebenfalls sehr hoch, da durch die gleichmäßige Aufteilung die Wahrscheinlichkeit relativ hoch ist, dass sich zwei Aufträge mit dem gleichen Werkzeug überschneiden. Zusätzlich werden durch die gleichmäßige Verteilung die Maschinen mit hohen Betriebskosten mehr oder weniger gleichviel belegt, wie Maschinen mit niedrigen Betriebskosten, was dazu führt, dass die Betriebskosten sehr hoch sind. Aus der Sicht der Gesamtkosten entstehen für diese initiale Population zwar keine sehr schlechten Individuen, jedoch ist es auch nicht möglich, sehr gute Individuen zu erstellen.

(4) Die initiale Population mit möglichst geringen Lagerkosten erfüllt ihren Zweck und erstellt Individuen mit sehr geringen Lagerkosten. Dadurch, dass die Batches auf den Maschinen eingeplant werden, auf denen die Startzeit möglichst nahe ans Lieferdatum

fällt, jedoch wenn möglich ohne Verspätung (siehe auch genaue Beschreibung dieser initialen Population in Abschnitt 5.6.1.3), fallen die Verspätungskosten auch relativ gering aus. Nichtsdestotrotz haben alle Individuen in dieser Population zumindest eine kleine Verspätung, da die minimalen Verspätungskosten ungleich Null sind. Auffällig ist, dass diese Initiale Population die höchsten durchschnittlichen und maximalen Rüstkosten aufweist.

(5) Die initiale Population mit möglichst geringen Verspätungen erfüllt ebenfalls ihren Zweck und weist die geringsten durchschnittlichen Verspätungskosten auf, wobei die restlichen Kosten im Vergleich zu den anderen Population eher überdurchschnittlich ausfallen. Im Endeffekt sind die restlichen Durchschnittskosten sehr ähnlich der initialen Population Nr. 1, da für die Batches nur selten eine Maschine mit einer Verspätung aus den möglichen Maschinen entfernt werden muss und somit die zufällige Maschinenauswahl nur relativ wenig beschränkt wird. Dementsprechend bleibt in den meisten Fällen die Maschinenauswahl identisch mit der von der initialen Population Nr. 1, bei der aus allen möglichen Maschinen zufällig eine ausgewählt wird.

(6) Die initiale Population mit möglichst geringen Rüstkosten wird zwar bei den durchschnittlichen Rüstkosten von der Population mit möglichst geringen Betriebskosten geschlagen, jedoch weist diese trotzdem sehr gute durchschnittliche Rüstkosten auf. Sehr eindeutig fallen hier die Rüststartwartezeitkosten aus, für die diese Population in allen drei Abbildungen am besten ausfällt. Dies liegt daran, dass, wenn für einen Batch auf allen möglichen Maschinen gerüstet werden muss, die Maschine ausgewählt wird, die die geringste Wartezeit auf den Rüststart hat (siehe auch genaue Beschreibung dieser initialen Population in Abschnitt 5.6.1.3). Auffällig sind bei dieser initialen Population die sehr hohen Verspätungskosten, da sowohl die durchschnittlichen also auch maximalen Verspätungskosten sehr hoch ausfallen und auch die minimalen Verspätungskosten ungleich Null sind, was bedeutet, dass es in dieser Population keine Individuen ohne Verspätung gibt. Die Lagerkosten fallen ebenfalls überdurchschnittlich aus.

(7) Die initiale Population mit möglichst geringen Betriebskosten erfüllt sehr gut ihren Zweck und weist in allen drei Abbildungen die besten Werte für die Betriebskosten auf, wobei dies am eindeutigsten für die maximalen Betriebskosten ausfällt. Auffällig sind hier ebenfalls die sehr guten, sprich sehr geringen Rüstkosten, die dadurch entstehen, dass die geringsten Betriebskosten auf der Maschine vorkommen, die den geringsten Stundensatz aufweist und auf der kein Rüstvorgang benötigt wird. Wieso jedoch die Rüstkosten sogar besser ausfallen, als für die initiale Population, die sich nur auf die Rüstkosten konzentriert, ist nicht ganz nachvollziehbar. Die sehr hohen Verspätungskosten lassen sich durch die Überlastung der Maschinen mit geringem Stundensatz erklären.

(8) Die initiale Population ohne Werkzeugwartezeiten erfüllt ebenfalls ihren Zweck sehr gut bezogen auf die Werkzeugwartezeit, jedoch auch nicht mehr. Außer dass alle Individuen in dieser Population keine Werkzeugwartezeiten haben, fallen die restlichen Kosten relativ durchschnittlich aus, mit Ausnahme der Verspätungskosten, die sehr hoch ausfallen. Die erhöhten Verspätungskosten lassen sich dadurch erklären, dass oft nur die Maschine keine Werkzeugwartezeit aufweist, auf der das benötigte Werkzeug das letzte Mal benutzt wurde. Dadurch entstehen mit einer erhöhten Wahrscheinlichkeit überlastete Maschinen. Nichtsdestotrotz schafft es diese initiale Population auch Individuen ohne Verspätungen zu erstellen.

8.3 Box-Whisker-Plot

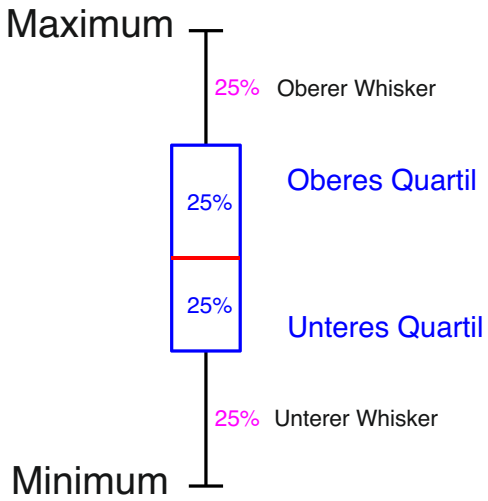

Abbildung 8.4: Erklärung des Box-Wisker-Plots

In einem Box-Plot oder auch Box-Wisker-Plot genannt, wird der Median, das untere und obere Quartil und der Minimal- und Maximalwert einer Stichprobe dargestellt.[173] In Abbildung 8.4 ist die detaillierte Beschreibung des Box-Wisker-Plots zu sehen. Der Medianwert ist in Rot abgebildet. Dieser ist so definiert, dass sich 50% der Werte sowohl oberhalb als auch unterhalb befinden. Die blaue Box gibt den Bereich der mittleren 50% der Daten an, wobei diese durch den Median eben in das untere und obere Quartil geteilt wird. Im unteren und oberen Whisker befinden sich jeweils die restlichen 25% der Daten, wobei das obere Whisker durch das Maximum und das untere Whisker durch das Minimum begrenzt ist.

[173] vgl. Becker u. a., 2016, S.62

Literaturverzeichnis

Allahverdi, A. (2015). "The third comprehensive survey on scheduling problems with setup times/costs". In: *European Journal of Operational Research* 246.2, S. 345–378.

Arnold, D. u.a., Hrsg. (2008). *Handbuch Logistik*. 3. Auflage. Berlin: Springer.

Becker, T. u. a. (2016). *Stochastische Risikomodellierung und statistische Methoden: ein anwendungsorientiertes Lehrbuch für Aktuare*. 1. Auflage. Statistik und ihre Anwendungen. Berlin: Springer Spektrum.

Borenstein, Y. und A. Moraglio, Hrsg. (2014). *Theory and principled methods for the design of metaheuristics*. Natural computing series. Berlin: Springer.

Brunner, F. J. (2017). *Japanische Erfolgskonzepte: KAIZEN, KVP, Lean Production Management, Total Productive Maintenance Shopfloor Management, Toyota Production System, GD3 - Lean Development*. 4. Auflage. München: Hanser Verlag.

Bullinger, H.-J. u. a., Hrsg. (2009). *Handbuch Unternehmensorganisation: Strategien, Planung, Umsetzung*. 3. Auflage. Berlin Heidelberg: Springer.

Burke, E. K. und G. Kendall (2013). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Boston, MA: Springer US.

Cormen, T. H. u. a., Hrsg. (2004). *Algorithmen - eine Einführung*. München: Oldenbourg.

Delahaye, D. u. a. (2019). "Simulated Annealing: From Basics to Applications". In: *Handbook of Metaheuristics*. Hrsg. von M. Gendreau und J.-Y. Potvin. Bd. 272. Cham: Springer International Publishing, S. 1–35.

Dickmann, P., Hrsg. (2015). *Schlanker Materialfluss: mit Lean Production, Kanban und Innovationen*. 3. Auflage. VDI-Buch. Berlin Heidelberg: Springer Vieweg.

Dittes, F.-M. (2015). *Optimierung: wie man aus allem das Beste macht*. Technik im Fokus: Daten, Fakten, Hintergründe. Berlin: Springer Vieweg.

Domschke, W. (1993). *Produktionsplanung: ablauforganisatorische Aspekte*. Springer-Lehrbuch. Berlin: Springer.

Dorigo, M. und T. Stützle (2019). "Ant Colony Optimization: Overview and Recent Advances". In: *Handbook of Metaheuristics*. Hrsg. von M. Gendreau und J.-Y. Potvin. Bd. 272. Cham: Springer International Publishing, S. 311–351.

Erlach, K. (2010). *Wertstromdesign: der Weg zur schlanken Fabrik*. 2. Auflage. VDI-Buch. Berlin: Springer.

Falkenauer, E. (1998). *Genetic algorithms and grouping problems*. Chichester, New York: Wiley.

Feo, T. A. und M. G. C. Resende (1995). "Greedy Randomized Adaptive Search Procedures". In: *Journal of Global Optimization* 6.2, S. 109–133.

Fritzsche, A. (2009). *Heuristische Suche in komplexen Strukturen: zur Verwendung Genetischer Algorithmen bei der Auftragseinplanung in der Automobilindustrie*. 1. Auflage. Gabler Edition Wissenschaft Produktion und Logistik. Wiesbaden: Gabler.

Garey, M. R. und D. S. Johnson (1979). *Computers and intractability: a guide to the theory of NP-completeness*. A Series of books in the mathematical sciences. San Francisco: W. H. Freeman.

Gebhard, M. und H. Kuhn (2009). *Hierarchische Produktionsplanung bei Unsicherheit*. 1. Auflage. Gabler Edition Wissenschaft Produktion und Logistik. Wiesbaden: Gabler.

Gendreau, M. und J.-Y. Potvin (2019). "Tabu Search". In: *Handbook of Metaheuristics*. Hrsg. von M. Gendreau und J.-Y. Potvin. Bd. 272. Cham: Springer International Publishing, S. 37–55.

Glover, F. (2003). *Handbook of metaheuristics*. International series in operations research & management science. Boston, Mass.: Kluwer.

Goss, S. u. a. (1989). "Self-organized shortcuts in the Argentine ant". In: *Naturwissenschaften* 76.12, S. 579–581.

Grimaud, F. u. a. (2014). "Exponential Smoothing for Multi-Product Lot-Sizing With Heijunka and Varying Demand". In: *Management and Production Engineering Review* 5.2, S. 20–26.

Grinninger, J. (2012). *Schlanke Produktionssteuerung zur Stabilisierung von Auftragsfolgen in der Automobilproduktion*. München: Lehrstuhl für Fördertechnk Materialfluss Logistik.

Gudehus, T. (2012). *Dynamische Disposition: Strategien, Algorithmen und Werkzeuge zur optimalen Auftrags-, Bestands- und Fertigungsdisposition*. 3. Auflage. Berlin: Springer.

Hansen, P und N. Mladenović (2009). "Variable Neighborhood Search Methods". In: *Encyclopedia of Optimization*. Hrsg. von C. A. Floudas und P. M. Pardalos. Boston, MA: Springer US, S. 3975–3989.

Hansen, P. u. a. (2019). "Variable Neighborhood Search". In: *Handbook of Metaheuristics*. Hrsg. von M. Gendreau und J.-Y. Potvin. Bd. 272. Cham: Springer International Publishing, S. 57–97.

Heistermann, J. (1994). *Genetische Algorithmen: Theorie und Praxis evolutionärer Optimierung*. Teubner-Texte zur Informatik. Stuttgart: Teubner.

Hopp, W. J. und M. L. Spearman (2000). *Factory physics: foundations of manufacturing management*. 2nd edition. Boston: Irwin/McGraw-Hill.

Hütter, S. (2008). "Simulation einer nivellierten Produktion in der Automobilzuliefererindustrie". In: *Hrsg. von: Rabe, M., Frauenhofer IRB Verlag* Advances in Simmulation for Production and Logistics Applications, S. 71–80.

Igel, C. (2014). "No Free Lunch Theorems: Limitations and Perspectives of Metaheuristics". In: *Theory and Principled Methods for the Design of Metaheuristics*. Hrsg. von Y. Borenstein und A. Moraglio. Berlin, Heidelberg: Springer, S. 1–23.

Klevers, T. (2013). *Wertstrom-Management: mehr Leistung und Flexibilität für Unternehmen*. Management. Frankfurt am Main: Campus-Verlag.

Lourenço, H. R. u. a. (2019). “Iterated Local Search: Framework and Applications”. In: *Handbook of Metaheuristics*. Hrsg. von M. Gendreau und J.-Y. Potvin. Bd. 272. Cham: Springer International Publishing, S. 129–168.

März, L. und W. Krug, Hrsg. (2011). *Simulation und Optimierung in Produktion und Logistik: praxisorientierter Leitfaden mit Fallbeispielen*. VDI-Buch 130. Heidelberg: Springer.

Mladenović, N. und P. Hansen (1997). “Variable neighborhood search”. In: *Computers & Operations Research* 24.11, S. 1097–1100.

Moscato, P. und C. Cotta (2019). “An Accelerated Introduction to Memetic Algorithms”. In: *Handbook of Metaheuristics*. Hrsg. von M. Gendreau und J.-Y. Potvin. Bd. 272. Cham: Springer International Publishing, S. 275–309.

Müller-Merbach, H. (1973). *Operations Research: Methoden und Modelle der Optimalplanung*. 3. Auflage. Vahlens Handbücher der Wirtschafts- und Sozialwissenschaften. München: Vahlen.

Mutingi, M. und C. Mbohwa (2017). “Grouping Genetic Algorithms: Advances for Real-World Grouping Problems”. In: *Grouping Genetic Algorithms*. Bd. 666. Cham: Springer International Publishing, S. 45–66.

Ohno, T. (1988). *Toyota production system: beyond large-scale production*. Cambridge, MA: Productivity Press.

Osman, I. H. und G. Laporte (1996). “Metaheuristics: A bibliography”. In: *Annals of Operations Research* 63.5, S. 511–623.

Pearl, J. (1985). *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley series in artificial intelligence. Boston, Mass.: Addison-Wesley.

Pinedo, M. (2016). *Scheduling: theory, algorithms, and systems*. Fifth Edition. Cham: Springer International Publishing.

Pohl, K. (2002). *Produktionsmanagement mit SAP R/3*. Berlin: Springer.

Pomberger, G. und H. Dobler (2008). *Algorithmen und Datenstrukturen: eine systematische Einführung in die Programmierung*. It-informatik. München: Pearson Studium.

Rager, M. (2008). *Energieorientierte Produktionsplanung: Analyse, Konzeption und Umsetzung*. Wiesbaden: Gabler Verlag.

Rayward-Smith, V. u. a. (1995). “Modern Heuristic Search Methods”. In: *Operational Research Tutorial Papers 1995*, S. 122.

Redlich, T. und J. P. Wulfsberg (2011). *Wertschöpfung in der Bottom-up-Ökonomie*. VDI-Buch. Berlin: Springer.

Reeves, C. (1996). “Heuristic Search Methods: A Review”. In: *Operational Research: KeynotePapers 1996*. Operational Research Society, S. 122–149.

Resende, M. G. C. und C. C. Ribeiro (2019). “Greedy Randomized Adaptive Search Procedures: Advances and Extensions”. In: *Handbook of Metaheuristics*. Hrsg. von M. Gendreau und J.-Y. Potvin. Bd. 272. Cham: Springer International Publishing, S. 169–220.

Reusch, P. (2006). "Abstimmungsmechanismen zwischen Programmplanung und Mengenplanung in der mehrstufigen Produktionsplanung". Diss. Duisburg-Essen: Universität Duisburg-Essen.

Rimscha, M. (2010). *Algorithmen kompakt und verständlich: Lösungsstrategien am Computer*. 2. Auflage. Programmiersprachen, Datenbanken und Softwareentwicklung. Wiesbaden: Vieweg + Teubner.

Robert Jacobs, F. und F.C. Ted Weston (2007). "Enterprise resource planning (ERP): A brief history". In: *Journal of Operations Management* 25.2, S. 357–363.

Sarin, S. C. u. a. (2010). *Stochastic Scheduling: Expectation-Variance Analysis of a Schedule*. Cambridge: Cambridge University Press.

Scharnbacher, K. (1992). *Statistik im Betrieb Lehrbuch mit Praktischen Beispielen*. Wiesbaden: Gabler Verlag.

Slack, N. u. a. (2005). *Operations management*. 4. edition. Harlow: Financial Times Prentice Hall.

Sörensen, K. (2015). "Metaheuristics-the metaphor exposed". In: *International Transactions in Operational Research* 22.1, S. 3–18.

Stadtler, H. (2005). "Supply chain management and advanced planning: basics, overview and challenges". In: *European Journal of Operational Research* 163.3, S. 575–588.

Sterman, J. D. (1989). "Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamic Decision Making Experiment". In: *Management Science* 35.3, S. 321–339.

Sujata, D. u. a. (2017). *Handbook of Research on Modeling, Analysis, and Application of Nature-Inspired Metaheuristic Algorithms*. Hershey, US: IGI Global.

Tautrim, J. (2014). *Lean Administration: Taschenbuch/Beraterleitfaden: Wesentliche Konzepte und Werkzeuge für mehr Effizienz in der Verwaltung*. Berlin: epubli.

Tavakkoli-Moghaddam, R. u. a. (2009). "Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints". In: *Computers & Operations Research* 36.12, S. 3224–3230.

Tegel, A. (2013). *Analyse und Optimierung der Produktionsglättung für Mehrprodukt-Fließlinien: Eine Studie zum Lean-Production-Konzept*. Wiesbaden: Springer-Gabler Verlag.

Thonemann, U. und M. Albers (2010). *Operations Management: Konzepte, Methoden und Anwendungen*. 2. Auflage. Wi - Wirtschaft. München: Pearson Studium.

Toksari, M. D. (2016). "A hybrid algorithm of Ant Colony Optimization (ACO) and Iterated Local Search (ILS) for estimating electricity domestic consumption: Case of Turkey". In: *International Journal of Electrical Power & Energy Systems* 78, S. 776–782.

Wang, X.-J. u. a. (2008). "A Survey and Future Trend of Study on Multi-Objective Scheduling". In: *2008 Fourth International Conference on Natural Computation*. Jinan, Shandong, China: IEEE, S. 382–391.

Wannenwetsch, H. (2014). *Integrierte Materialwirtschaft, Logistik und Beschaffung*. 5. Auflage. Berlin: Springer Vieweg.

Whitley, D. (2019). "Next Generation Genetic Algorithms: A User's Guide and Tutorial". In: *Handbook of Metaheuristics*. Hrsg. von M. Gendreau und J.-Y. Potvin. Bd. 272. Cham: Springer International Publishing, S. 245–274.

Whitley, D. und J. P. Watson (2005). "Complexity Theory and the No Free Lunch Theorem". In: *Search Methodologies*. Hrsg. von E. K. Burke und G. Kendall. Boston, MA: Springer US, S. 317–339.

Wiendahl, H.-H. (2011). *Auftragsmanagement der industriellen Produktion: Grundlagen, Konfiguration, Einführung*. VDI-Buch. Berlin: Springer.

Wilson, J. M. (2016). "The origin of material requirements planning in Frederick W. Taylor's planning office". In: *International Journal of Production Research* 54.5, S. 1535–1553.

Wolpert, D.H. und W.G. Macready (2005). "Coevolutionary Free Lunches". In: *IEEE Transactions on Evolutionary Computation* 9.6, S. 721–735.

Womack, J. P. (1992). *Die zweite Revolution in der Autoindustrie: Konsequenzen aus der weltweiten Studie aus dem Massachusetts Institute of Technology*. 6. Auflage. Frankfurt am Main: Campus-Verl.

Womack, J. P. und D. T. Jones (2003). *Lean thinking: banish waste and create wealth in your corporation*. 1st Free Press edition. New York: Free Press.

Yang, X.-S. (2010). *Engineering Optimization: An Introduction with Metaheuristic Applications*. Hoboken, NJ, USA: John Wiley & Sons.

Yavuz, M. und E. Akçali (2007). "Production smoothing in just-in-time manufacturing systems: a review of the models and solution approaches". In: *International Journal of Production Research* 45.16, S. 3579–3597.

Zäpfel, G. und R. Braune (2005). *Moderne Heuristiken der Produktionsplanung: am Beispiel der Maschinenbelegung*. WiSo-Kurzlehrbücher Reihe Betriebswirtschaft. München: Vahlen.

Zheng, S. u. a. (2000). "The New Spectrum of the Cross-Enterprise Solution: The Integration of Supply Chain Management and Enterprise Resources Planning Systems". In: *Journal of Computer Information Systems* 41.1, S. 84–93.