



# Losgrößenoptimierung in der mechanischen Bauteilfertigung

## Ein heuristischer Ansatz

### DIPLOMARBEIT

zur Erlangung des akademischen Grades

### Diplom-Ingenieur

im Rahmen des Studiums

### Wirtschaftsingenieurwesen und Maschinenbau

eingereicht von

**Thomas Weiler**

Matrikelnummer 0326460

an der Fakultät für Maschinenwesen und Betriebswissenschaften  
am Institut für Fertigungstechnik und Hochleistungslasertechnik E311  
der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Burkhard Kittl

Wien, 30. September 2015

Thomas Weiler

Burkhard Kittl



# Erklärung zur Verfassung der Arbeit

Thomas Weiler  
Donaufelderstraße 54, 1210 Wien

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 an 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, 30. September 2015

---

Thomas Weiler



# Danksagung

In erster Linie möchte ich mich bei meinem Betreuer Prof. Burkhard Kittl bedanken. Von Beginn der Arbeit bis zur Fertigstellung dieses Dokuments haben mich die Gespräche mit ihm nicht nur fachlich inspiriert, sondern auch meine persönliche Entwicklung gefördert. Herzlichen Dank für die individuelle, ausgiebige und vor allem geduldige Unterstützung.

Ein besonderes Dankeschön gilt meiner Schwester Eva, die diese Arbeit penibel Korrektur gelesen hat und mir dadurch so manche Peinlichkeit ersparte.

Natürlich möchte ich mich auch bei meinen Freunden bedanken, welche mich fortwährend unterstützt haben und mir verständnisvoll zur Seite gestanden sind.



# Kurzfassung

Im Verlauf der industriellen Revolution wurde die Fertigung in Handwerksbetrieben und Manufakturen immer mehr durch mechanisierte Produktionsprozesse ersetzt. Beim Wechsel der Produktion auf ein anderes Erzeugnis mussten diese immer komplexer werdenden Anlagen nun für das jeweilige Produkt gerüstet werden. Aus dieser Entwicklung heraus entstand die Fragestellung nach der idealen Losgröße, welche auch heute in der mechanischen Bauteilfertigung von zentralem Interesse ist.

Diese Arbeit hat sich das Ziel gesetzt, eine heuristische Herangehensweise an die Losgrößenproblematik zu finden.

Anfangs werden überblicksartig die gängigen Verfahren beschrieben und teilweise auch deren Einschränkungen behandelt. In einem nächsten Schritt wird ein alternativer Lösungsansatz entwickelt und die hierfür notwendigen Algorithmen vorgestellt. Im weiteren Verlauf der Arbeit wird die Implementierung der Heuristik in einen Software-Prototypen geschildert und dessen Funktionalitäten und potentielle Einsatzszenarien diskutiert.

Die im Zuge dieser Arbeit entwickelte Software, lässt sich auch als Losgrößen-Automatismus charakterisieren.





# Abstract

Over the course of the Industrial Revolution the production in craft shops was mostly replaced by mechanized production processes. These increasingly complex systems had to be set up for each product and therefore changing the production to a different product became more and more difficult. This development resulted in the search for the ideal lot size. A problem which is still of central interest in the mechanical component manufacturing industry.

In this work a heuristic approach, for the task of finding the ideal lot size, was created.

The most common methods and their limitations are described at the beginning of this document. An alternative approach is developed and its necessary algorithms are discussed in the next step. In the further course of the work, the implementation of the heuristic is explained. Finally the resulting software prototype with its capabilities, limitations and potential utilization scenarios, is described.

The developed software can be characterized as a lot-size-automatism.



# Inhaltsverzeichnis

<b>Kurzfassung</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>Inhaltsverzeichnis</b>	<b>xi</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Problemstellung . . . . .	2
1.2 Motivation . . . . .	3
1.3 Zielsetzung . . . . .	5
1.4 Anforderungen an einen Losgrößen-Automatismus . . . . .	6
1.5 Definitionen . . . . .	7
<b>2 Grundlagen</b>	<b>11</b>
2.1 Gängige Verfahren . . . . .	11
2.2 Andler – Harris . . . . .	13
2.3 Input und Output . . . . .	15
2.4 Rechenleistung . . . . .	15
2.5 Algorithmen . . . . .	16
2.5.1 Brute-Force . . . . .	16
2.5.2 Rucksackproblem . . . . .	17
2.5.3 Problem des Handlungsreisenden . . . . .	17
2.5.4 Nearest Neighbor . . . . .	17
<b>3 Lösungsansatz</b>	<b>19</b>
3.1 Stufe 1 – Trial and Error . . . . .	19
3.2 Stufe 2 – Schrittweise Näherung . . . . .	22
<b>4 Umsetzung</b>	<b>29</b>
4.1 Struktur und Technologien . . . . .	29
4.2 Stufe 1 – Brute Force . . . . .	32
4.2.1 Input Stufe 1 . . . . .	35
4.2.2 Strategien und Parameter . . . . .	38
4.2.3 Implementierung . . . . .	39
4.2.4 Output Stufe 1 . . . . .	43
4.2.5 Performance . . . . .	45
4.2.6 IBF (Intelligent-Brute-Force) . . . . .	48
4.2.7 Optimierung im Vorfeld . . . . .	49
4.3 Stufe 2 – Nearest Neighbor . . . . .	50

4.3.1	Input Stufe 2 . . . . .	51
4.3.2	Analogien . . . . .	55
4.3.3	Implementierung Nearest Neighbor . . . . .	56
4.3.4	Output Stufe 2 . . . . .	57
4.3.5	Performance . . . . .	58
4.4	GUI . . . . .	59
4.4.1	Hauptmenü . . . . .	59
4.4.2	Input . . . . .	61
4.4.3	Prognose . . . . .	63
4.4.4	Daten . . . . .	66
4.4.5	IBF (Stufe 1) . . . . .	67
4.4.6	Nearest-Neighbor (Stufe 2) . . . . .	69
4.4.7	Auswertung . . . . .	70
4.4.8	Export . . . . .	72
4.4.9	Einstellungen . . . . .	72
<b>5</b>	<b>Ergebnisse</b>	<b>75</b>
5.1	Anwendungsszenario . . . . .	76
5.1.1	Kontinuierliche Berechnung . . . . .	76
5.1.2	Analyse/Evaluierung . . . . .	77
5.1.3	Szenarien und deren Simulation . . . . .	78
5.2	Problemfelder . . . . .	78
5.2.1	Prognose . . . . .	79
5.2.2	Anbindung – Datenintegrität . . . . .	79
5.2.3	Exponentielles Wachstum . . . . .	80
5.2.4	Ausnahmefälle . . . . .	81
5.3	Entwicklungspotentiale . . . . .	81
5.3.1	Parallelisierung in Stufe 1 . . . . .	81
5.3.2	Alternative Algorithmik für Stufe 2 . . . . .	82
5.3.3	Systemparameter Periodendauer . . . . .	82
<b>6</b>	<b>Fazit</b>	<b>85</b>
	<b>Abkürzungen</b>	<b>87</b>
	<b>Abbildungsverzeichnis</b>	<b>88</b>
	<b>Tabellenverzeichnis</b>	<b>89</b>
	<b>Literaturverzeichnis</b>	<b>91</b>

# Einleitung

Die Mechanisierung von Handarbeit leitete Mitte des 18. Jahrhunderts den Beginn des Maschinenzeitalters ein. Das Textilgewerbe übernahm hierbei eine Vorreiterrolle. Durch die Verwendung von mechanischen Webstühlen und Spinnmaschinen konnten ungeahnte Produktivitätssteigerungen erreicht werden.

Im weiteren Verlauf dieser Entwicklung wurde die Fertigung von Erzeugnissen in Handwerksbetrieben und Manufakturen immer mehr durch einen industriellen Produktionsprozess ersetzt.

Durch die Verwendung von stetig komplexer werdenden Produktionsmaschinen wurde der Wechsel zwischen den unterschiedlichen Erzeugnissen erschwert. Im Gegensatz zum Menschen, welcher als Handwerker sehr flexibel arbeiten kann, entstehen bei Produktionsmaschinen zusätzliche Aufwände beim Wechsel von einem Erzeugnis zum nächsten. Dieser Prozess wird allgemein als *Rüsten* einer Produktionsanlage bezeichnet. Die hierbei entstehenden Aufwände bzw. Kosten stellen den Anreiz dar, den Rüstprozess detailliert zu betrachten und nach möglichen Optimierungspotentialen zu suchen. Der Wechsel zwischen unterschiedlichen Teilen bzw. Erzeugnissen, die auf einer Produktionsressource hergestellt werden, ist seit der industriellen Revolution Gegenstand intensiver Untersuchungen <sup>1</sup>.

Losfertigung ist ein Begriff, welcher durch den Wandel während der industriellen Revolution zunehmend an Bedeutung gewann. Hierbei werden Produktionsaufträge

---

<sup>1</sup>Siehe Kapitel 2.2.

als geschlossene Einheiten sequentiell abgewickelt. Der definierende Parameter ist die Losgröße, die Anzahl der zu fertigenden gleichartigen Teile pro Auftrag.

Spätestens im Jahre 1913 existiert ein theoretischer Ansatz von Ford W. Harris welcher sich mit der Optimierung der Losgröße beschäftigt und später als Andler-Formel, speziell im deutschsprachigen Raum, bekannt wird.[2, S. 380]

Das Festlegen der Losgröße ist ein Problem, welches in jedem produzierenden Unternehmen gelöst werden muss.<sup>2</sup> Die Frage nach der optimalen Losgröße ergibt sich aus den Charakteristika der mechanisierten Bauteilfertigung und dem Bestreben nach bestmöglicher Wirtschaftlichkeit des gesamten Unternehmens.

## 1.1 Problemstellung

Die Losgröße ist einer der wichtigsten, wenn nicht überhaupt der wichtigste Parameter in Hinblick auf die Stückkosten in der mechanischen Bauteilfertigung. Sie bestimmt welche Produktionstechnologien für die jeweiligen Teile in Frage kommen. Des Weiteren hat die Losgröße der zu produzierenden Teile einen wesentlichen Einfluss auf die Beschaffungsstrategie der Rohmaterialien und die Planung der zur Verfügung stehenden Produktionsressourcen. Aufgrund dieser Tatsache ist es von großem Interesse, die Losgröße für die jeweiligen Erzeugnisse möglichst optimal festzulegen. Dies bedeutet, dass die benötigte Menge an qualitativ entsprechenden Gütern, zum passenden Termin und zu minimalen Stückkosten bereitgestellt werden kann.

Abbildung 1.1 veranschaulicht, welche Konsequenzen eine Variation der Losgröße für andere Parameter haben kann.

---

<sup>2</sup>Die Frage nach der idealen Bestellmenge in einem Handelsunternehmen ist eng verwandt mit der nach der Losgröße. Die Problematik ist somit nicht beschränkt auf Produktionsunternehmen. Im weiteren Verlauf dieser Arbeit wird allerdings gemäß dem Titel nur auf produzierende Betriebe eingegangen.

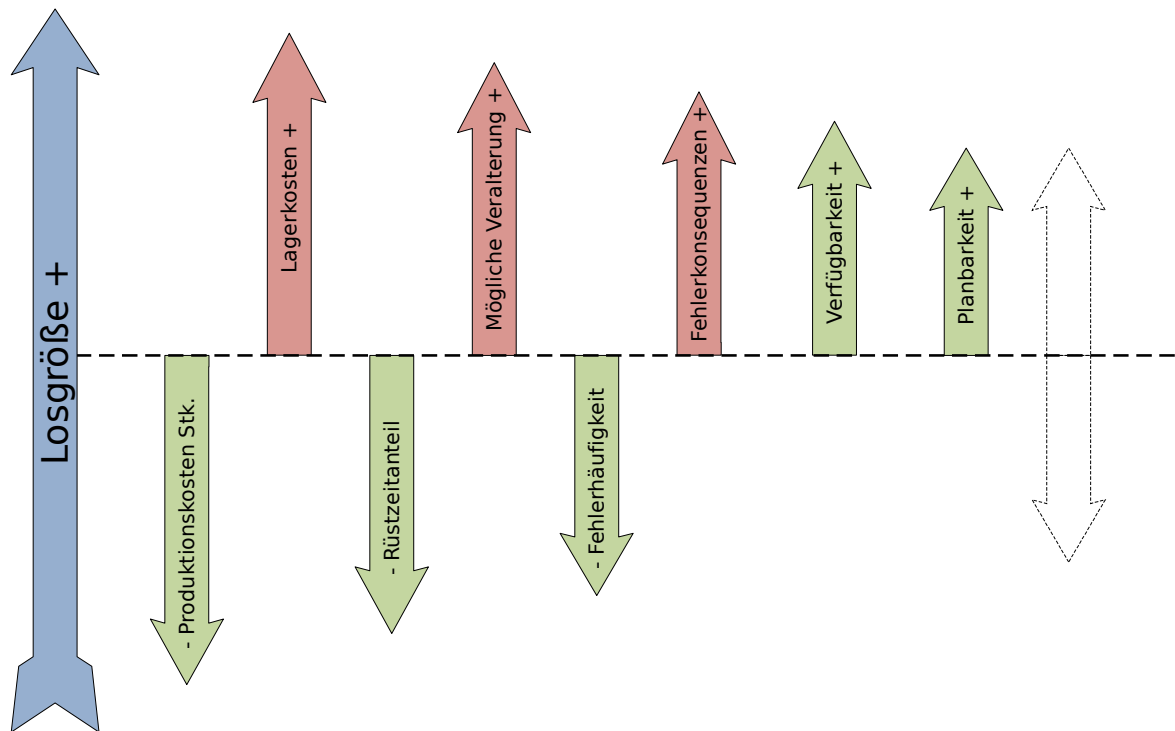


Abbildung 1.1: Möglicher Einfluss einer Losgrößenvariation auf andere Parameter

Im Wesentlichen ergibt sich bei der Optimierung der Losgröße ein Zielkonflikt bedingt durch das Wesen der Rüst- und Lagerkosten. Abbildung 1.2 zeigt dieses Spannungsfeld aus losgrößenvariablen Kosten und Fixkosten, um das zu Grunde liegende Optimierungsproblem anschaulich zu machen.

Die bekannten und bewährten Methoden, die zum Auffinden der Losgröße verwendet werden, sind in Kapitel 2 detaillierter aufgelistet und beschrieben. Vorwegnehmen lässt sich die Tatsache, dass die in Frage kommenden Verfahren und die zu erwartende Performance massiv von den jeweiligen Gegebenheiten (Vorhersehbarkeit der Bedarfe, gegenseitige Beeinflussungen in der Fertigung, ...) abhängen.

## 1.2 Motivation

Produktionsanlagen sind im Allgemeinen Investitionsgüter, welche über eine mittelfristige Nutzungsdauer (mehrere Jahre bis Jahrzehnte) verfügen. Die Abschreibungen für diese Investitionsgüter und deren benötigte Infrastruktur bilden den Großteil der Fixkosten dieser Anlagen bzw. Maschinen.

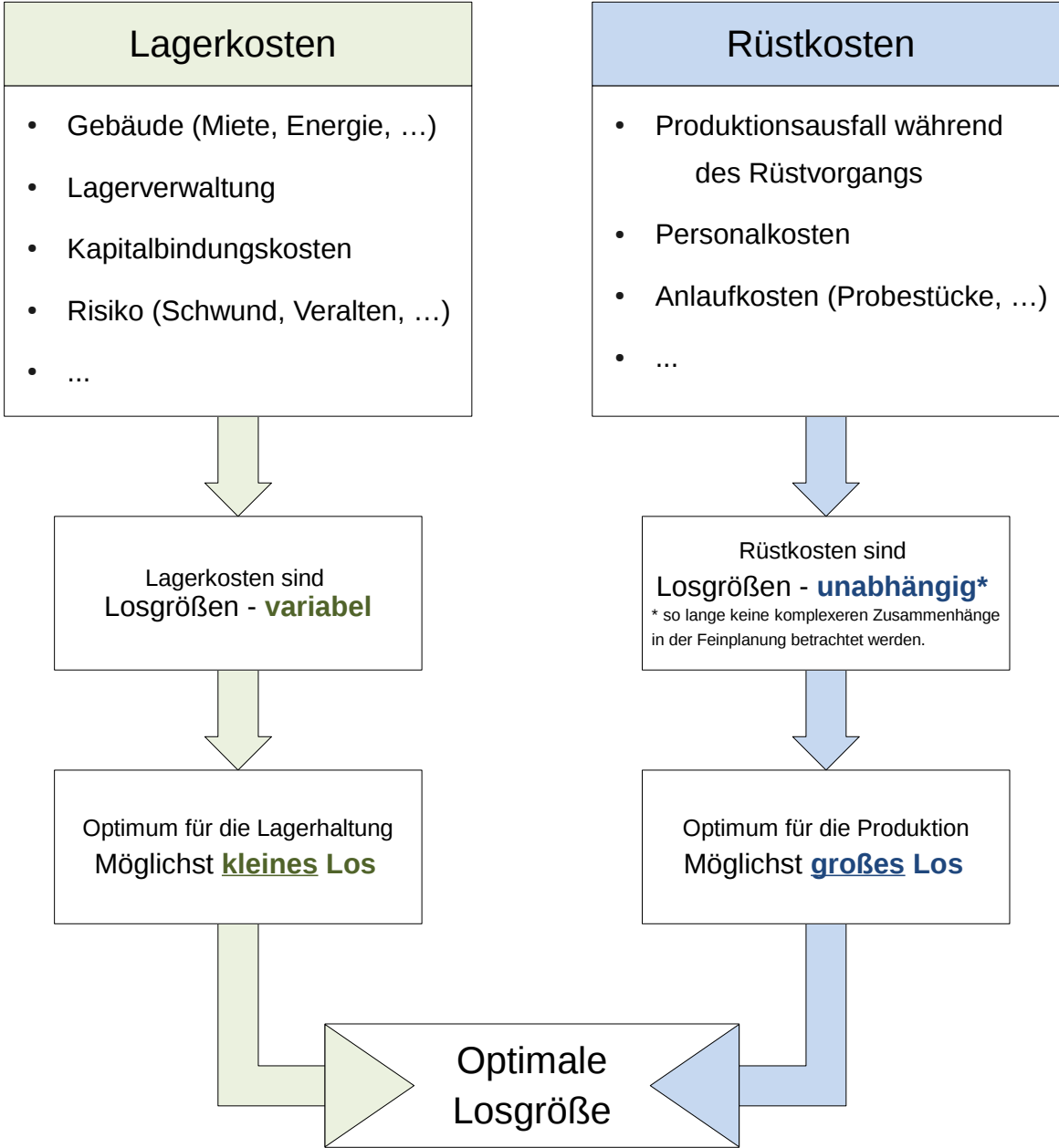


Abbildung 1.2: Zielkonflikt zwischen Lagerkosten und Rüstkosten beim Festlegen der Losgröße



Während der Nutzungsdauer von Produktionsanlagen lassen sich die Fixkosten nur noch marginal beeinflussen, da sie primär aus den Anschaffungskosten resultieren<sup>3</sup>. Um möglichst kostenminimal zu arbeiten, rücken die variablen Kosten in den Mittelpunkt der Optimierungsbestrebungen. Diese werden in der mechanischen Bauteilfertigung am maßgeblichsten durch die Losgröße der einzelnen Fertigungsaufträge beeinflusst<sup>4</sup>. Denn die Losgröße ist der organisatorische Parameter in der mechanischen Bauteilfertigung, welcher letztendlich den größten Einfluss auf die Stückkosten ausübt.

Die Kenntnis der optimalen Losgröße für einen zu produzierenden Artikel ist somit ein erstrebenswertes Ziel. Allerdings muss sich der Aufwand, welcher durch die Ermittlung dieses Optimums entsteht, in einem ökonomisch angemessenen Rahmen bewegen.

Die Beschäftigung mit der Losgrößenproblematik führt zu der Frage nach einem effizient arbeitenden Mechanismus<sup>5</sup>, welcher eine qualitativ hochwertige Antwort auf die Frage nach der Losgröße für die Bedienung der jeweiligen Bedarfe gibt.

### 1.3 Zielsetzung

Im Rahmen dieser Arbeit soll ein Automatismus geschaffen werden, welcher die Disponenten bei der Festlegung der Losgrößen unterstützen kann. Diese Hilfestellung kann direkt erfolgen in Form einer vorgeschlagenen Losgröße für den jeweiligen Artikel, oder auch indirekt mittels optimierter Parameter für die bereits in der Unternehmenssoftware vorhandenen Module und deren Dispositionsstrategien. Dieser Mechanismus soll deshalb in die bestehenden Enterprise Resource Planning (ERP) bzw. Manufacturing Execution System (MES) Systeme integrierbar sein und deren Funktionalität bei der Festlegung der Losgröße erweitern. Die Leistungsfähigkeit aktueller Central Processing Units (CPUs) ermöglicht die Implementierung dieses Automatismus in Form von rechenintensiven Algorithmen bzw. Heuristiken. Die Zielgruppe für die beschriebene Software sind primär mittelständische bis große Unternehmen im allgemeinen Maschinenbau. Nachfolgende Auflistung beinhaltet die angestrebten Eigenschaften des erwünschten softwarebasierten

---

<sup>3</sup>Alternative Finanzierungsmodelle wie z. B. Leasing werden hier außer Acht gelassen.

<sup>4</sup>Diese Aussage setzt voraus, dass die Kosten für Arbeitszeit, Material, Werkzeuge und Energie nicht außerordentlichen Schwankungen unterliegen.

<sup>5</sup>Hierunter kann man sich einen Algorithmus, eine analytische Formel, ein spezielles Verfahren usw. vorstellen.

Automatismus zum Auffinden einer möglichst guten Antwort auf die Frage nach der Losgröße:

- Bedarfe werden in Form von diskreten Lagerzugriffen modelliert.
- Planungszeiträume können beliebig festgelegt werden.
- Sämtliche Kostensätze können beliebig variiert werden, auch während der Planungsperiode.
- Produktionskapazitäten und die gegenseitige Beeinflussung der unterschiedlichen Artikel werden berücksichtigt.
- Verschiedenste Dispositionstrategien können abgebildet und simuliert werden.
- Die Anbindung an bestehende Systeme (ERP, MES, ...) ist möglich.
- Die bereits verfügbaren Informationen in dieser bestehenden Unternehmenssoftware sind als Input-Daten ausreichend.

## 1.4 Anforderungen an einen Losgrößen-Automatismus

Ziel jeder Losgrößenoptimierung ist es, unter Zuhilfenahme gewisser Eingangsparameter, eine möglichst ideale Lösung für die betrachteten Artikel zu generieren. Bei mathematischen Nachbildungen realer Prozesse wird immer nur ein Teil der Realität modelliert. Es werden Vereinfachungen in Kauf genommen und nur gewisse Teilaspekte betrachtet. Dies geschieht zu Gunsten einer verringerten Modellkomplexität. Aus der Perspektive des Anwenders ist die Qualität eines Optimierungsverfahrens am Verhältnis zwischen Aufwand und Nutzen zu erkennen.

Um ein Modell (insbesondere im speziellen Fall der Losgrößenoptimierung) als praxistauglich zu bezeichnen, lassen sich folgende Anforderungen formulieren:

**Qualität:** Die Ergebnisse unter Zuhilfenahme dieses Modells müssen den Aufwand rechtfertigen. Sollte ein erfahrener Mitarbeiter mit angemessenem Zeitaufwand, ohne

dieses Hilfsmittel in der Lage sein, dieselben oder sogar bessere Ergebnisse zu erzielen, so bringt das Modell keinen praktischen Nutzen mit sich.

**Praktikabilität:** Modelle können nur mit bereits vorhandenen Informationen als Input arbeiten. Aus diesen wird eine Lösung für den gewünschten Zeitraum ermittelt. Da diese Eingangsinformationen aber Änderungen im Zeitverlauf unterworfen sind, verändern sich auch die Outputs aus dem Modell. Die Ergebnisse müssen hinreichend nahe an dem Verhalten des realen Systems liegen, sodass die Lösungen stetig genug für die Detailplanung in der Produktion sind.

**Ökonomie:** Ein praxistaugliches Modell darf nur einen gewissen Aufwand erfordern, um sich selbst wirtschaftlich rechtfertigen zu können. Sowohl die benötigte Informationsbeschaffung als auch der Rechenaufwand dürfen die wirtschaftlichen Vorteile aus der Verwendung des Modells nicht zur Gänze konsumieren.

Im Rahmen dieser Arbeit wird ein Softwarepaket erstellt, welches die soeben beschriebenen Ziele zumindest konzeptionell erreichbar machen soll.

## 1.5 Definitionen

Im weiteren Verlauf dieser Arbeit wird die Kenntnis folgender Begriffe bzw. Abkürzungen vorausgesetzt. Die Berücksichtigung der nachfolgenden Definitionen ist besonders relevant für Begriffe, die situationsabhängig unterschiedliche Bedeutungen haben können.

**C++** ist eine von der ISO genormte Programmiersprache, die sowohl eine maschinen-nahe Programmierung als auch eine Programmierung auf einem hohen Abstraktionsniveau erlaubt.

**CPU** die Central Processing Unit ist jene Einheit in einem Computer, welche Anweisungen eines Programms durch grundlegende arithmetische, logische und Ein-/Ausgabeoperationen ausführt. Sie wird umgangssprachlich meistens einfach nur als Prozessor bezeichnet.

**CSS** Cascading Style Sheet ist eine deklarative Sprache, um Formateigenschaften von HTML-Objekten festzulegen.

**CSV** Comma-Separated Values bezeichnet ein Dateiformat und beschreibt den Aufbau einer Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten.

**ERP** steht für Enterprise Resource Planning. Unter ERP-System wird im Allgemeinen eine integrierte betriebswirtschaftliche Standardsoftware verstanden. Mit ihr lassen sich betriebswirtschaftliche Aufgaben aus den verschiedenen Bereichen (Produktion, Logistik, ...) IT-gestützt bearbeiten [5, S. 2].

**GPU** die Graphics Processing Unit ist ein spezialisierter Prozessor, der hauptsächlich für die Berechnung von Bildschirmdarstellungen verwendet wird.

**GUI** das Graphical User Interface ist eine Software-Komponente, die dem Benutzer eines Computers die Interaktion mit der Maschine via grafischer Symbole erlaubt.

**HTML** Hypertext Markup Language ist eine Auszeichnungssprache um Inhalte wie Text, Bilder und Hyperlinks in Dokumenten zu gliedern. HTML-Dokumente werden von einem Webbrowser interpretiert und anschließend für den Benutzer dargestellt.

**ID** unter dem Begriff Identifikator wird in diesem Dokument ein eindeutiges Kennzeichen eines Datensatzes verstanden.

**LAMP** Linux, Apache, MySQL, PHP: Unter einem LAMP-Setup versteht man ein Bündel von Softwareprogrammen, welche im Verband dynamische Webseiten betreiben. Hauptbestandteile dieser Webplattform sind Linux als Betriebssystem, Apache als Webserver, MySQL als Datenbank und PHP als verwendete Programmiersprache [4, S. 204].

**Los** bezeichnet die Menge von Erzeugnissen, welche ohne Umrüsten bzw. Unterbrechungen an der jeweiligen Maschine gefertigt wird.

**Losgröße** ist die Stückzahl der Teile, welche ohne Unterbrechung bzw. Wechsel auf einer Produktionsressource gefertigt werden [7, S. 186].

**MES** Bei Manufacturing Execution Systems geht es darum, die Produktion mit einem integrierten Informations- und Kontrollinstrumentarium zu steuern, um vorgegebene Zielgrößen zu erreichen [1, S. 1 f.].

**PHP** ist eine Programmiersprache, welche hauptsächlich zur Erstellung von dynamischen Webseiten verwendet wird.

**SQL** Structured Query Language ist eine Datenbanksprache zur Definition von Datenstrukturen in relationalen Datenbanken sowie zum Bearbeiten (Einfügen, Verändern, Löschen) und Abfragen von darauf basierenden Datenbeständen.

**UNIX-Timestamp** ist eine einfache Zeitdefinition, bei der die Sekunden beginnend mit dem 1. Januar 1970 00:00 Uhr gezählt und in Form eines 32-Bit-Integer gespeichert werden.

**XML** Extensible Markup Language ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Textform.



# Grundlagen

Die optimale Losgröße ist seit den Umwälzungen durch die industrielle Revolution immer wieder Gegenstand intensiver Forschungstätigkeit. Dieses Kapitel stellt nur einen kurzen Überblick einiger etablierter Ansätze zur Verfügung. Des Weiteren werden einige Algorithmen vorgestellt, welche große Relevanz für den in Kapitel 3 vorgestellten Lösungsansatz aufweisen. Auch die technischen Grundlagen (verfügbare Rechenleistung) für diesen Ansatz werden betrachtet.

## 2.1 Gängige Verfahren

Es existieren verschiedenste Vorgehensweisen, um die optimale Losgröße für ein bestimmtes Teil zu ermitteln. Die nachfolgenden Tabellen [6] stellen einen Auszug der verwendeten Verfahren dar, beginnend mit dem bekanntesten – der Andler-Formel.

Bei all diesen Modellen werden die zur Verfügung stehenden Produktionskapazitäten nicht berücksichtigt. In einem Produktionsbetrieb wird meist mehr als ein Artikel gefertigt. Somit kommt es durch die Begrenztheit der zur Verfügung stehenden Produktionskapazitäten zu Situationen, in denen die errechneten optimalen Losgrößen (des gesamten Produktionsprogramms) nicht mit den daraus resultierenden Maschinenbelegungen und den tatsächlich verfügbaren Ressourcen vereinbar sind. Dies ist eine Problematik, die sich durch die Ermittlung der Losgröße aus einer isolierten Perspektive ergibt.

Deterministische Modelle		
	Andler bzw. Harris	Wagner Whitin
Bedarfsverlauf	deterministisch konstant	deterministisch variabel
Planungszeitraum	statisch	dynamisch
Lagerkostensatz	konstant	variabel
Fixkosten	konstant	variabel
Berücksichtigung der Kapazitäten	nein	nein

Abbildung 2.1: Deterministische Modelle

Stochastische Modelle		
	Arrow Harris - Marschak - Typ	Modell von Beckmann
Bedarfsverlauf	Stochastische Mengen	Zeitliche Folge und Menge stochastisch
Planungszeitraum	dynamisch	statisch
Lagerkostensatz	konstant	konstant
Fixkosten	konstant	konstant
Berücksichtigung der Kapazitäten	nein	nein

Abbildung 2.2: Stochastische Modelle



## 2.2 Andler – Harris

Das Hauptaugenmerk dieser Arbeit liegt auf einem heuristischen Automatismus, welcher zur Ermittlung der Losgröße einsetzbar sein soll. Trotzdem ist es sinnvoll, die klassische Losformel nach Harris bzw. Andler in diesem Kapitel detaillierter zu beschreiben. Dies unterstützt ein grundlegendes Verständnis für die Thematik an sich und auch für die Grenzen analytischer Modelle.

Abbildung 2.3 zeigt den Verlauf des Lagerstandes für das von Andler-Harris beschriebene Szenario.

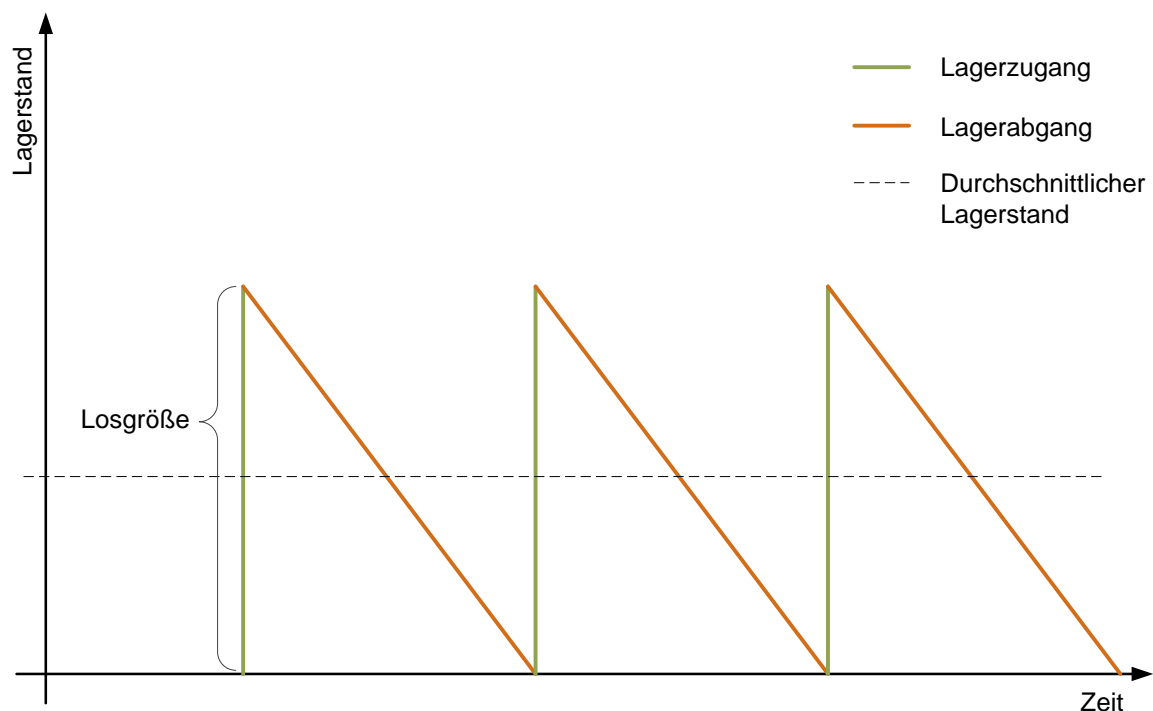


Abbildung 2.3: Verlauf des Lagerstandes bei Andler-Harris

Für die Berechnung der kostenminimalen Losgröße  $y_{Kmin}$  publizierte Andler 1929 folgenden Ansatz<sup>1</sup>:

Die Gesamtkosten  $K(y)$  exklusive der stückzahlvariablen Produktionskosten (Material, ...) errechnen sich nach Formel 2.1 wie folgt:

$$K(y) = \frac{R}{y} * C_R + y * \frac{C_L}{2} * T \quad (2.1)$$

<sup>1</sup>Vgl. <http://de.wikipedia.org/wiki/Andler-Formel>

Parameter der Andler-Formel	
$y$	Losgröße
$T$	Periodendauer
$V$	Lagerabgangsrate
$R$	Maximale Absatzmenge (Periodenbedarf)
$C_R$	Rüstkosten
$C_L$	Lagerkosten

Tabelle 2.1: Benötigte Parameter für die Berechnung der optimalen Losgröße nach Andler

Um nun das Minimum dieser Funktion zu finden, wird nach  $y$  abgeleitet und null gesetzt.

$$K'(y) = \frac{-R}{y^2} * C_R + \frac{C_L}{2} * T = 0 \quad (2.2)$$

Für die kostenminimale Losgröße  $y_{Kmin}$  ergibt sich nun:

$$y_{Kmin} = \sqrt{\frac{2 * R * C_R}{C_L * T}} \quad (2.3)$$

Dieses Ergebnis, auch als klassische Losformel bezeichnet, fußt auf folgenden Annahmen:

- beliebige Lose produzierbar (Kapazität vorhanden)
- konstante Lagerkosten
- konstante Rüstkosten
- unbegrenzte Lagerkapazität
- es werden keine Fehlmengen toleriert
- konstanter Bedarf
- alle Parameter bleiben im Zeitverlauf *konstant* (Lagerabgang findet kontinuierlich statt)

## 2.3 Input und Output

Abbildung 2.4 stellt die für das entstehende Softwarepaket zur Lösung der Losgrößenproblematik in einem produzierenden Unternehmen relevanten Ein- und Ausgabedaten in sehr kompakter Form dar.

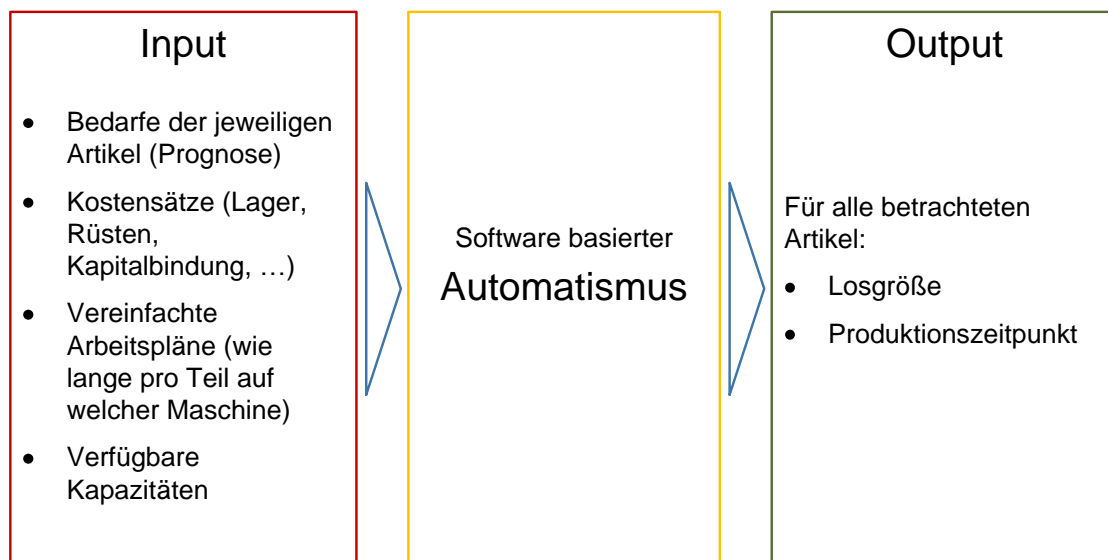


Abbildung 2.4: Grobdarstellung der Ein- und Ausgabedaten der angestrebten Software

Von besonderem Interesse ist an dieser Stelle die Akquise der in obiger Abbildung gelisteten Eingangsdaten. Das Vorhandensein eines ERP-Systems in einem mittelständischen Produktionsbetrieb lässt sich heute als Normalfall bezeichnen. Aus den Daten dieses Systems sollten alle benötigten Informationen extrahierbar sein. Möglicherweise kann auch auf ein MES zurückgegriffen werden.

Die genaue Vorgehensweise, bei der Datenextraktion aus den bestehenden Systemen, wird im Kapitel 4 beschrieben.

## 2.4 Rechenleistung

Um den im nächsten Kapitel beschriebenen Lösungsansatz im Kontext der heute verfügbaren Rechenleistung betrachten zu können, ist es sinnvoll, die Leistungsdaten eines aktuellen Standard-PC etwas detaillierter zu betrachten.

Als Beispiel soll die Hardware, welche während der Programmierung der nachfolgend beschriebenen Software verwendet wurde, dienen. Es handelt sich hierbei um einen AMD Phenom II X4 955 Prozessor mit vier Kernen und einer Taktfrequenz von 3,2 GHz. Des Weiteren verfügt das System über 4 GB Arbeitsspeicher wobei die CPU selbst auch über 4\*512 KB Level 2 Cache und über 6 MB Level 3 Cache verfügt. Laut diverser Benchmarks <sup>2</sup> erreicht die CPU ca. 40 GFLOPS (Giga Floating Point Operations per Second) =  $40 * 10^9$  Gleitkomma-Operationen pro Sekunde. Ausgehend von der vereinfachenden Annahme, dass wir nur 64-Bit-Datentypen verwenden, kann ein Gigabyte Arbeitsspeicher  $1 * 10^9 * 8/64 \sim 1,25 * 10^8$  Variablen fassen. In den 6 MB Level 3 Cache finden analog dazu ca.  $7 * 10^5$  Variablen Platz.

Diese Leistungsdaten ermöglichen einen Lösungsansatz für die Losgrößenproblematik, welcher auf die Verwendung rechenintensiver Algorithmen und Heuristiken zurückgreift.

## 2.5 Algorithmen

Im Zuge dieser Arbeit werden einige bekannte Problemstellungen aus der Informatik sowie dazugehörige Algorithmen angesprochen und auch implementiert. Die nachfolgende Auflistung umfasst nur Punkte, welche eine direkte Relevanz zur Problemstellung aufweisen. Viele andere Algorithmen, die z. B. zur Sortierung dienen, werden zwar in der erstellten Software verwendet (C++ Library<sup>3</sup>), aber hier nicht weiter beschrieben.

### 2.5.1 Brute-Force

Als Brute-Force wird eine Vorgehensweise bezeichnet, welche alle potentiellen Lösungen einer bestimmten Problemstellung durchprobiert, um das gesuchte Ergebnis zu liefern. Als Beispiel kann hier eine sehr plakative Anwendung aus der Kryptographie dienen:

Um ein unbekanntes Passwort bzw. einen Schlüssel ermitteln zu können, ist das Durchprobieren aller möglichen Kombinationen eine gängige Vorgehensweise. Dies wird als Brute-Force-Attacke bezeichnet.

---

<sup>2</sup>Vgl.: <http://www.cpubenchmark.net>, <http://www.techpowerup.com>, <http://www.cpu-world.com>

<sup>3</sup>Vgl.: <http://www.cplusplus.com/reference/algorithm/>

## 2.5.2 Rucksackproblem

Hierbei handelt es sich um ein Optimierungsproblem betreffend die Auswahl von unterschiedlichen Objekten mit ihrem jeweiligen Nutzwert und Gewicht. Es soll eine Teilmenge dieser Objekte gefunden werden, die den gesamten Wert maximiert, allerdings Restriktionen durch das aggregierte Gewicht berücksichtigt. Als fertigungstechnisches Beispiel fungiert folgendes Problem:

Spanplatten werden in einem definierten Format geliefert und sollen für die Produktion von Möbeln verwendet werden. Hierfür müssen aus dem Rohmaterial Zuschnitte mit unterschiedlichsten Abmessungen und Stückzahlen gefertigt werden. Das Ziel hierbei ist, eine ideale Ausnutzung des Plattenmaterials zu erreichen und folglich möglichst wenig Verschnitt tolerieren zu müssen.

Das Rucksackproblem tritt in zahlreichen Variationen auf und ist eine der häufigsten Optimierungsaufgaben in der Informatik [3].

## 2.5.3 Problem des Handlungsreisenden

Das Problem des Handlungsreisenden, auch Travelling Salesman Problem (TSP) genannt, ist ein kombinatorisches Optimierungsproblem. Es geht um die Planung einer Rundreise über definierte Zwischenstationen. Ziel hierbei ist eine möglichst geringe Gesamtweglänge zurückzulegen. Algorithmen, welche die bestmögliche Lösung liefern, verursachen einen mit der Anzahl der Stationen exponentiell ansteigenden Rechenaufwand<sup>4</sup>.

In der Praxis tritt diese Aufgabenstellung angefangen bei der Routenplanung in Logistikunternehmen bis hin zum Design von integrierten Schaltkreisen auf. Dementsprechend intensive Forschungstätigkeit auf diesem Gebiet führte zu einem breiten Spektrum an Heuristiken, welche sehr gute Lösungen liefern (Differenz von 1% zur Ideallösung)<sup>5</sup>.

## 2.5.4 Nearest Neighbor

Das Nearest-Neighbor (N-N)-Verfahren ist eine Heuristik, welche unter anderem zur Lösung des TSP verwendet wird.

---

<sup>4</sup>Dies ist formal noch nicht bewiesen, wird allerdings in Fachkreisen als sehr wahrscheinlich eingestuft.

<sup>5</sup>Vgl. [http://de.wikipedia.org/wiki/Problem\\_des\\_Handlungsreisenden](http://de.wikipedia.org/wiki/Problem_des_Handlungsreisenden)

Die Vorgehensweise hierbei ist relativ einfach: Ausgehend von dem aktuellen Knoten (Aufenthaltort beim TSP) wird der am geringsten gewichtete Pfad (kürzester Weg beim TSP) zu einem der anderen Knoten als nächster Schritt gewählt. Dies setzt sich solange fort, bis alle Knoten besucht wurden.

Ergebnisse welche auf diese Art und Weise ermittelt werden, können sehr weit von der Optimal-Lösung abweichen. Der Vorteil dieser Heuristik ist allerdings ein geringer Rechenaufwand und die einfache Implementierung<sup>6</sup>.

---

<sup>6</sup>Vgl. <https://de.wikipedia.org/wiki/Nearest-Neighbor-Heuristik>

## Lösungsansatz

Aufgrund der in den vorigen Kapiteln beschriebenen Erwartungen<sup>1</sup>, an den im Rahmen dieser Arbeit entstehenden Mechanismus zur Ermittlung der Losgröße, wird anstatt eines analytischen Ansatzes eine heuristische Vorgehensweise gewählt. Diese Heuristik ist im Wesentlichen zweistufig.

In Stufe 1 wird jeweils nur ein einziger Artikel betrachtet, für den die Optimallösung und mehrere möglichst gute Alternativlösungen der Losgröße ermittelt werden.

In der zweiten Stufe werden die vorhandenen Produktionskapazitäten berücksichtigt. Die in Stufe 1 generierten Lösungen werden so gereiht, dass Überbelegungen der Kapazitäten vermieden werden und die geplante Produktion der Artikel durchführbar wird.

### 3.1 Stufe 1 – Trial and Error

In der ersten Stufe des Verfahrens wird nur ein Artikel (Teil) betrachtet. Für diesen Artikel existiert mindestens eine Dispositionsstrategie. Diese wiederum zeichnen sich im Wesentlichen durch zwei Merkmale aus:

- die zugrunde liegende Logik
- die verwendeten Parameter (insbesondere deren Werte)

---

<sup>1</sup>Siehe Kapitel 1.3 und 1.4.

Angenommen ein Artikel erreicht den Meldebestand, so wird eine definierte Menge nachbestellt bzw. in Produktion gegeben. Dieses Vorgehen beschreibt eine sehr einfache Dispositionsstrategie mit genau zwei Parameter, der Losgröße ( $Q$ ) und dem Meldebestand ( $s$ ).

Die Logik in diesem Beispiel kann noch weiter vereinfacht werden indem der Meldebestand  $s = 0$  gesetzt wird. Weitere vereinfachende Annahmen sind ein konstanter Bedarf, Produktionszeit gleich null, Rüst- und Lagerkosten sind konstant und die einzige Variable bleibt nun die Losgröße. Sobald der Lagerstand null erreicht, wird die Produktion mit der optimalen Losgröße ausgelöst und die produzierte Menge gelangt umgehend in das Lager. Für dieses äußerst simple Modell existiert ein einfacher analytischer Lösungsweg (vgl. Kapitel 2.2).

Stufe 1 der Heuristik sollte aber zumindest prinzipiell mit jeder Strategie arbeiten können. Die einzige Vorgehensweise die diese Forderung erfüllt, ist wohl am besten mit den Schlagworten „Brute Force“ umreißbar. Hierbei wird die Zeit diskretisiert (z. B. Wochen) und die Logik ähnlich einer Lagerbestandssimulation abgebildet. Des Weiteren wird eine endliche Periode, für die es Bedarfsprognosen gibt, festgelegt. Nun wird noch vor dem Start der eigentlichen Heuristik nach einem sinnvollen Wertebereich für die zu optimierenden Parameter gesucht. Im oben beschriebenen einfachsten Fall lässt sich der einzige Parameter, die Losgröße, in einem Bereich zwischen Null und dem gesamten Bedarf im betrachteten Zeitraum festlegen. Nun werden die Lagerbewegungen über die gesamte Periode mit allen möglichen Werten für die Losgröße simuliert. Aus den Lagerbewegungen lassen sich nun die durch diesen Teil verursachten Kosten errechnen. Als Ergebnis erhält man die Losgröße, bei der die Kosten minimal waren. Dieses Vorgehen liefert für jede abbildbare Dispositionsstrategie die optimale Lösung.

Abbildung 3.1 zeigt eine Tabelle, welche diesen Brute-Force-Ansatz an einem allgemeinen Beispiel nochmals illustriert. Die angenommene Dispositionsstrategie verfügt über die drei Parameter  $A, B, C$  mit den folgenden Wertebereichen  $A = \{1, 2, 3\}$ ,  $B = \{1, 2, 3\}$ ,  $C = \{1, 2, 3\}$ . Dadurch ergeben sich  $3 * 3 * 3 = 27$  mögliche Parameterkombinationen, für welche die daraus resultierenden Lagerbewegungen und in weiterer Folge die hierdurch entstehenden Kosten berechnet werden müssen. Es lässt sich anschließend jenes Parameterset ermitteln, welches die geringsten Gesamtkosten verursacht hat. In diesem simplen Beispiel könnte nun Parameter  $A$  für die Losgröße stehen und Parameter



$B$  und  $C$  für zwei Produktionszeitpunkte in der betrachteten Periode.

A	B	C	Kosten	
1	1	1	10,63	
1	1	2	8,54	
1	1	3	15,54	
1	2	1	11,31	
1	2	2	10,77	
1	2	3	10,59	
1	3	1	9,24	
1	3	2	<b>8,16</b>	3. Lösung
1	3	3	13,30	
2	1	1	10,48	
2	1	2	8,30	
2	1	3	10,03	
2	2	1	16,98	
2	2	2	14,88	
2	2	3	<b>8,12</b>	2. Lösung
2	3	1	15,92	
2	3	2	8,75	
2	3	3	<b>7,60</b>	Beste Lösung
3	1	1	9,44	
3	1	2	10,98	
3	1	3	11,96	
3	2	1	16,26	
3	2	2	11,14	
3	2	3	8,18	
3	3	1	10,67	
3	3	2	10,82	
3	3	3	9,49	

Abbildung 3.1: Brute-Force-Illustration mit drei Parametern: A,B,C.

Jenes Parameterset welches die beste Performance (die niedrigsten Kosten) bietet, wird somit als optimale Lösung ermittelt. Beachtenswert hierbei ist, dass die zu optimierenden Parameter ebenso wie die Zeit diskretisiert sein müssen, da man nur so eine

endliche Menge an Kombinationen erhält.

Bei der Betrachtung von Tabelle 3.1 fallen auch schnell die Grenzen der Brute-Force-Methode ins Auge. Die Anzahl der möglichen Kombinationen  $Kom$  errechnet sich nach Formel 3.1.

$$Kom = \prod_{i=1}^c \left( \frac{A_{i_{max}} - A_{i_{min}}}{A_{i_{step}}} \right) \quad (3.1)$$

Die in Formel 3.1 verwendeten Variablen werden in Tabelle 3.1 genauer beschrieben.

$Kom$	Anzahl der möglichen Parameterkombinationen
$A_{1...c}$	Die Parameter der jeweiligen Strategie
$c$	Anzahl der verwendeten Parameter
$A_{i_{max}}$	Maximaler Wert des Parameters $A_i$
$A_{i_{min}}$	Minimaler Wert des Parameters $A_i$
$A_{i_{step}}$	Schrittweite bei der Diskretisierung

Tabelle 3.1: Variablen für die Berechnung der Anzahl an Parameter-Kombinationen

Das Hinzufügen eines Parameters führt zu einem exponentiellen Anstieg der möglichen Kombinationen. Aus der Forderung der Praxistauglichkeit und mit den heute verfügbaren Rechenkapazitäten ergibt sich ein maximaler Komplexitätsgrad einer Strategie (Anzahl und Wertebereich der Parameter), ab dem der Rechenaufwand nicht mehr zu rechtfertigen ist. Diese Problematik wird im Kapitel 4 noch ausführlich behandelt.

## 3.2 Stufe 2 – Schrittweise Näherung

Aus der Brute-Force-Methode ergibt sich die optimale Lösung für einen Teil, es werden jedoch keine kapazitiven Beschränkungen berücksichtigt. In einem realen Unternehmen werden meist viele unterschiedliche Artikel auf denselben Produktionsressourcen hergestellt. Somit ist eine kapazitive Abstimmung unabkömmlich.

Um diese Funktionalität gewährleisten zu können, muss schon in der ersten Stufe der Heuristik eine Vorkehrung getroffen werden. Bei der Brute-Force-Methode wird nicht nur die optimale Parameterkombination ermittelt, sondern auch eine gewisse Teilmenge der möglichen Kombinationen, die nur geringfügig von der optimalen Performance abweichen. Die erste Stufe liefert also nicht nur eine Lösung, sondern für jeden Teil

eine Lösungsmenge mit Parameterkombinationen die jeweils leicht unterschiedliche Produktionszeitpunkte bzw. -mengen zur Folge haben.

In der zweiten Stufe der Heuristik gilt es nun, aus den möglichen Lösungen für die einzelnen Teile ein Kombination zu finden, welche die Produktionskapazitäten berücksichtigt, aber gleichzeitig eine möglichst gute Gesamtperformance bietet (möglichst geringe Gesamtkosten). Anhand nachfolgendem Beispiel soll die gewünschte Funktionalität veranschaulicht werden:

Wir betrachten ein Unternehmen mit genau einer Produktionsmaschine, welche pro Periode eine konstante Kapazität zur Verfügung stellt. Dieses Beispielunternehmen fertigt drei unterschiedliche Artikel. In Abbildung 3.2 werden die Losgrößen und Produktionszeitpunkte, die aus der jeweils besten Parameterkombination der Stufe 1 resultieren, für einen Betrachtungszeitraum von drei Perioden dargestellt.

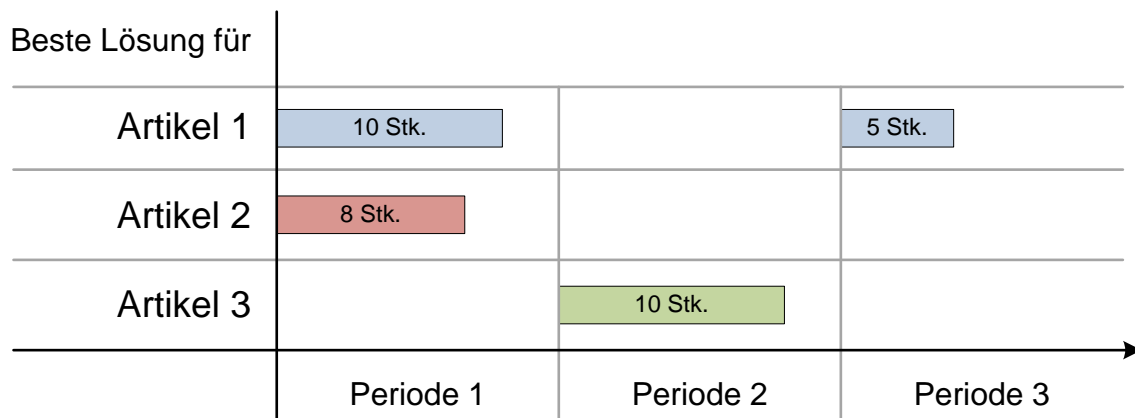


Abbildung 3.2: Optimallösungen für drei Artikel (Beispiel)

Die Kapazität der Produktionsmaschine beträgt 13 Stück pro Periode. Zur Vereinfachung des Beispiels lautet die Annahme, dass alle drei Artikel dieselben Produktionskapazitäten benötigen.

Abbildung 3.3 zeigt die resultierende Maschinenbelegung, falls für jeden Artikel die Optimallösung aus Stufe 1 verwendet wird.

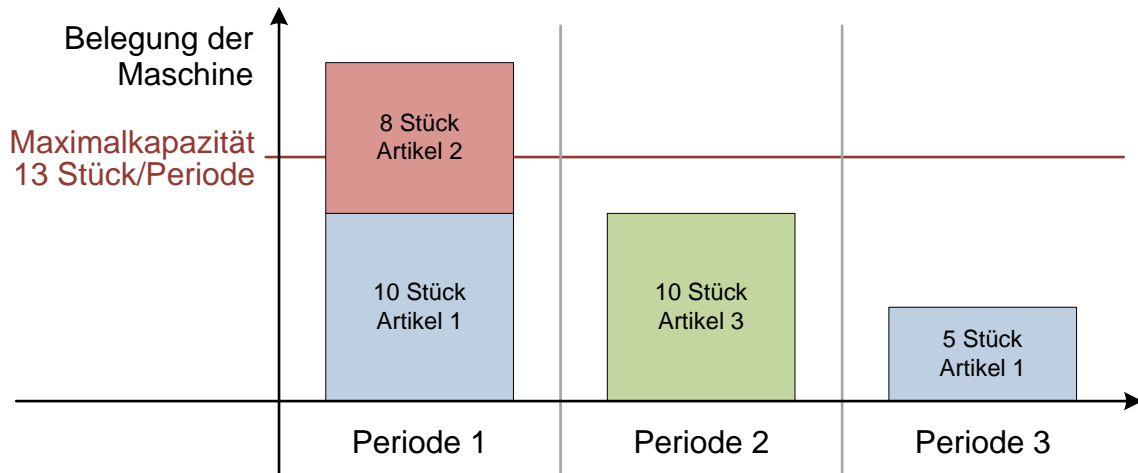


Abbildung 3.3: Überbelegung der Produktionskapazitäten (Beispiel)

Die Kapazitätsüberschreitung in Periode eins führt in einem realen Unternehmen zu Problemen bzw. überproportional erhöhten Kosten (Fehlmengen, Überstunden, ...). Die Anforderung an die Heuristik in Stufe 2 lautet nun aus den bekannten Lösungen eine durchführbare und falls möglich gute Alternativ-Kombination zu wählen.

In Abbildung 3.4 sind die Lösungsmengen dargestellt, welche für die jeweiligen Artikel in Stufe 1 der Heuristik generiert wurden. Die Ergebnisse (Parameterkombinationen) wurden aufsteigend nach den verursachten Kosten sortiert.



Abbildung 3.4: Lösungsmenge aus Stufe 1 der Heuristik (Beispiel)

Werden nun anstatt der besten Lösung für die betreffenden Artikel auch die anderen Möglichkeiten in Betracht gezogen, so lässt sich die Belegungssituation der Produktionsressourcen beeinflussen.

Abbildung 3.5 illustriert die resultierende Maschinenbelegung für die in Abbildung 3.4 orange markierten Produktionsstrategien.

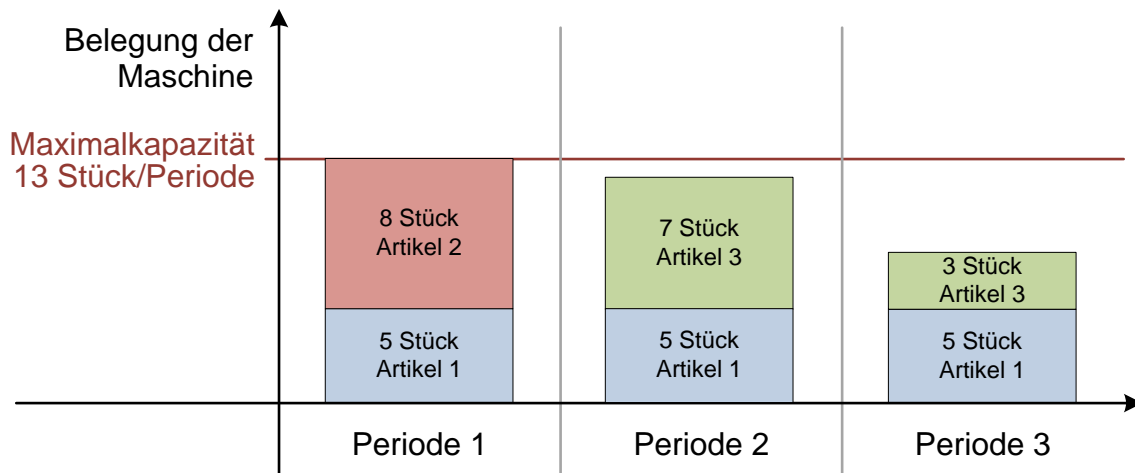


Abbildung 3.5: Maschinenbelegung nach Heuristik Stufe 2 (Beispiel)

Die Anforderung an die Heuristik in Stufe 2 lautet, eine durchführbare Lösung auf Basis der zur Verfügung stehenden Lösungsmengen der Artikel und unter Berücksichtigung aller Kapazitäten für die Fertigung des gesamten Produktionsprogramms zu finden.

Der Brute-Force-Ansatz aus Stufe 1 ist allerdings hier keine Option, da das einfache Durchprobieren aller möglichen Kombinationen nicht mit den zur Verfügung stehenden Rechenkapazitäten in einem angemessenen zeitlichen Rahmen machbar ist. Dies wird in Kapitel 4 noch ausführlich begründet.

Die Anforderungen an die Heuristik in Stufe 2 weisen gewisse Analogien zu klassischen Problemen der Informatik auf.

Die erste offensichtliche Ähnlichkeit ergibt sich mit dem Problem des Handlungsreisenden (TSP). Wir haben von Stufe 1 eine Menge an Lösungen für jeden Artikel erhalten. Diese Lösungen sind in absteigender Reihenfolge nach ihrer Performance (verursachte Kosten) sortiert. Kombinieren wir die Gesamtlösung aus der besten (preiswertesten) Lösung von jedem Artikel, so kann der hierbei entstehende Produktionsplan Überbelegungen von Ressourcen verursachen. Die Analogie zum Problem des Handlungsreisenden ergibt sich sobald man die Artikel als Orte und die Strategien als mögliche Pfade interpretiert. Jeder Artikel soll im geforderten Ausmaß bereitgestellt werden. Die von den jeweiligen Strategien verursachten Kosten können als Länge des Pfades zu dem zugehörigen Artikel interpretiert werden.

Im Gegensatz zum klassischen Travelling-Salesman-Problem ist bei unserer Aufgabenstellung die Ideallösung aber bereits bekannt. Wir suchen nach einer möglichst guten Lösung, welche die beschränkten Kapazitäten berücksichtigt. Außerdem muss das Erreichen eines Ortes (Bereitstellung eines Artikels) keine direkten Konsequenzen für alle nachfolgenden Operationen haben. Dies ist beim Problem des Handlungsreisenden sehr wohl der Fall.

Die Nearest-Neighbour-Heuristik 2.5 ist eine bekannte Herangehensweise an das Problem des Handlungsreisenden. Auch für die Aufgabenstellung in Stufe 2 ist diese Vorgehensweise denkbar. Man erstellt einen Produktionsplan mit den Ideallösungen von jedem Artikel. Im nächsten Schritt wird eine kapazitive Überbelegung gesucht. Für jenen Artikel, der diese Überbuchung verursacht, wird die nächstbeste Strategie gewählt und die Suche nach der kapazitiven Überbelegung im Produktionsplan beginnt von neuem.

Die Aufgabenstellung weist außerdem eine gewisse Ähnlichkeit mit dem Rucksack-Problem 2.5 auf. Die begrenzte Kapazität einer Produktionsressource pro Periode verkörpert die Größe des Rucksacks. Die einzelnen Lose der unterschiedlichen Artikel die pro Periode gefertigt werden sollen, belegen die Kapazitäten durch ihren Bedarf an Bearbeitungszeit. Beim klassischen Rucksackproblem ist allerdings das Ziel der Optimierung ein möglichst vollständig gefüllter Rucksack. Dies ist hier keine Priorität, vielmehr geht es um die Vermeidung von Überbelegungen durch Auswahl einer Strategie, die zu einer praktikablen Belegung aller Produktionsressourcen führt. Das Ziel hierbei ist außerdem möglichst wenig Mehrkosten, durch die Wahl von suboptimalen Strategien für die einzelnen Artikel, entstehen zu lassen.





# Umsetzung

Im Rahmen dieser Arbeit wurde ein Heuristik erdacht um die Frage nach der Losgröße automatisiert beantworten zu können. Die Beschreibung der Softwareimplementierung folgt nun in diesem Kapitel einem Top-Down-Ansatz. Ausgehend von einer systemweiten Betrachtungsweise und der zum Einsatz kommenden Technologien, wird zu Beginn die entstandene (Software-) Struktur des Projektes beschrieben.

In der weiteren Folge werden die einzelnen Komponenten, deren Details und Implementierungen genauer betrachtet.

Abschließend werden die für einen potentiellen Anwender zur Verfügung stehenden Funktionalitäten näher beleuchtet, unter der Zuhilfenahme von Screenshots der jeweiligen Menüs.

## 4.1 Struktur und Technologien

Alle Bestandteile des Projektes basieren auf quelloffenen Software-Komponenten, welche außerdem unter einer kostenfreien Lizenz verfügbar sind. Die Software orientiert sich an einem konventionellen Client-Server-Modell. Der Benutzer bedient das System von einem Client-Terminal aus, welches nur über einen Browser verfügen muss. Die aus den Bedieneingaben resultierenden rechenintensiven Algorithmen werden auf dem Server abgearbeitet. Als Betriebssystem auf Serverseite kommt Linux (Debian) zum Einsatz. Das GUI ist webbasierend (Hypertext Markup Language (HTML) 4.0 strict mit Cascading Style Sheets (CSS)) und gewährleistet somit auf Seite des Anwenders

Plattformunabhängigkeit. Serverseitig wird das Graphical User Interface (GUI) von dem Webserver Apache mit Hypertext Preprocessor (PHP) als Skriptsprache zur Verfügung gestellt. Als Datenbank kommt MySQL zum Einsatz. Dies ist eines der am weitest verbreiteten relationalen Datenbankverwaltungssysteme, welches oft in Kombination mit dem Apache Webserver verwendet wird. Die Entscheidung für MySQL begründet sich hauptsächlich in meinen persönlichen Erfahrungen und Kenntnissen. Ein technischer Grund welcher für MySQL und gegen z. B. PostgreSQL spricht ist nicht vorhanden. Für die Abarbeitung der in Kapitel 3 beschriebenen Rechenoperationen, kommen Konsolenapplikationen zum Einsatz. Diese beiden Programme (Stufe 1 und 2) sind in C++ programmiert. Die soeben beschriebenen Komponenten und deren Verbindungen untereinander, werden in der Abbildung 4.1 dargestellt: Dies soll einen besseren Überblick ermöglichen.

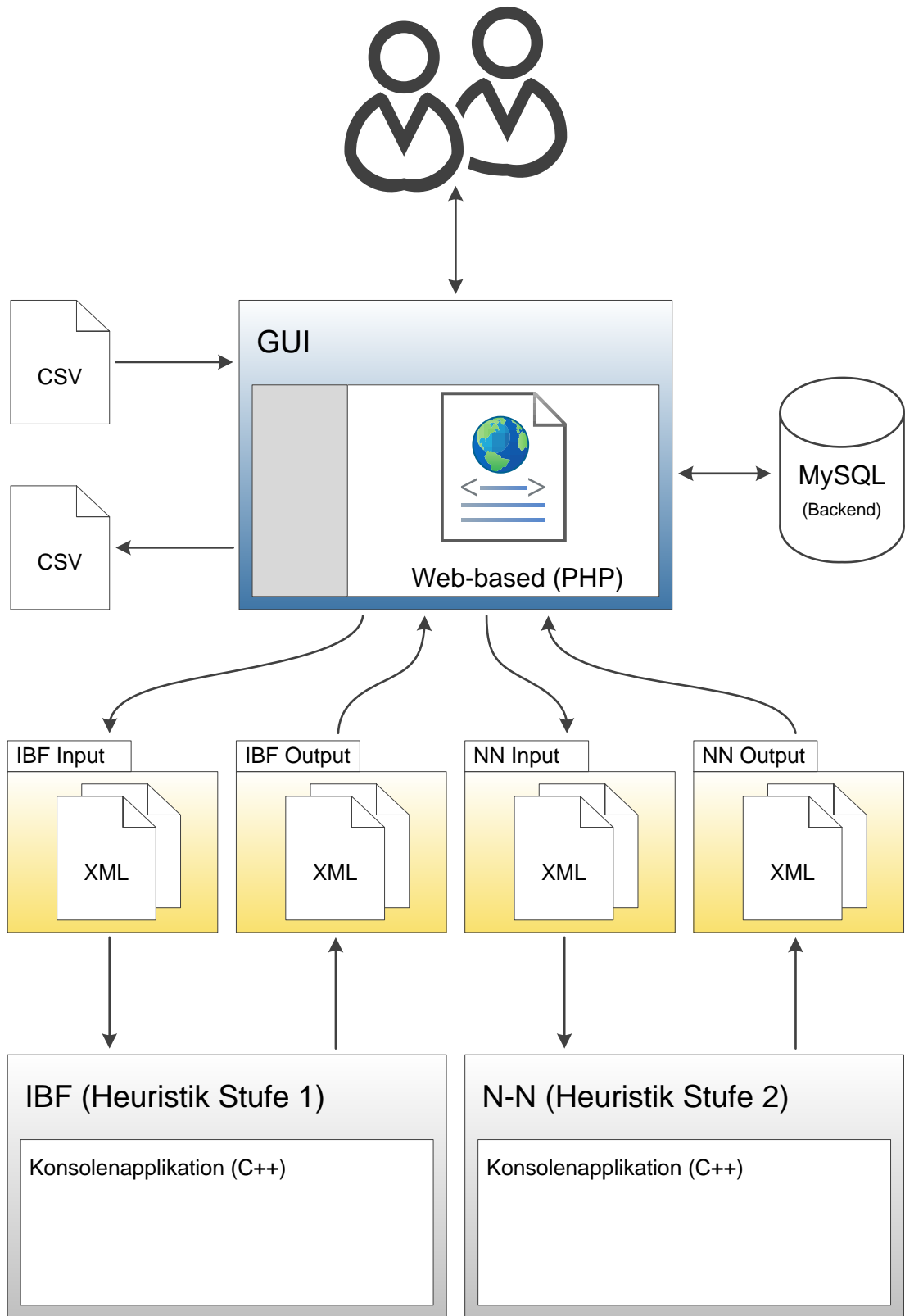


Abbildung 4.1: Struktur des Software-Prototypen

Ausgehend von den Datenströmen kann das GUI als zentrales Element bezeichnet werden. Die Bedienoberfläche bietet dem Benutzer die Möglichkeit, Daten aus anderen Systemen (ERP, MES, ...) via Comma-Separated Values (CSV) Dateien zu importieren. Die Daten werden für die weitere Verarbeitung in der SQL-Datenbank abgelegt. Nach Abfrage aus der Datenbank wird der Export von errechneten Ergebnissen ebenfalls via CSV-Dateien ermöglicht. Das GUI verfügt somit über mehr Funktionalitäten als nur die reine Benutzinteraktion. Es müssen die Daten aus der Datenbank aufbereitet und modifiziert werden, ehe sie z. B. an die Optimierungsalgorithmen in den Konsolenprogrammen (Stufe 1 und 2) übergeben werden. Diese Vorgänge sind ebenfalls in der Komponente GUI realisiert, da hier die Datenbankanbindung gegeben ist.

Die Koppelung zwischen Benutzeroberfläche und den Konsolenapplikationen, welche die Heuristiken für Stufe 1 und 2 durchführen, erfolgt über das Dateisystem. Es existiert für jede Konsolenapplikation ein Input- und Output-Verzeichnis. Hier werden Extensible Markup Language (XML) Dokumente abgelegt, welche die benötigten Informationen für die Berechnung bzw. die Ergebnisse beinhalten.

In den nächsten Kapitel dieser Arbeit werden alle Komponenten des Softwareprojektes detailliert beschrieben. Den Anfang machen hierbei die beiden Konsolenapplikationen welche jeweils Stufe 1 bzw. 2 der Optimierungsheuristik verkörpern. Umfangreiche Kenntnis über die Vorgänge in diesen beiden C++ Programmen erleichtert die abschließende Beschreibung des GUI.

## 4.2 Stufe 1 – Brute Force

In Stufe 1 wird die Losgrößenproblematik isoliert für jeden Artikel betrachtet. Die Vorgehensweise hierbei lässt sich, wie schon in Kapitel 3.1 erwähnt, am besten mit dem Schlagwort *Brute-Force* umreißen.

Das GUI hat auf Anstoß des Benutzers einen Datensatz für die erste Stufe der Heuristik generiert und in das IBF-Input-Verzeichnis in Form einer XML-Datei abgelegt. Im ersten Schritt werden nun alle benötigten Informationen (siehe 4.2.1) aus diesem XML gelesen.

Für den bzw. die betrachteten Artikel kommen eine oder auch mehrere mögliche Dispositionsstrategien in Frage. Aus den gelesenen Daten werden jetzt die Anzahl

und die Wertebereiche der benötigten Parameter für die nachfolgenden Berechnungen ermittelt.

Die Applikation beginnt nun für jeden der möglichen Parametersätze eine komplette Betrachtungsperiode zu berechnen. Als Ergebnis erhält man die Kosten für die Bereitstellung der jeweiligen Artikel im gewählten Zeitraum. Dies basiert auf der gewählten Strategie, den Materialbewegungen und den Kostensätzen für Rüstoperationen, Lagerhaltung usw.

Bei dieser immer wiederkehrenden Kostenberechnung für die unterschiedlichen Parametersätze, werden nur jene Kombinationen abgespeichert, welche ein gewisses Potential aufweisen. Zum Beispiel wird eine Kombination nur dann abgespeichert, wenn sie zur Laufzeit zu den 1000 besten Parametersets gehört. Alle anderen Kombinationen werden verworfen. Dies ist realisiert mit einer sortierten Liste, welche die aktuell 1000 besten Parametersets beinhaltet. Wird ein neues Set berechnet und erreicht günstigere Gesamtkosten als das schlechteste Set in der Liste, so wird das neue Ergebnis an passender Position in die Liste eingefügt und das letzte Set aus der Liste gelöscht.

Abschließend werden diese Parameterkombinationen aus der Liste, welche zu besonders günstigen Gesamtkosten führen, in eine XML-Datei geschrieben und über das Ausgabeverzeichnis von Stufe 1 dem Anwender zur Verfügung gestellt. Dieser kann nun wiederum unter Verwendung des browserbasierten GUI die Ergebnisse in die Datenbank importieren.

Das Flussdiagramm in Abbildung 4.2 soll die soeben skizzierten Vorgänge nochmals veranschaulichen.

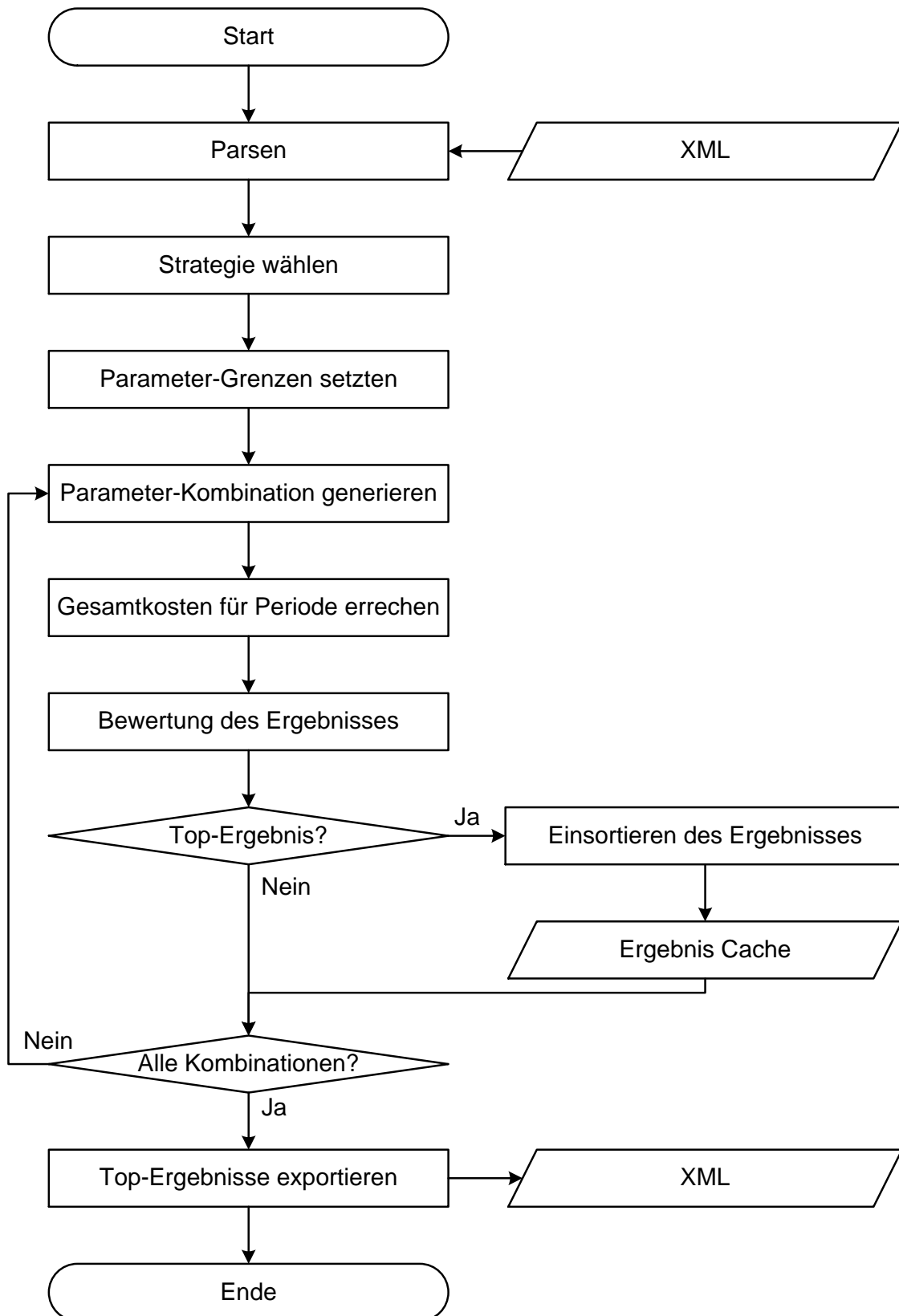


Abbildung 4.2: Flussdiagramm von Stufe 1 der Heuristik

### 4.2.1 Input Stufe 1

Die Daten, auf denen alle Operationen in dieser Stufe basieren, lassen sich in drei Kategorien einordnen.

- **Artikel-Stammdaten:** Angefangen bei der Artikelnummer und der Bezeichnung bis zu den Parametern, welche benötigt werden, um die Gesamtkosten in der betrachteten Periode zu berechnen. Hier sind vor allem die Kosten für die Rüstoperation und die Lagerkosten pro Periode maßgebend. Auch der Anfangsbestand, Fehlkosten, Kosten für einen Lagerzugriff und die Bearbeitungskosten werden übergeben und fließen später in die Berechnungen ein.
- **Strategie-Parameter:** Es muss definiert werden, welche Dispositionsstrategie für den Artikel verwendet werden soll (kein Klartextname, sondern ein numerischer Code für die hinterlegten Strategien). Außerdem müssen die hierbei verwendeten Parameter und deren Wertebereiche angegeben werden, innerhalb derer sich die Optimierung bewegen darf. Die Wertebereiche im XML wurden schon von dem GUI auf eine Schrittweite von 1 bei der Diskretisierung normiert. Die für die Umrechnung benötigten Multiplikatoren werden in der Datenbank gespeichert und beim Import der Ergebnisse berücksichtigt.
- **Bedarfsverlauf:** Die Materialbewegungen werden im Betrachtungszeitraum simuliert, um die Kosten des aktuellen Parametersatzes ermitteln zu können. Hierfür sind Angaben über die auftretenden Bedarfe notwendig. Es werden zu diesem Zweck alle Lager-Abgänge mit ihrer Menge und dem zugehörigen Zeitpunkt aufgelistet. Da es sich um Abgänge handelt, sind die Mengenangaben mit einem negativen Vorzeichen versehen. Auf diese Art und Weise lassen sich prinzipiell auch Lager-Zugänge berücksichtigen, welche nicht der Dispositionsstrategie bzw. der Optimierung unterliegen.

Abbildung 4.3 zeigt einen Auszug aus einem exemplarischen XML-Dokument, welches als Datenquelle für Stufe 1 der Heuristik fungiert.

```

<S1>
  <dataset>0001</dataset>
  <artikel>
    <stammdaten>
      <artikelnummer>100001</artikelnummer>
      <bezeichnung>Testartikel1</bezeichnung>
      <lagerstand>0</lagerstand>
      <ruestkosten>1333</ruestkosten>
      <bearbeitungskosten>132</bearbeitungskosten>
      <lagerkosten>4</lagerkosten>
      <einlagerkosten>0</einlagerkosten>
      <auslagerkosten>0</auslagerkosten>
      <fehlkosten>1465</fehlkosten>
      <von>1241042400</von>
      <bis>1278194400</bis>
    </stammdaten>
    <strategie>
      <typ>1</typ>
      <pla>0</pla>
      <p1s>1</p1s>
      <p1e>62</p1e>
      <p2a>0</p2a>
      <p2s>1</p2s>
      <p2e>450</p2e>
      <p3a>0</p3a>
      <p3s>1</p3s>
      <p3e>62</p3e>
      ...
      <p6a>0</p6a>
      <p6s>1</p6s>
      <p6e>450</p6e>
    </strategie>
    <bedarf>
      <buchung>
        <menge>-11</menge>
        <datum>1242381600</datum>
      </buchung>
      ...
      <buchung>
        <menge>-27</menge>
        <datum>1277114400</datum>
      </buchung>
    </bedarf>
  </artikel>
  <artikel>
    ...
  </artikel>
</S1>

```

Abbildung 4.3: Daten für Stufe 1 (XML)



Das Wurzelement des dargestellten XML-Dokuments ist mit S1 bezeichnet. Dies steht für Stufe 1 der Losgrößenoptimierung. Das erste Element `<dataset>` beinhaltet einen Identifier (ID), welcher die aktuelle Konfiguration dieses Rechendurchlaufs kennzeichnet. Konkret lassen sich mit dieser ID die errechneten Ergebnisse später in dem GUI zuordnen. Die Zusammenhänge betreffend das Element `<dataset>` werden in Kapitel 4.2.4 noch genauer beschrieben.

Im nächsten Element `<artikel>` finden sich alle für die Berechnung relevanten Informationen. Das `<artikel>` Element tritt für jeden in Betracht gezogenen Artikel einmal auf. Unter `<stammdaten>` finden sich die Artikelnummer, die Bezeichnung, der Lagerstand und die Kostensätze. Die Lagerkosten beziehen sich hierbei auf eine Periodendauer, welche für die Berechnung im gesamten Softwarepaket konsistent definiert wird. In den beiden Elementen `<von>` und `<bis>` befinden sich Unix-Timestamps, welche den Beginn und das Ende des betrachteten Zeitraums angeben.

Im `<strategie>` Element werden alle für die Logik der Berechnung notwendigen Informationen transportiert. Der Inhalt des `<typ>` Elements identifiziert die zu verwendende Dispositionsstrategie. Die benötigten Parameter finden sich in den `<pxx>` Elementen. Das Minimum für Parameter 1 ist in `<p1a>` (a steht für Anfang) enthalten, das Maximum in `<p1e>` (e steht für Ende). Die Schrittweite steht im Element `<p1s>`. Es können prinzipiell beliebig viele Parameter nach diesem Schema definiert werden.

Im Element `<bedarf>` werden alle Materialflüsse aus dem Lager in der betrachteten Periode aufgelistet. Eine `<buchung>` ist mit der `<menge>` und dem Zeitpunkt (`<datum>`, wieder ein Unix-Timestamp) eindeutig definiert. Die Mengenangaben sind negativ, da es sich um Abgänge aus dem Lager handelt. Auf diese Art und Weise ist es auch möglich Materialzugänge, welche nicht von der Optimierung beeinflusst werden sollen, zu berücksichtigen. Die hier gelisteten Materialflüsse sind entweder tatsächlich geschehen, liegen also schon in der Vergangenheit, oder sind das Ergebnis einer Prognose. Materialbewegungen die in der Zukunft liegen, können nur aus Prognosen bzw. Ableitung aus den Auftragsbüchern generiert werden. Je weiter die Lagerbuchungen in der Zukunft liegen, umso größer ist die hierbei entstehende Unsicherheit. Dies ist ein wesentlicher Schwachpunkt in der hier vorgestellten Heuristik zur Losgrößenoptimierung. In Kapitel 5.2 wird noch detaillierter auf diese Problematik eingegangen.

Mit den in diesem XML-Dokument transportierten Informationen können alle Berechnungen in Stufe 1 der Heuristik durchgeführt werden.

### 4.2.2 Strategien und Parameter

Bis jetzt wurden die verwendeten Dispositionsstrategien nicht näher erörtert. Dies soll nun anhand der später in dem GUI angebotenen Optionen geschehen. Die nachfolgenden Strategien wurden aus zwei Gründen für die prototypische Implementierung ausgewählt. Zum einen sind sie in dieser Form gut kompatibel zu den Möglichkeiten der meisten ERP- und MES-Systeme und die errechneten Parameter können somit leicht von diesen Produktivsystemen verwendet werden. Außerdem sind sie einfach zu implementieren, insbesondere was die Berechnung der Gesamtkosten betrifft. Prinzipiell kann jede beliebige Strategie hier angeboten werden, sofern man bereit ist den damit verbundenen Aufwand für die Implementierung der Berechnungen in Stufe 1, zu betreiben.

- **Planstrategie-2-Punkt:** In der bei der Simulation betrachteten Periode kann ein Artikel höchstens zweimal produziert werden. Die Zeitpunkte an denen der Artikel durch die Fertigung läuft sind frei innerhalb des Betrachtungszeitraums (z. B. sechs Monate) wählbar, ebenso die hierbei produzierte Menge (Losgröße). Die Strategie benötigt also vier Parameter: die beiden Produktionszeitpunkte plus deren jeweilige Losgröße. Die Wertebereiche für die beiden Zeiten lassen sich durch die gewählte Periode eingrenzen. Die Werte der Losgrößen liegen zwischen null und dem Gesamtbedarf in der Periode.
- **Planstrategie-3-Punkt:** Wie 2-Punkt, nur dass drei Produktionszeitpunkte in Betracht gezogen werden. Dies resultiert in sechs benötigten Parametern.
- **s-Q-Strategie:** Hierbei werden nur zwei Parameter verwendet. Der als s bezeichnete Wert gibt den Auslösebestand an. Erreicht die Anzahl der lagernden Artikel diese Schwelle, so wird ein Produktionsauftrag ausgelöst. Q bestimmt die hierbei gewählte Losgröße. Die resultierenden Produktionszeitpunkte werden bei dieser Strategie nur indirekt von den Parametern bestimmt, da sie sich erst im Zeitverlauf durch die Lagerbewegungen ergeben.

In Stufe 1 muss die Berechnung der Kosten für ein bestimmtes Parameterset anhand der von der verwendeten Dispositionsstrategie vorgegebenen Logik erfolgen. Konkret werden dabei die Lagerbewegungen für die gesamte Planungsperiode durchgespielt und somit die Lager-, Rüst-, Bearbeitungs-, Kapitalbindungs- und Fehlkosten berechnet. Die hierfür benötigten Informationen wurden im vorigen Kapitel 4.2.1 erörtert.

An dieser Stelle müssen noch die globalen Parameter erwähnt werden, welche für die Berechnungen notwendig sind. Im Gegensatz zu den in Kapitel 4.2.1 genannten Punkten, sind die globalen Parameter nicht von dem jeweiligen Artikel abhängig, sondern gehören zu einem Dataset.

- Temporale Schrittweite (Tick): Zeitpunkte werden als Integer-Wert mit einer Auflösung von einer Sekunde abgespeichert (Unix-Timestamp). Die Sekunde ist somit die kleinste Einheit, mit der die Simulation rechnen kann. Für das Durchspielen der Lagerbewegungen eines bestimmten Parametersets wäre eine Schrittweite von nur einer Sekunde nicht sinnvoll. Ein praktikabler Wert ist ein Tag (86400 s) bzw. eine Woche (604800 s). Der Tick diskretisiert somit die gewählte Periode in eine überschaubare Menge an Zeitpunkten, zu denen alle Operationen (Lagerzugang, Lagerabgang, Rüsten, ...) und deren finanzielle Konsequenzen betrachtet werden.
- Limits: Es existieren Begrenzungen betreffend die Kapazität des Lagers und die maximal mögliche Losgröße. Diese Parameter müssen ebenfalls im Vorfeld für ein gesamtes Dataset vorgegeben werden.

### 4.2.3 Implementierung

Die Konsolenapplikation, welche für die Berechnungen in Stufe 1 verantwortlich ist, lässt sich grob in drei Abschnitte gliedern.

Zu Beginn erfolgen die *Initialisierung und der Datenimport* aus dem XML-File des jeweiligen Artikels. Hierbei werden alle benötigten Informationen in den Arbeitsspeicher geladen, angefangen bei den Kostensätzen, über die Strategieparameter, bis hin zu den Materialbewegungen. Die Datenmenge ist vergleichsweise gering, was die Schätzung in Tabelle 4.1 zeigt.

Stammdaten	12 * 32	Bit	=	48	Byte
Strategieparameter	18 * 32	Bit	=	72	Byte
Materialbewegungen	250 * 2 * 32	Bit	=	2000	Byte

Tabelle 4.1: Beispiel für den Speicherbedarf der Input-Daten von Stufe 1 pro Artikel

Bildet man die Summe der rechten Spalte von Tabelle 4.1 erhält man einen Speicherbedarf von 2120 Byte  $\sim$  2 Kibibyte (KiB). Zusätzlich wird für die folgenden Berechnungen noch weiterer Speicher allokiert, welcher aber in Summe unter einem KiB bleibt. Die hier abgelegten Variablen sind nötig, um Zwischenergebnisse aufzunehmen und dienen auch als Laufvariablen für Arrayindices.

Aktuelle Prozessoren verfügen über Level 2 Cache in Größenordnungen die weit über den hier abgeschätzten Werten liegen (in etwa 256 KiB bis 12 Mebibyte (MiB))<sup>1</sup>. Die benötigten Daten können also sehr „nahe“ an der CPU gehalten werden und stehen somit äußerst IO-performant zur Verfügung.

Nachdem alle Vorbereitungen getroffen wurden, kommt es im nächsten rechenintensiven Schritt zur Simulation aller Parameterkombinationen, um die Sets mit der besten Performance zu finden. Hierfür werden zu den jeweiligen Parameterkombinationen die resultierenden Kosten in der betrachteten Periode berechnet. Noch bevor die Materialbewegungen durchgespielt werden, um eine Bewertung auf Basis der Kosten durchführen zu können, muss das dafür maßgebende Parameterset generiert werden. Dieser Schritt ist nicht trivial, da die Art und Weise, in der die Parametersets erzeugt und somit auch durchlaufen werden, wesentlich die benötigte Rechenkapazität und den Speicherbedarf bestimmt.

Abbildung 4.4 zeigt das bereits aus Kapitel 3 bekannte Szenario einer Strategie mit drei Parametern (A, B, C) die jeweils Werte von 1 bis 3 annehmen können. Jede Zeile der Tabelle repräsentiert ein Parameterset, für welches die verursachten Kosten ermittelt werden müssen. Die Abbildung 4.4 macht außerdem die zu Grunde liegende Logik erkennbar, nach der durch alle möglichen Parametersets iteriert wird.

Die vorgeschlagene Vorgehensweise in Stufe 1 ist prinzipbedingt nicht sparsam, was den Verbrauch an Rechenleistung betrifft. Bei der Umsetzung dieser Brute-Force-Stufe sollte man daher besonders auf eine effiziente Implementierung achten. Der grüne Pfeil

<sup>1</sup>Vgl.: <http://de.wikipedia.org/wiki/Cache>

A	B	C
1	1	1
1	1	2
1	1	3
1	2	1
1	2	2
1	2	3
1	3	1
1	3	2
1	3	3
2	1	1
2	1	2
2	1	3
2	2	1
<b>2</b>	<b>2</b>	<b>2</b>
2	2	3
2	3	1
2	3	2
2	3	3
3	1	1
3	1	2
3	1	3
3	2	1
3	2	2
3	2	3
3	3	1
3	3	2
3	3	3




Abbildung 4.4: Schrittweises Erzeugen von Parametersets

in Abbildung 4.4 symbolisiert den Aufruf der als `kombi` benannten Funktion. Ihr Zweck ist es, ausgehend von einem Parameterset zum nächsten zu gelangen und dabei das in Abbildung 4.4 erkennbare Muster zu befolgen. Dies ermöglicht ein Durchlaufen aller Parametersets, ohne die bereits berechneten Kombinationen abspeichern zu müssen. In weiterer Folge führt dies zur Datenhaltung in den Caches der CPU während den Berechnungen<sup>2</sup> und somit zu einer maximal performanten Abarbeitung des Programmes.

In dieser Diplomarbeit wird kein Source-Code gedruckt, mit Ausnahme der in Abbildung 4.5 gezeigten Zeilen der Funktion `kombi`. Da die Implementierung der Iteration durch die Parametersets die größte Bedeutung für die Effizienz der Software hat, wird auf die Funktionsweise hier genauer eingegangen.

```

1  void kombi(int *limp, int *aktp) {
2      for (int i = 0; i < 10; ++i) {
3          if (aktp[i] < limp[i * 2 + 1]) {
4              aktp[i] = aktp[i] + 1;
5              if (i > 0) {
6                  for (int k = 0; k < i; ++k) {
7                      aktp[k] = limp[i * 2];
8                  }
9              }
10             break;
11         }
12     }
13 }

```

Abbildung 4.5: Quelltext der Funktion `kombi` zum Erzeugen der Parameterkombinationen

Neben den beiden Zählvariablen `i` und `k` verwendet die Funktion nur die beiden Arrays `limp` und `aktp`.

Das Array `aktp` (aktuelles Parameterset) beinhaltet Werte vom Typ `Integer` und hat als Länge die Anzahl der verwendeten Parameter (in Abbildung 4.5 festgesetzt auf 10). In jeder Zelle findet sich der aktuelle Wert des jeweiligen Parameters. Man könnte auch sagen, das Array beinhaltet die aktuelle Zeile aus der Tabelle in Abbildung 4.4.

Das Array `limp` (Limit-Parameter) beinhaltet ebenfalls Werte vom Typ `Integer`, hat aber die doppelte Länge. In den einzelnen Feldern werden die Minima und Maxima

<sup>2</sup>Kann der Speicherverbrauch eines Programms deutlich unter den verfügbaren Werten des Level-Caches einer CPU gehalten werden, so werden mit großer Wahrscheinlichkeit alle Variablen mit hoher Zugriffshäufigkeit im Level-Cache des Prozessors gehalten und nicht im Arbeitsspeicher.

der jeweiligen Parameter gespeichert.

Durch wiederholtes Aufrufen der Funktion `kombi`, kann mittels der beiden Arrays (welche als Pointer übergeben werden), die notwendige Iteration durch alle möglichen Parametersets bewerkstelligt werden. In Abbildung 4.4 symbolisiert der grüne Pfeil einen Aufruf der Funktion `kombi`.

Nachdem `kombi` aufgerufen wurde, muss nun die Wertigkeit des neu entstandenen Parametersets ermittelt werden. Da die einzelnen Parameter zu einer bestimmten Dispositionsstrategie gehören, ist es nun möglich, mit den Informationen über die Bedarfe und Kostensätze die verursachten Gesamtkosten<sup>3</sup> in dem Betrachtungszeitraum zu ermitteln. Diese Berechnung basiert auf dem Durchspielen aller Materialflüsse (Lager Zu- und Abgänge) des betrachteten Artikels, da durch die simulierten Lagerbewegungen und deren Zeitpunkt die benötigten Informationen zur exakten Ermittlung der Kosten generiert werden können.

Es existiert für Stufe 1 ein Output-Buffer, dessen Funktion es ist, die besten Parametersets in aufsteigender Reihenfolge, nach den verursachten Gesamtkosten sortiert, zu speichern. Je größer der Buffer (Speicherplatz) ist, um so mehr Ausweichmöglichkeiten für die Heuristik in Stufe 2 werden geschaffen<sup>4</sup>. Ein größerer Buffer bedeutet aber auch einen erhöhten Arbeitsspeicherverbrauch in Stufe 1 und mehr Rechenleistung für das Einsortieren von neuen Ergebnissen. Bei den Testläufen für dieses Projekt wurde ein Buffer-Größe von 1000 Parametersets gewählt. Ob dieser Wert in der Realität praktikabel ist, hängt letztendlich von den jeweiligen Gegebenheiten ab (Anzahl der Parameter, Dispositionsstrategie, verfügbare Produktionsressourcen, ...) und lässt sich nur experimentell in dem Anwendungsszenario ermitteln.

#### 4.2.4 Output Stufe 1

Das Ergebnis aus Stufe 1 wird wiederum in Form eines XML-Dokuments abgelegt, welches anschließend von dem GUI importiert und in die Datenbank eingetragen wird. Abbildung 4.6 zeigt einen exemplarischen Auszug aus dem Output von Stufe 1.

---

<sup>3</sup>Unter Gesamtkosten werden hier die Rüstkosten, Bearbeitungskosten, Lagerkosten, Fehlmengenkosten und Kapitalbindungskosten verstanden.

<sup>4</sup>Siehe Kapitel 3.2.

```

<S1_Result>
  <dataset>0001</dataset>
  <artikel>
    <artikelnummer>100001</artikelnummer>
    <strategie>1</strategie>
    <von>1241042400</von>
    <bis>1278194400</bis>
    <set>
      <setID>1</setID>
      <gk>116061</gk>
      <rk>2666</rk>
      <bk>59400</bk>
      <lk>49600</lk>
      <fk>4395</fk>
      <sk>258</sk>
      <p1>8</p1>
      <p2>269</p2>
      <p3>42</p3>
      <p4>181</p4>
      ...
      <fertigung>
        <buchung>
          <menge>269</menge>
          <datum>1249827630</datum>
        </buchung>
        <buchung>
          <menge>181</menge>
          <datum>1260046700</datum>
        </buchung>
      </fertigung>
    </set>
    <set>
      <setID>2</setID>
      <gk>116155</gk>
      ...
    </set>
    ...
  </artikel>
  <artikel>
    ...
  </artikel>
  ...
</S1_Result>

```

Abbildung 4.6: Beispielhafter Output aus Stufe 1 (Auszug)

Das erste Element unter dem Wurzelement ist das `<dataset>`, welches als ID schon im Input-XML angegeben wurde. Dadurch wird die Zuordnung von Inputs zu Out-



puts in Stufe 1 ermöglicht. Die nächsten Elemente sind benannt mit `<artikel>`. Die unterschiedlichen Artikel werden wiederum durch den Inhalt des `<artikelnummer>` Elements unterschieden.

Der Output-Buffer (siehe Kapitel 4.2.3) von dem Durchlauf jedes einzelnen Artikels, wird sequenziell in das XML geschrieben, welches dann für jeden Artikel 1000<sup>5</sup> unterschiedliche Parametersets enthält. Diese sind wiederum nach den von ihnen verursachten Gesamtkosten aufsteigend sortiert. Jenes Set mit den niedrigsten Gesamtkosten für die Bereitstellung des betreffenden Artikels, beinhaltet die zu favorisierenden Parameter für die verwendete Strategie.

#### 4.2.5 Performance

Der Rechenaufwand in dieser ersten Stufe der Optimierungsheuristik ist aufgrund der gewählten Herangehensweise erheblich. Die benötigten Rechenoperationen skalieren linear mit der Anzahl an möglichen Parameterkombinationen der jeweilig verwendeten Strategie. Formel 3.1 gibt die Anzahl der möglichen Kombinationen in Abhängigkeit der Parameterzahl und deren Auflösung (Schrittweite bei der Diskretisierung) an.

Eine grundlegende Anforderung an die Losgrößenoptimierung ist die Praktikabilität. Der erste Berechnungsschritt bei der Optimierung, welcher für jeden betrachteten Artikel durchgeführt wird, darf keinen unververtretbaren Rechenaufwand verursachen. Es stellt sich nun die Frage wie viele Parameter mit ihren dazugehörigen Wertebereichen und Auflösungen für eine Strategie verwendbar sind, ohne die benötigte Rechenzeit pro Artikel in unbrauchbare Höhen zu treiben.

Der Rechenaufwand für die Ermittlung der Kosten pro Artikel und verwendetem Parametersatz wird in weiterer Folge als konstant angenommen. Somit hängt der Gesamtaufwand nur von der Anzahl der getesteten Parameterkombinationen ab. Abbildung 4.7 zeigt die möglichen Kombinationen über der Anzahl der verwendeten Parameter. Hierbei werden pro Parameter 50 Diskretisierungsschritte zwischen den Bereichsgrenzen angenommen.

---

<sup>5</sup>Dieser Wert wurde als Puffer-Größe für die Testläufe der Software verwendet. Er wird als Systemparameter global festgelegt und kann somit den jeweiligen Gegebenheiten angepasst werden.

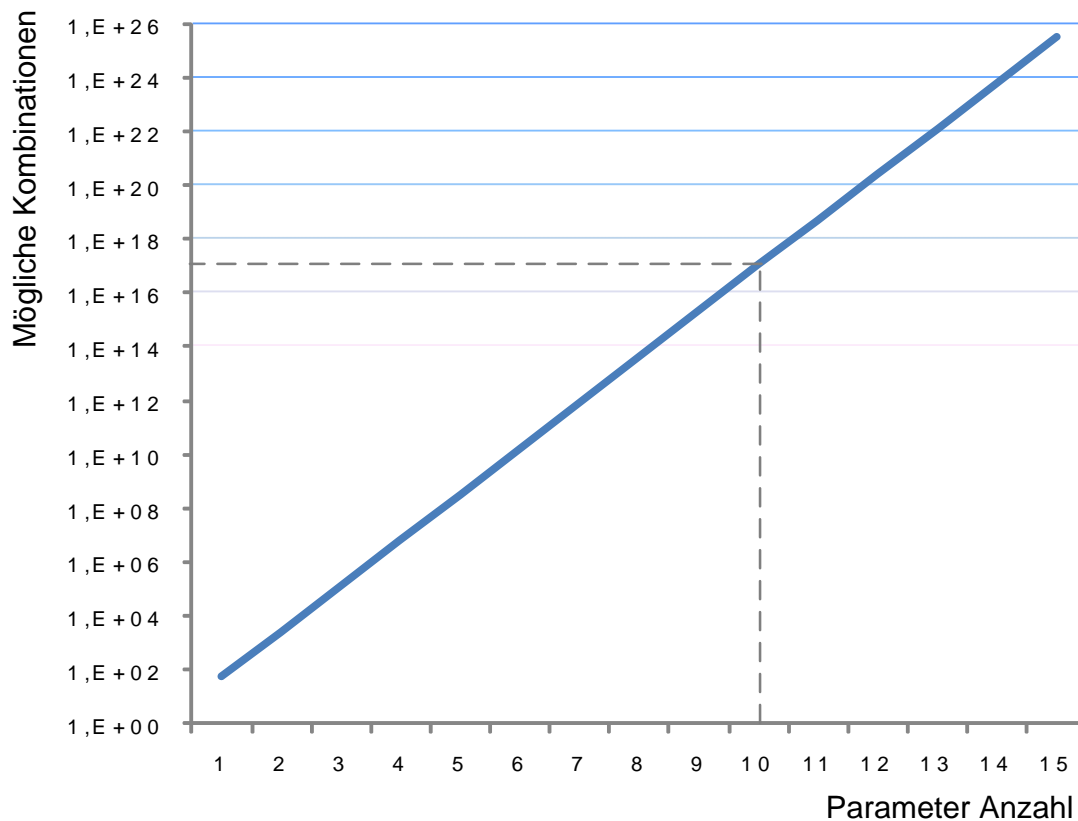


Abbildung 4.7: Mögliche Kombinationen in Abhängigkeit zur Parameter-Anzahl (50 Werte pro Parameter)

Bei der Betrachtung von Abbildung 4.7 ist der Hinweis auf die logarithmische Ordinate essentiell, da sonst die zu Grunde liegende Problematik verborgen bleibt. Die Anzahl der möglichen Parameterkombinationen steigt exponentiell mit der Anzahl der verwendeten Parameter (bei konstanter Anzahl an Diskretisierungsschritten pro Parameter). Dieses Verhalten ließ sich schon an Formel 3.1 erkennen.

Die grau strichlierten Linien zeigen den Punkt, an dem die Berechnung für alle Kombinationen auf einem Testsystem (siehe Kapitel 2.4) eine Stunde gedauert hat. Bei diesem Testlauf wurden keine Kostenberechnungen für das Parameterset durchgeführt, sondern nur die einzelnen Kombinationen durchlaufen.

Die Anzahl der verwendbaren Parameter stellt aufgrund ihres exponentiellen Einflusses auf die Rechenleistung eine harte Grenze für die Heuristik in Stufe 1 dar. Selbst mit entsprechenden Aufwänden (Parallelisierung) lässt sich diese Grenze kaum über

eine Anzahl von 10 Parameter ausdehnen, auch nicht in Zukunft unter Berücksichtigung potentieller Zuwächse bei der Rechenleistung.

In Abbildung 4.8 wird nun die Anzahl der möglichen Parameterkombinationen, in Abhängigkeit der verwendeten Diskretisierungsschritte bei konstanter Parameteranzahl (6), dargestellt. Die Ordinate ist auch hier wieder logarithmisch skaliert. Der Graph wurde unter Verwendung von Formel 3.1 erstellt und zeigt den quadratischen Zusammenhang der Anzahl an möglichen Parameterkombinationen mit dem Zuwachs an Diskretisierungsschritten.

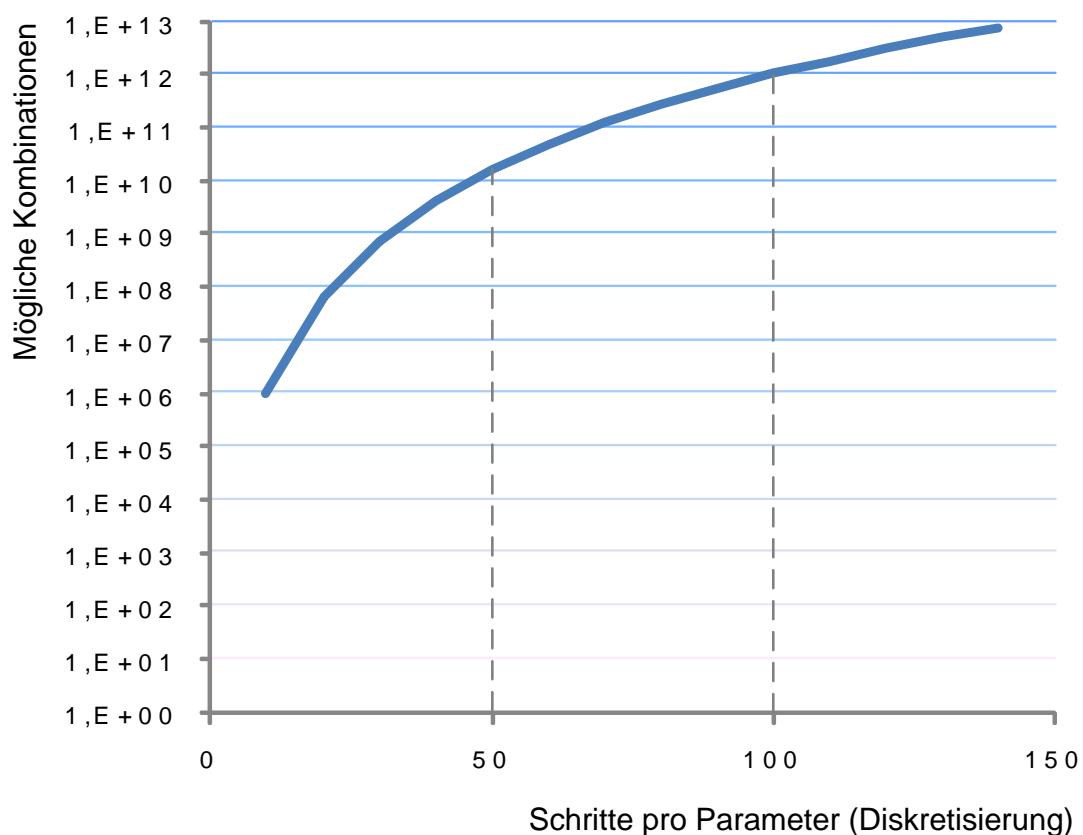


Abbildung 4.8: Mögliche Kombinationen in Abhängigkeit zur Auflösung bei der Diskretisierung (festgehaltene Parameteranzahl)

Der Anstieg der benötigten Rechenleistung mit der Auflösung bei der Diskretisierung folgt zwar quadratischen Zusammenhängen, nicht jedoch exponentiellen. Die Grenze der Berechenbarkeit ist hier somit „weicher“ und kann somit zu einem gewissen Grad durch entsprechenden Mehraufwand (Parallelisierung) verschoben werden. Auf dem ver-

wendeten Testsystem (2.4) führten Werte zwischen 50 und 100 Diskretisierungsschritte pro Parameter (Anzahl 6) zu sinnvollen Rechenzeiten (unter einer Stunde).

Die in diesem Kapitel genannten Grenzen, insbesondere für die Parameterzahl, sind als hartes Limit für den Einsatz der in dieser Arbeit vorgeschlagenen Heuristik zu verstehen. Sollten sich die Vorgänge in der Produktion eines Unternehmens nicht mittels Strategien und Parametersets abbilden lassen, die diesen Grenzen folgen, so ist eine Implementierung der gegenständlichen Herangehensweise nicht zielführend.

#### 4.2.6 IBF (Intelligent-Brute-Force)

In Abbildung 4.1 wurde die Konsolen-Applikation für Stufe 1 mit Intelligent Brute Force (IBF) titulierte. Die berechnete Frage lautet nun, warum ein Brute-Force-Ansatz als intelligent bezeichnet werden kann. Per Definition beruht die Vorgehensweise hierbei auf sturem Durchprobieren aller möglichen Parameterkombinationen, um die Lösung für ein bestimmtes Problem zu finden.

Das Attribut *intelligent* bezieht sich auf die Berechnung der Kosten für den aktuellen Parametersatz. Die benötigte Rechenzeit für das reine Durchlaufen aller Kombinationen (Funktion in Abbildung 4.5) ist verschwindend gering im Vergleich zur Kostenberechnung, die bei jeder Iteration notwendig ist, um den Parametersatz bewerten zu können. Hierbei lässt sich durch vergleichsweise einfache Abbruchbedingungen das komplette Simulieren aller Lagerbewegungen und Produktionsoperationen für die überwiegende Mehrheit aller Parametersätze vermeiden.

Exemplarisch gehen wir von einer Strategie mit sechs Parametern aus. Die ersten drei Parameter ( $W_1, W_2, W_3$ ) stehen für den jeweiligen Produktionszeitpunkt des Artikels. Die letzten drei Parameter ( $M_1, M_2, M_3$ ) sind die zugehörigen Mengen (die Losgröße). Ohne Materialflüsse durchzurechnen, um die durch diese Strategie verursachten Gesamtkosten ermitteln zu können, wird zu Beginn die Summe  $\sum_{i=1}^3 (M_i)$  gebildet. Weicht diese Summe deutlich von der in der gesamten Betrachtungsperiode benötigten Menge ab, so kann die Strategie sofort verworfen werden.

Es lassen sich je nach Strategie unterschiedliche Kriterien definieren, welche ein sehr schnelles Verwerfen eines Parametersets ermöglichen. Dies hat zur Folge, dass eine komplette Berechnung der verursachten Kosten für vergleichsweise wenige Parametersets durchgeführt werden muss. Eine ungefähre Abschätzung wie viel Rechenaufwand durch

einen vorzeitigen Abbruch eingespart werden kann, ist relativ schwierig, da dies in großem Maße von der jeweiligen Strategie und der Parameterwahl abhängig ist. Bei Testläufen der Applikation von Stufe 1 mit generierten Daten (Bedarfe, Kosten, ...) konnten nur mit der oben beschriebenen Mengen-Überprüfung 98% der Parametersets verworfen werden.

## 4.2.7 Optimierung im Vorfeld

Bei der Erstellung des Datasets für Stufe 1 werden auch die Wertebereiche für die zu variierenden Parameter errechnet. Hier kann durch eine geschickte Abschätzung der Ausgangswerte eine eklatante Einsparung an Rechenaufwand erreicht werden. Formel 3.1 ermöglicht folgendes Beispiel:

Wir betrachten eine Strategie mit vier Parametern  $A$ ,  $B$ ,  $C$  und  $D$ . Deren Wertebereiche nach Tabelle bezeichnet werden.

$$\begin{array}{rcl}
 \hline
 A_{max} - A_{min} & = & a \\
 B_{max} - B_{min} & = & b \\
 C_{max} - C_{min} & = & c \\
 D_{max} - D_{min} & = & d \\
 \hline
 \end{array}$$

Tabelle 4.2: Wertebereiche einer Beispiel-Strategie mit vier Parametern

Die zu berechnende Summe an möglichen Kombinationen errechnet sich nun zu  $a * b * c * d$  (ausgehend von auf Ganzzahlen normierte Werte).

Wird nun im Vorfeld eine Reduktion der Wertebereiche um 20% ermöglicht, so führt dies zu einer Einsparung des Rechenaufwands von annähernd 60%. Diese Aussage lässt sich wie folgt nachvollziehen:

$$a * 0.8 * b * 0.8 * c * 0.8 * d * 0.8 = a * b * c * d * 0.8^4 = a * b * c * d * 0.4096$$

Beachtenswert bei dieser Optimierung ist allerdings die mögliche Konsequenz auf das Ergebnis. Wird im Vorfeld das erreichbare Optimum eines Parameters durch zu enge Grenzen ausgeschlossen, führt dies trotz gewonnener Einsparung bei der Rechenzeit zu einem unbrauchbaren Resultat. Ein relativ sicherer Indikator für diesen Fall sind Topergebnisse, bei denen ein oder mehrere Parameter direkt auf den im Vorfeld definierten Grenzwerten zu liegen kommen.

## 4.3 Stufe 2 – Nearest Neighbor

Werden nun die besten Lösungen für jeden Artikel aus Stufe 1 zu einem Produktionsplan zusammengefügt, so kann es zu Überbelegungen der verfügbaren Produktionsressourcen kommen. Die kapazitiven Restriktionen wurden in Stufe 1 nicht berücksichtigt, da jeder Artikel isoliert betrachtet wurde. In Stufe 2 wird also überprüft, ob die Ergebnisse aus Stufe 1 direkt in den Produktionsplan übernommen werden können, oder ob es für die hier vorgeschlagene Heuristik zur Optimierung der Losgröße, einen weiteren Schritt bedarf.

Abbildung 4.9 zeigt das Flussdiagramm, welches die Abläufe in Stufe 2 auf kompakte Weise verdeutlichen soll. Zu beachten ist hierbei die Tatsache, dass in dem Flussdiagramm nicht alle möglichen Input-Szenarien berücksichtigt sind. Laut Flussdiagramm entsteht eine Endlosschleife, sobald es keine Möglichkeit gibt, mit den vorhandenen Ressourcen die geforderten Artikelmen gen zu produzieren. In der Software sind hier Mechanismen vorgesehen, welche derartige Ausnahme-Konstellationen erkennen und entsprechende Meldungen geben.

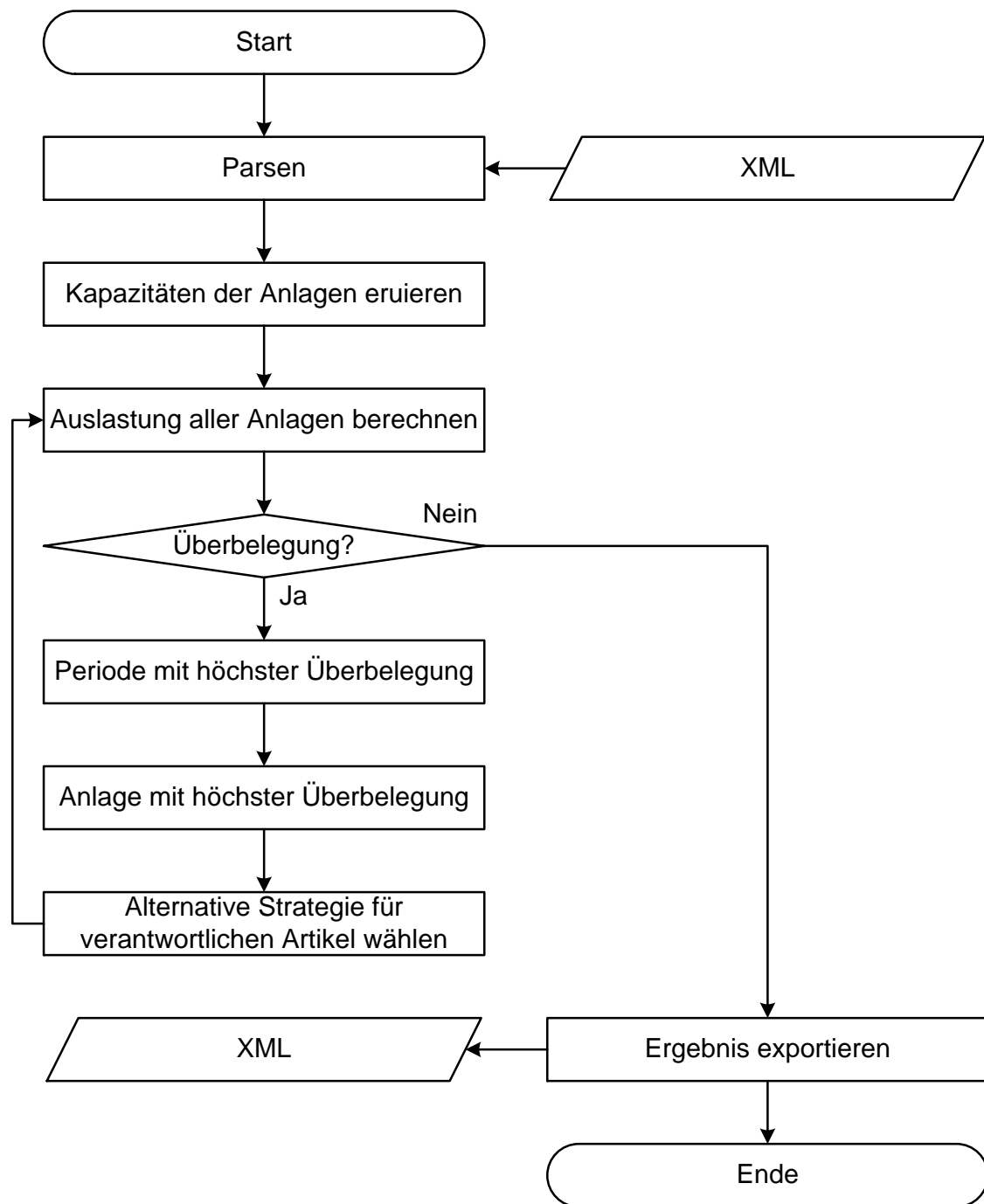


Abbildung 4.9: Flussdiagramm von Stufe 2 der Heuristik

### 4.3.1 Input Stufe 2

Die Informationen, welche in Form einer XML-Datei für Stufe 2 zur Verfügung stehen, sind in Abbildung 4.10 als exemplarischer Auszug der Datei dargestellt.

Unter dem Wurzelement `<S2>` findet sich als erstes Child-Element<sup>6</sup> `<algorithmus>`,

<sup>6</sup>Siehe [http://www.w3schools.com/dom/dom\\_nodetree.asp](http://www.w3schools.com/dom/dom_nodetree.asp) für nähere Erklärungen der Begriffe.

welches den gewünschten Optimierungsalgorithmus für Stufe 2 angibt. Im Rahmen dieser Arbeit wurde nur die N-N-Heuristik implementiert. Sollten in Zukunft unterschiedliche Verfahren zum Einsatz kommen, so kann hiermit eine Auswahl getroffen werden.

Das `<dataset>` erfüllt dieselbe Funktion wie schon in Kapitel 4.2.1 beschrieben. Auch die beiden Elemente `<von>` und `<bis>` sind schon aus dem Input für Stufe 1 bekannt und erfüllen dieselbe Funktion.

Das nächste Element `<artikel>` ist für alle betrachteten Artikel einmal vorhanden. Die unterschiedlichen Artikel werden durch das `<nummer>` Element unterschieden, welches die Artikelnummer beinhaltet.

Für die jeweiligen Artikel gibt es nun als Ergebnis aus Stufe 1 unterschiedliche Parametersets für die Dispositionsstrategien. Die Parametersets finden sich in den Child-Elementen `<variante>` des betreffenden Artikels. Jedes Parameterset lässt sich mit dem Wert des Elements `<id>` eindeutig identifizieren. Außerdem beinhaltet das Element `<gk>` die in Stufe 1 errechneten Gesamtkosten für die Bereitstellung des Artikels basierend auf den Werten des zugehörigen Parametersets.

Für die Optimierung in Stufe 2 besonders wichtige Informationen beinhaltet das Element `<produktion>`. Für jede vom Artikel benötigte Produktionsanlage wird ein `<anlage>` Element angelegt, welches eindeutig mit dem Wert des Elements `<anummer>` identifizierbar ist. Nun folgen als `<buchung>` bezeichnete Elemente, welche wiederum die Childs `<datum>` und `<dauer>` beinhalten. In `<datum>` befindet sich der UNIX-Timestamp eines vorgesehenen Produktionszeitpunktes. Das Element `<dauer>` gibt Auskunft über die Belegungsdauer der jeweiligen Anlage, welche zum Zeitpunkt `<datum>` für den jeweiligen Artikel eingeplant ist.

Beachtenswert ist, dass die Anzahl der `<variante>` Elemente abhängig von der Buffer-Größe in Stufe 1 ist. Wie viele `<buchung>` Elemente pro `<anlage>` existieren, hängt von der in Stufe 1 verwendeten Dispositionsstrategie ab.

Stehen drei Punkte unter einem Element in dem in Abbildung 4.10 gezeigten Auszug der XML-Datei, so deutet dies auf möglicherweise multiples Vorhandensein des Elementes hin. Die Anzahl der `<variante>` Elemente wird von der Buffer-Größe in Stufe 1 bestimmt. Die Zahl an `<buchung>` Elementen lässt sich durch die Wahl der Dispositionsstrategie beeinflussen. Die Anzahl der benötigten Produktionsressourcen wird von den Arbeitsplänen der jeweiligen Artikel abgeleitet. Wie viele Artikel an



sich berücksichtigt werden, ist wiederum von der Größe des Artikel-Programms des betrachteten produzierenden Betriebes abhängig.

```

<S2>
  <algorithmus>Nearest-Neighbor</algorithmus>
  <dataset>0001</dataset>
  <von>1241042400</von>
  <bis>1278194400</bis>
  <artikel>
    <nummer>100001</nummer>
    <variante>
      <id>0000001</id>
      <gk>536492</gk>
      <produktion>
        <anlage>
          <anummer>001</anummer>
          <buchung>
            <datum>1258211400</datum>
            <dauer>215</dauer>
          </buchung>
          <buchung>
            <datum>1262239400</datum>
            <dauer>160</dauer>
          </buchung>
          ...
        </anlage>
        ...
      </produktion>
    </variante>
    <variante>
      <id>0000002</id>
      <gk>536623</gk>
      <produktion>
        <anlage>
          <anummer>001</anummer>
          <buchung>
            <datum>1258038400</datum>
            <dauer>215</dauer>
          </buchung>
          ...
        </anlage>
        ...
      </produktion>
    </variante>
    ...
  </artikel>
  ...
</S2>

```

Abbildung 4.10: Input-Daten für Stufe 2 der Heuristik (Auszug aus XML-Datei)

### 4.3.2 Analogien

In Kapitel 2.5 wurde das TSP und das Rucksackproblem erwähnt. Dies ist durch die in Stufe 2 erkennbaren Analogien zu diesen beiden klassischen Problemen der Informatik begründet.

Die Ähnlichkeiten, welche die Assoziationen zu diesen beiden Lehrbuch-Problemen verursachen, werden in der Folge den Unterschieden gegenübergestellt.

Beim Rucksackproblem wird versucht den Wert der Gegenstände im Rucksack zu maximieren, bei gleichzeitiger Berücksichtigung des maximal zulässigen Gesamtgewichtes des Rucksacks. Die Produktionsressourcen haben in der Heuristik Stufe 2 die Rolle des Rucksacks. Die Optimierungskriterien sind allerdings unterschiedlich, da das Ziel nur die Vermeidung von Überbelegungen ist, nicht aber die Maximierung der Belegung an sich. Außerdem ist in diesem Zusammenhang die Frage nach dem Wert der Artikel schwierig zu beantworten. Wertigkeiten werden erst existent, wenn aufgrund einer Überbelegungssituation andere Produktionsmengen (Losgrößen) oder Zeitpunkte gewählt werden müssen. Jetzt tritt die Frage nach der Abweichung von der Ideallösung aus Stufe 1 auf, welche den geringsten Schaden anrichtet. Wertigkeit ist hier also kein Attribut, welches auf die jeweiligen Artikel bezogen wird, sondern eine individuell von der Situation abhängige Fragestellung.

Auch beim TSP wird eine Aufgabe betrachtet, welche Assoziationen zu dem in Stufe 2 vorliegenden Problem zulässt. Die einzelnen Artikel lassen sich als Reiseziele interpretieren und ihre mit den Gesamtkosten gewichteten Parametersets als Pfade. Diese aus Stufe 1 errechneten Parametersets unterscheiden sich allerdings zu den gewichteten Pfaden im klassischen TSP, da die Pfade ja keine Verbindung zwischen den Artikeln darstellen. Die Wahl der Pfade beeinflusst die weitere Reise bei der hier vorliegenden Aufgabenstellung nicht immer auf die gleiche Art und Weise. Es ist möglich, dass für einen Artikel ein anderes Parameterset gewählt wird, ohne dass sich Rückwirkungen auf alle nachfolgenden Artikel ergeben. Wird beim klassischen TSP an einem Knoten ein anderer Pfad gewählt, so hat dies unvermeidliche Auswirkungen auf die gesamte Reise.

### 4.3.3 Implementierung Nearest Neighbor

Die im Zuge dieser Arbeit entstandene Implementierung der Stufe 2 weist ausgeprägte Ähnlichkeiten zum in Kapitel 2.5 beschriebenen N-N-Ansatz beim Lösen des TSP auf.

Das Flussdiagramm aus Abbildung 4.9 ermöglicht einen groben Überblick, die Abläufe in Stufe 2 betreffend. Nach dem Parsen der Input-Dateien wird in einem ersten Durchgang die Auslastung für jede Produktionsressource in jedem Betrachtungsintervall errechnet. Bei diesem ersten Schritt wird das beste zur Verfügung stehende Parameterset (`<variante>`) für jeden Artikel verwendet. Sollte diese Auswahl zu keiner Überbelegung führen, ist Stufe 2 hiermit beendet und dieses Ergebnis wird in das Output-XML geschrieben (wird in 4.3.4 im Detail erklärt). Kommt es zu Überbelegungen der Ressourcen (`<anlage>`), so wird die Periode und Ressource betrachtet, in der die (eine) Überbuchung am ausgeprägtesten war. Jetzt werden alle Artikel betrachtet, die in dieser Periode auf dieser Ressource produziert werden sollen. Es wird für jeden Artikel die anteilige Auslastung auf dieser Maschine und dieser Periode berechnet und derjenige Artikel mit dem zur Überbelegung passenden oder signifikantesten Anteil ermittelt. Für diesen Artikel wird nun das nächstbeste Parameterset aktiviert. Die Berechnung der Gesamtbelegung beginnt nun von neuem. Sollte die Überbelegung nicht behoben sein, so wird der vorhin geschilderte Schritt so oft wiederholt, bis entweder eine verwendbare Kombination an Parametersets gefunden wurde, oder alle verfügbaren Parametersets durchlaufen wurden und mit einer Fehlermeldung abgebrochen werden mussten.

Diese Vorgehensweise garantiert nicht für ein kostenminimales Ergebnis. Es besteht sogar die Möglichkeit, dass mit einer Fehlermeldung abgebrochen wird, obwohl eine Lösung existiert. Es ergibt sich hier dieselbe Problematik wie beim Lösen des TSP mit der N-N-Heuristik.

Komplexere und somit in der Implementierung wesentlich anspruchsvollere algorithmische Lösungen für das Optimierungsproblem in Stufe 2, wurden im Rahmen dieser Arbeit nicht näher betrachtet. Für eventuelle Weiterentwicklungen der gegenständlichen Software ist dies allerdings einer der vielversprechendsten Ansatzpunkte.

### 4.3.4 Output Stufe 2

Das Resultat aus Stufe 2 wird ebenfalls in Form eines XML-Dokuments abgespeichert. Diese Datei wird anschließend von dem GUI importiert und die Ergebnisse in der Datenbank eingetragen. Abbildung 4.11 zeigt einen beispielhaften Auszug aus dem XML-Output von Stufe 2.

```
<S2_Result>
  <dataset>0001</dataset>
  <heuristik>NN</heuristik>
  <artikel>
    <artikelnummer>100001</artikelnummer>
    <id>1</id>
  </artikel>
  <artikel>
    <artikelnummer>100002</artikelnummer>
    <id>65</id>
  </artikel>
  <artikel>
    <artikelnummer>100003</artikelnummer>
    <id>72</id>
  </artikel>
  ...
</S2_Result>
```

Abbildung 4.11: Beispielhafter Output aus Stufe 2 (Auszug)

Das Wurzelement ist hier als `<S2_Result>` bezeichnet und beinhaltet die Child-Elemente `<dataset>`, `<heuristik>` und `<artikel>`. Das Element `<dataset>` ermöglicht wiederum die Zuordnung der Ergebnisse zu einem bestimmten Rechendurchgang. Der Inhalt von `<heuristik>` gibt, wie die Bezeichnung schon vermuten lässt, die bei der Berechnung in Stufe 2 verwendete Algorithmik an.

Das einzige mehrfach vorkommende Child von `<S2_Result>` ist `<artikel>`. Dieses enthält wiederum die Child-Elemente `<artikelnummer>` und `<id>`. Die Artikelnummer identifiziert den jeweiligen Artikel und ist in dem Output-XML aus Stufe 2 einzigartig. Das Child `<id>` enthält den Index des verwendeten Parametersets aus dem Puffer in den vorhergegangenen Berechnungen von Stufe 1. Der Inhalt von `<id>` ist immer dann 1, wenn es im Zuge der Berechnungen von Stufe 2 keine Notwendigkeit gab, für den Artikel einen vom Idealszenario abweichenden Produktionszeitpunkt bzw. eine Losgröße zu wählen.

### 4.3.5 Performance

Die Abschätzung der benötigten Rechenleistung in Stufe 2 ist bei weitem komplexer als in Stufe 1. Bei der Performance-Abschätzung von Stufe 1 in Kapitel 4.2.5 war die Abhängigkeit der benötigten Rechenleistung eingrenzbar auf einen Parameter. Es wurden zwar Annahmen getroffen, um weitere Abhängigkeiten einzugrenzen, allerdings konnte dennoch eine klare Verbindung zu der Parameterzahl als primären Einflussfaktor hergestellt werden.

Eine Abschätzung der benötigten Rechenleistung in Stufe 2 ist in vergleichbarer Art und Weise nicht so einfach zu realisieren. Die Vorgehensweise welche in Abbildung 4.9 dargestellt ist, lässt die Reduktion auf einige wenige einflussreiche Parameter nicht zu. Dies ist der Tatsache geschuldet, dass die Anzahl der Iterationen (Ausprobieren unterschiedlicher Parametersets) massiv von den Eingangsdaten abhängig ist. Im Verlauf der Versuche mit dem Softwareprototypen hat sich allerdings gezeigt, dass zumindest für die verwendeten Testdatensätze die Rechenzeit in Stufe 2 signifikant unter den Werten der Stufe 1 liegt. Konkret bedeutet dies, dass die benötigte Rechenzeit in einem Testlauf mit 1000 betrachteten Artikel, 10 Produktionsressourcen und einem Betrachtungszeitraum von einem Jahr zu über 99 % von der Heuristik in Stufe 1 beansprucht wurde. Aufgrund dieser Tatsache, wurde die Performance-Fragestellung (betreffend Rechenleistung) für Stufe 2 nicht weiter verfolgt.

Neben dem Thema Rechenaufwand lässt sich unter Performance natürlich auch die Qualität der Ergebnisse diskutieren. Wie schon in Kapitel 4.3.3 erwähnt, kann eine Implementierung der N-N-Heuristik nicht für die Güte des Ergebnisses garantieren. Es ist sogar möglich, dass kein Ergebnis gefunden wird, obwohl eine Lösung für die aktuelle Problemstellung existiert<sup>7</sup>. Es gibt andere algorithmische Ansätze<sup>8</sup>, welche im Stande sind, in Stufe 2 eine deutlich bessere Gesamtpformance zu liefern als die N-N-Heuristik. Aufgrund der Komplexitäten, die bei der Implementierung dieser Methoden zu erwarten sind, wurde dieser Schritt nicht im Rahmen der gegenständlichen Arbeit durchgeführt, aber für eine mögliche Weiterentwicklung des Projektes vorgemerkt.

---

<sup>7</sup>Vgl. <https://de.wikipedia.org/wiki/Nearest-Neighbor-Heuristik>

<sup>8</sup>Z. B.: Minimum-Spanning-Tree-Heuristik, Algorithmus von Christofides, künstliche neuronale Netze (Hopfield-Netz), ...

## 4.4 GUI

Für die Implementierung des GUI wurde ein auf Web-Technologien basierender Ansatz gewählt. Auf dem Host kommt der Web-Server Apache und die Datenbank MySQL zum Einsatz. Als serverseitige Scriptsprache wird PHP in der Version 5 verwendet<sup>9</sup>. Clientseitiges Scripting (z. B. JavaScript) wird nicht verwendet und kann für den Funktionsumfang dieser prototypischen Implementierung auch keinen zusätzlichen Nutzen erbringen.

In diesem Kapitel wird anhand mehrerer Screenshots von den jeweiligen Menüs die für den Anwender verfügbare Funktionalität der Software illustriert. Die Screenshots wurden auf dem Testsystem (siehe Kapitel 2.4) unter dem Betriebssystem Debian 7 und mit dem Browser Firefox erstellt. Aus Platzgründen wurden die üblicherweise sichtbaren grafischen Elemente des Betriebssystems und des Browser-Fensters abgeschnitten. Es ist in den Abbildungen 4.12 bis 4.23 somit nur das Rendering der HTML-Elemente sichtbar, welches vom Browser auf dem Bildschirm ausgegeben wird.

In diesen Screenshots findet sich auch das Akronym SOE, welches über dem Menü angezeigt wird. Es steht für *Strategy Optimization Environment*, um dem im Rahmen dieser Arbeit erstellten Softwarepaket auch einen Namen zu geben.

### 4.4.1 Hauptmenü

Abbildung 4.12 zeigt das Hauptmenü des GUI. Dieses wird direkt nach dem erfolgreichen Login eines Anwenders angezeigt.

---

<sup>9</sup>Siehe 1.5 Eintrag LAMP.

**SOE**

**Navigation**

- Hauptmenü
- Input
- Prognose
- Daten
- IBF
- N-N
- Auswertung
- Export
- Einstellungen
- Logout

**Hauptmenü**

Sie sind angemeldet als **Thomas Weiler**

**Aktuelle Berechnungen**

Dataset	Heuristik	Erstellt
<b>W23</b>	Nearest-Neighbor	24.05.2013
<b>W24</b>	Nearest-Neighbor	01.06.2013

**Aktuelle Ergebnisse**

Artikelnummer	Dataset	Strategie	Fertiggestellt
<b>0812</b>	W23	Plan2	23.05.2013
<b>1089</b>	W23	Plan2	23.05.2013
<b>1135</b>	W23	sqStr	22.05.2013

**Serverstatus**

**214 IBF Dateien in der Warteschlange.**  
**2 N-N Dateien in der Warteschlange.**

18 angelegte Datasets.  
 1200 angelegte Artikel.  
 3 gespeicherte Strategien.

Abbildung 4.12: Screenshot – Hauptmenü

Neben der Möglichkeit mittels des links angezeigten Navigations-Fensters durch die gesamte Applikation zu steuern, bietet das Hauptmenü einen Überblick betreffend aller laufenden Berechnungen und eventuell fertig gewordener Ergebnisse. Im Fenster *Aktuelle Berechnungen* werden die Datasets angezeigt, welche gerade von einer Konsolenapplikation ( Stufe 1 oder 2) bearbeitet werden. Unter *Aktuelle Ergebnisse* sind alle Resultate der Konsolenapplikationen gelistet, beginnend mit dem jüngsten Ergebnis. Das Fenster *Serverstatus* zeigt die Anzahl an Dateien in den Input-Verzeichnissen von Stufe 1 und 2 und somit die Menge an Datasets, welche noch abgearbeitet werden muss. Diese Anzahl lässt sich auch als die Länge einer Warteschlange interpretieren, deren Elemente die einzelnen Rechenjobs für die jeweiligen Konsolenapplikationen sind. Die Angaben *Angelegte Datasets*, *Artikel*, *Strategien* sollen dem Anwender einen groben Überblick über den Umfang der vorhandenen Stammdaten verschaffen.



## 4.4.2 Input

Aufgrund der Tatsache, dass der hier vorgestellte Software-Prototyp ein Stand-Alone-System ist, müssen die benötigten Stammdaten in die verwendete MySQL-Datenbank importiert werden. Die Abbildungen 4.13 und 4.14<sup>10</sup> zeigen den Teil des GUI, welches den Import aller benötigten Informationen aus CSV-Dateien ermöglicht.

The screenshot shows the SOE (Software-Operatives Environment) GUI. On the left is a navigation menu with buttons for 'Startseite', 'Input', 'Prognose', 'Daten', 'IBF', 'N-N', 'Auswertung', 'Export', and 'Einstellungen'. The main window is titled 'Input - Stammdaten' and contains the following elements:

- A header bar: 'Artikelstammdaten'
- Text: 'Hier können Sie Daten im csv Format aus dem ERP System importieren.'
- Form fields: 'Bitte wählen Sie die .csv Datei aus in der die Artikel-Daten gespeichert sind:' with a text input and a 'Durchsuchen...' button; 'Trennzeichen:' with a text input.
- Table for column mapping:

Feld	Spalte	Feld	Spalte
Artikelnummer	<input type="text"/>	Bezeichnung	<input type="text"/>
Rüstzeit 1	<input type="text"/>	Bearbeitungszeit 1	<input type="text"/>
Rüstzeit 2	<input type="text"/>	Bearbeitungszeit 2	<input type="text"/>
Rüstzeit 3	<input type="text"/>	Bearbeitungszeit 3	<input type="text"/>
Anlage 1	<input type="text"/>	Anlage 2	<input type="text"/>
Anlage 3	<input type="text"/>	Kosten Einlagerung	<input type="text"/>
Lagerkosten pro Periode (Woche)	<input type="text"/>	Kosten Auslagerung	<input type="text"/>
Gewicht	<input type="text"/>	Fehlkosten	<input type="text"/>

At the bottom, there is a section 'Wählen Sie die Importoptionen:' with a dropdown menu set to 'Überschreiben und einfügen' and an 'Import starten' button.

Abbildung 4.13: Screenshot – Input 1

Wie die Bezeichnung schon vermuten lässt, ermöglicht die erste Sektion *Artikelstammdaten* den Import aller artikelbezogenen Daten. Es wird die entsprechende CSV-Datei ausgewählt und auch das verwendete Trennzeichen angegeben. Anschließend müssen in den weiteren Feldern die erwarteten Variablen der jeweiligen Spalte in der CSV-Datei zugeordnet werden. Hier wird eine Einschränkung des Prototypen sichtbar, denn es können nicht beliebig viele Produktionsressourcen zugeordnet werden, sondern nur

<sup>10</sup>Hier wurde ein Fenster aus Platzgründen auf zwei Abbildungen aufgeteilt.

maximal drei. Beim Klick auf die Schaltfläche *Import starten* werden die Daten in die MySQL-Datenbank geschrieben. Hierbei kann noch der Einfügemodus gewählt werden, welcher das Verhalten bei bereits vorhandenen Artikeln beeinflusst (überschreiben oder überspringen).

The screenshot shows two identical-looking import forms. The top form is titled "Lagerbuchungen - Bedarfsverlauf" and the bottom one is "Betriebsmittel - Anlagen - Maschinen".

**Lagerbuchungen - Bedarfsverlauf:**

- Bitte wählen Sie die .csv Datei aus in der die gewünschten Daten gespeichert sind:  - Trennzeichen:
- Table mapping:
 

Feld	Spalte	Feld	Spalte
Artikelnummer	<input type="text"/>	Buchung	<input type="text"/>
Datum	<input type="text"/>		
- Wählen Sie die Importoptionen:

**Betriebsmittel - Anlagen - Maschinen:**

- Bitte wählen Sie die .csv Datei aus in der die gewünschten Daten gespeichert sind:  - Trennzeichen:
- Table mapping:
 

Feld	Spalte	Feld	Spalte
Inventarnummer	<input type="text"/>	Bezeichnung	<input type="text"/>
Rüstzeitkosten	<input type="text"/>	Bearbeitungskosten	<input type="text"/>
Kapazität pro Periode	<input type="text"/>		
- Wählen Sie die Importoptionen:

Abbildung 4.14: Screenshot – Input 2

In dem Abschnitt *Lagerbuchungen – Bedarfsverlauf* können für jeden Artikel die Lagerbewegungen importiert werden. Hierbei kann es sich um Warenflüsse aus der Vergangenheit handeln, aber auch um Bedarfsprognosen, die als Basis für die Produktionsplanung der nächsten Betrachtungsperiode dienen. Negative Werte in der Spalte *Buchungen* bedeuten einen Lagerabfluss und somit einen Bedarf in der angegebenen

Menge, während ein positiver Wert einen Lagerzufluss und somit ein produziertes Los in der genannten Menge angibt.

Die Sektion *Betriebsmittel – Anlagen – Maschinen* ermöglicht den Import der Daten, die für die Berücksichtigung der Produktionsressourcen bei der Berechnung der Gesamtkosten in Stufe 1 (siehe Kapitel 4.2.1) und für die Ermittlung der Anlagenbelegung in Stufe 2 (siehe Kapitel 4.3.1) notwendig sind.

### 4.4.3 Prognose

Die Ermittlung der Losgröße fußt letztendlich zu einem gewissen Grad auf Erwartungen betreffend die zukünftigen Bedarfe an den jeweiligen Artikeln. In den Abbildungen 4.15 und 4.16 werden die von der Software gebotenen Prognose-Möglichkeiten dargestellt. Diese Funktionalität bietet eine Extrapolation der Bedarfe, ausgehend von vergangenen Daten, für eine definierte Zeitspanne in die Zukunft an. Hier kann zwischen linearer Interpolation oder Polynomen vierten Grades gewählt werden, um die zukünftigen Bedarfsverläufe approximieren zu können.

Diese Art der Bedarfsprognose ist in dem hier dargestellten Zusammenhang rein exemplarisch zu verstehen. In einer Produktiv-Implementierung der Software, sind natürlich die wesentlich höher entwickelten Prognosefunktionen bestehender Systeme (z. B. ERP) zu verwenden.

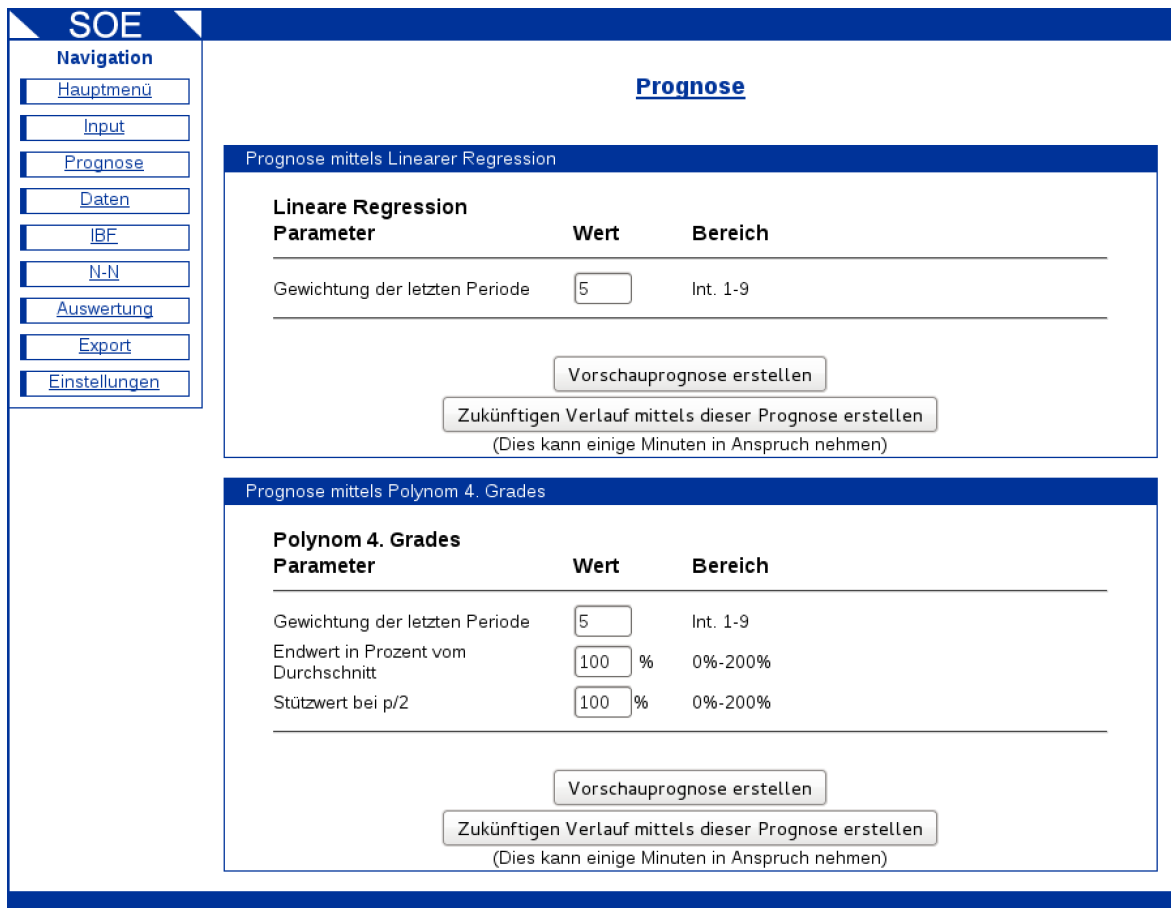


Abbildung 4.15: Screenshot – Prognose 1

Ein Klick auf die Schaltfläche *Vorschau* zeigt das Ergebnis der eingestellten Parameter an dem ersten Artikel aus der Datenbank. Abbildung 4.16 zeigt die Ausgabe eben dieser Vorschau. Der Klick auf die Schaltfläche *Zukünftigen Verlauf mit dieser Prognose erstellen* wendet das gewählte Extrapolationsverfahren auf alle Artikel des aktuellen Datasets <sup>11</sup> im definierten Betrachtungszeitraum an <sup>12</sup>.

<sup>11</sup>Siehe Kapitel 4.4.5.

<sup>12</sup>Vorausgesetzt das gewählte Enddatum liegt in der Zukunft und das gewählte Startdatum in der Vergangenheit, da sonst keine Extrapolation für die zukünftigen Bedarfe möglich ist.

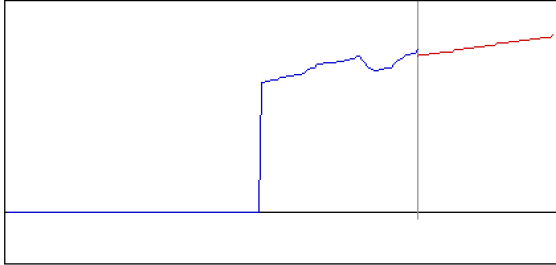
**SOE**

**Navigation**

- Startseite
- Input
- Prognose
- Daten
- IBF-Simulation
- TSP-Kalkulation
- Auswertung
- Export
- Einstellungen

## Prognose

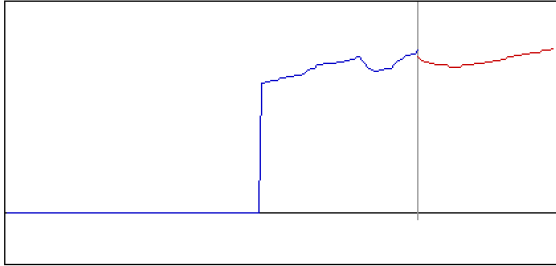
**Prognose mittels Linearer Regression**



Lineare Regression Parameter	Wert	Bereich
Gewichtung der letzten Periode	<input type="text"/>	Int. 1-9

(Dies kann einige Minuten in Anspruch nehmen)

**Prognose mittels Polynom 4. Grades**



Polynom 4. Grades Parameter	Wert	Bereich
Gewichtung der letzten Periode	<input type="text" value="5"/>	Int. 1-9
Endwert in Prozent vom Durchschnitt	<input type="text" value="100"/> %	0%-200%
Stützwert bei p/2	<input type="text" value="100"/> %	0%-200%

(Dies kann einige Minuten in Anspruch nehmen)

Abbildung 4.16: Screenshot – Prognose 2

## 4.4.4 Daten

Im Menü *Artikel-Daten* können nun nochmals alle artikelbezogenen Daten, die durch den Import aus CSV-Dateien und eventuell durch die Anwendung von den in Kapitel 4.4.3 beschriebenen Verfahren in die Datenbank eingetragen wurden, betrachtet und auch editiert werden.

**SOE**

**Navigation**

- Hauptmenü
- Input
- Prognose
- Daten
- IBF
- N-N
- Auswertung
- Export
- Einstellungen

**Artikel-Daten**

**Artikel suchen**

Artikelnummer:

**Artikel Daten**

Artikelnummer	Bezeichnung	erstellt am	Bearbeiten
0001	Testartikel_1	03.05.2013	⊘
0002	Testartikel_2	03.05.2013	⊘
0003	Testartikel_3	03.05.2013	⊘
0004	Testartikel_4	03.05.2013	⊘
0005	Testartikel_5	03.05.2013	⊘
0006	Testartikel_6	03.05.2013	⊘
0007	Testartikel_7	03.05.2013	⊘
0008	Testartikel_8	03.05.2013	⊘
0009	Testartikel_9	03.05.2013	⊘
0010	Testartikel_10	03.05.2013	⊘

[←](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [⇒](#)

**Artikel bearbeiten**

Hier haben Sie die Möglichkeit die einzelnen Datensätze manuell zu bearbeiten.

<b>Artikelnummer</b>	<input type="text"/>	<b>Bezeichnung</b>	<input type="text"/>
Rüstzeit 1	<input type="text"/>	Bearbeitungszeit 1	<input type="text"/>
Rüstzeit 2	<input type="text"/>	Bearbeitungszeit 2	<input type="text"/>
Rüstzeit 3	<input type="text"/>	Bearbeitungszeit 3	<input type="text"/>
Anlage 1	<input type="text"/>	Anlage 2	<input type="text"/>
Anlage 3	<input type="text"/>	Kosten Einlagerung	<input type="text"/>
Lagerkosten	<input type="text"/>	Kosten Auslagerung	<input type="text"/>
Gewicht	<input type="text"/>	Fehlkosten	<input type="text"/>

Abbildung 4.17: Screenshot – Daten

In der Sektion *Artikel suchen* lässt sich durch Eingabe der Artikelnummer der

zugehörige Artikel auffinden und direkt anzeigen. Im Bereich *Artikel Daten* wird eine Auflistung aller in der Datenbank verfügbaren Artikel angezeigt. Der Anwender hat hier die Möglichkeit, die Datensätze durchzublättern und eventuell durch Klick auf das Stift-Symbol, in den Bereich *Artikel bearbeiten* zu laden. Hier werden alle Felder des Artikels angezeigt und können auf Wunsch auch editiert werden. Die Schaltfläche *Speichern* triggert den Schreibvorgang in die Datenbank.

#### 4.4.5 IBF (Stufe 1)

In Abbildung 4.18 wird das Menü zum Erstellen eines Datasets für Stufe 1 dargestellt. In der Sektion *Betrachtungszeitraum* kann der Anwender mit dem Start und Enddatum das gewünschte zeitliche Intervall festlegen, mit dem sich die Berechnungen befassen sollen. Anschließend lassen sich in der Sektion *Artikel Auswahl* die zu betrachtenden Artikel selektieren. Letztendlich gilt noch zu definieren, welche Dispositionsstrategie bei der Simulation (Kostenberechnung) <sup>13</sup> verwendet werden soll.

---

<sup>13</sup>Siehe Kapitel 3.1.





sechs: drei Zeitpunkte und die jeweilige Losgröße. Das Ergebnis von Stufe 1 wird also für den gewählten Betrachtungszeitraum eine Grobproduktionsplanung für alle berücksichtigten Artikel generieren. Es wird der Zeitpunkt und die Losgröße der Produktionsaufträge ausgegeben, mit der Einschränkung, dass zu maximal drei unterschiedlichen Zeitpunkten produziert werden kann. Daher rührt auch die Bezeichnung dieser Strategie.

- **Planstrategie-2-Punkt:** Diese Strategie ist äquivalent zu *Planstrategie-3-Punkt* mit der Einschränkung, dass hier nur vier Parameter verwendet werden: Zwei Zeitpunkte mit ihren dazugehörigen Losgrößen.
- **s-Q-Strategie:** Bei dieser Strategie gibt es nur zwei Parameter. Q steht für die Losgröße, welche bei einem Produktionsauftrag verwendet wird, s gibt den Meldebestand an, bei dem der Produktionsauftrag ausgelöst wird. Im Unterschied zu den beiden Planstrategien, kann hierbei aus dem Output von Stufe 1 keine direkter Grobproduktionsplan für den Betrachtungszeitraum abgeleitet werden, da die Zeitpunkte erst im zeitlichen Verlauf bekannt werden.

Der Klick auf *Dataset für Simulation erstellen* führt zum Export der gewählten Daten in eine XML-Datei (siehe Abbildung 4.3).

#### 4.4.6 Nearest-Neighbor (Stufe 2)

Die Ergebnisse aus einem Durchlauf der Stufe 1 bilden die Datasets für die Stufe 2 der Heuristik. Abbildung 4.19 zeigt das Menü, mit dem die Datasets für Stufe 2 generiert werden können.

Hierbei kann in der Sektion *Dataset* das Resultat<sup>15</sup> von Stufe 1 ausgewählt werden, für welches die Kapazitätsprüfung und wenn nötig Anpassungen durchgeführt werden soll.

---

<sup>15</sup>Hierbei handelt es sich um Parametersets für eine definierte Menge von Artikel in einem definierten Betrachtungszeitraum, welche pro Artikel aufsteigend nach den Gesamtkosten sortiert sind.



Abbildung 4.19: Screenshot – Stufe 2

Im Bereich *Algorithmus auswählen* kann noch zwischen einem exakten Verfahren und der Nearest-Neighbor-Implementierung gewählt werden. Das exakte Verfahren, bei dem jede mögliche Kombination ausprobiert wird, ist nur für Testzwecke verfügbar. Es können auf diese Art und Weise maximal 10 Artikel betrachtet werden, um die notwendige Rechenzeit im vertretbaren<sup>16</sup> Rahmen zu halten.

#### 4.4.7 Auswertung

Der Menüpunkt *Auswertung*, der in den Abbildungen 4.20 und 4.21 dargestellt ist, gibt dem Anwender die Möglichkeit, alle verursachten Kosten eines gewählten Datasets anzeigen zu lassen.

Wird im Bereich *Dataset* eine Auswahl getroffen, so wird der Erstellungszeitpunkt und das Beginn- und Enddatum des Betrachtungszeitraums direkt bei dem Auswahlfeld angezeigt. Die von dem Dataset verursachten Kosten werden im Falle von Stufe 1 im Bereich *Minimalkosten* dargestellt. Die hier angezeigten Kosten errechnen sich aus der besten für jeden Artikel ermittelten Parameterkombination in diesem Dataset.

<sup>16</sup>Als vertretbar werden hier maximal 24 Stunden bezeichnet.

**SOE**

**Navigation**

- Hauptmenü
- Input
- Prognose
- Daten
- IBF
- N-N
- Auswertung
- Export
- Einstellungen

**Intelligent Brute-Force (Stufe 1) - Auswertung**

**Dataset**

**Dataset:** Dataset erstellt am Artikel  Erstellt am: 01.05.2013  
 Von: 01.01.2013 Bis: 31.12.2013

**Minimalkostenkombination**

<b>Rüstkosten</b>	<b>15.328,16 €</b>
<b>Bearbeitungskosten</b>	<b>82.321,43 €</b>
<b>Lagerkosten</b>	<b>14.961,72 €</b>
<b>Fehlkosten</b>	<b>420,00 €</b>
<b>Gesamtkosten</b>	<b>113.031,31 €</b>

Abbildung 4.20: Screenshot – Auswertung Stufe 1

Falls die Produktionskapazitäten mitberücksichtigt werden sollen, muss Stufe 2 durchlaufen werden. In Abbildung 4.21 findet sich im Bereich *Beste ermittelte kapazitiv abgestimmte Kombination* die Ausgabe der Gesamtkosten des gewählten Datasets nach den Berechnungen von Stufe 2.

**SOE**

**Navigation**

- Hauptmenü
- Input
- Prognose
- Daten
- IBF
- N-N
- Auswertung
- Export
- Einstellungen

**Nearest-Neighbor (Stufe 2) - Auswertung**

**Dataset**

**Dataset:** Dataset erstellt am Artikel  Erstellt am: 02.05.2013  
 Von: 01.01.2013 Bis: 31.12.2013

**Beste ermittelte kapazitiv abgestimmte Kombination**

<b>Rüstkosten</b>	<b>16.103,58 €</b>
<b>Bearbeitungskosten</b>	<b>82.321,43 €</b>
<b>Lagerkosten</b>	<b>16.015,94 €</b>
<b>Fehlkosten</b>	<b>210,00 €</b>
<b>Gesamtkosten</b>	<b>114.650,95 €</b>

Abbildung 4.21: Screenshot – Auswertung Stufe 2

## 4.4.8 Export

Wie schon in Kapitel 4.4.2 erläutert, verfügt die Software im prototypischen Zustand nicht über eine direkte Anbindung zu einem übergelagerten System<sup>17</sup>. Um nun die in den beiden Stufen errechneten Ergebnisse auch außerhalb der Software betrachten bzw. verwenden zu können, wurde eine Export-Funktion integriert. Abbildung 4.22 zeigt das hierfür notwendige Menü.

The screenshot shows the 'SOE' software interface. On the left is a navigation sidebar with buttons for 'Hauptmenü', 'Input', 'Prognose', 'Daten', 'IBF', 'N-N', 'Auswertung', 'Export', and 'Einstellungen'. The main content area is titled 'Export' and contains the instruction 'Hier können Sie Ergebnisse im csv Format exportieren.' Below this are two sections: 'IBF (Stufe 1)' and 'N-N (Stufe 2)'. Each section includes a text input field for the directory, a 'Durchsuchen...' button, a 'Trennzeichen:' input field, a dropdown menu for 'Wählen Sie das Dataset:' (currently showing '18'), and an 'Export starten' button.

Abbildung 4.22: Screenshot – Export

Es können die Ergebnisse von beiden Stufen der Heuristik separat in Form von CSV-Dateien exportiert werden. Hierfür muss das gewünschte Verzeichnis angegeben werden und das Trennzeichen, welches in den Dateien verwendet werden soll. Natürlich ist es auch hier notwendig das Dataset auszuwählen, um dann via Klick auf *Export starten* die Daten aus der internen MySQL-Datenbank in die gewünschte CSV-Datei zu schreiben.

## 4.4.9 Einstellungen

Die notwendigen Einstellungen für das im Rahmen dieser Arbeit entwickelte Softwarepaket, können in dem in Abbildung 4.23 dargestellten Menü getroffen werden.

<sup>17</sup>Datenbanken mit den notwendigen Informationen: ERP, MES, ...

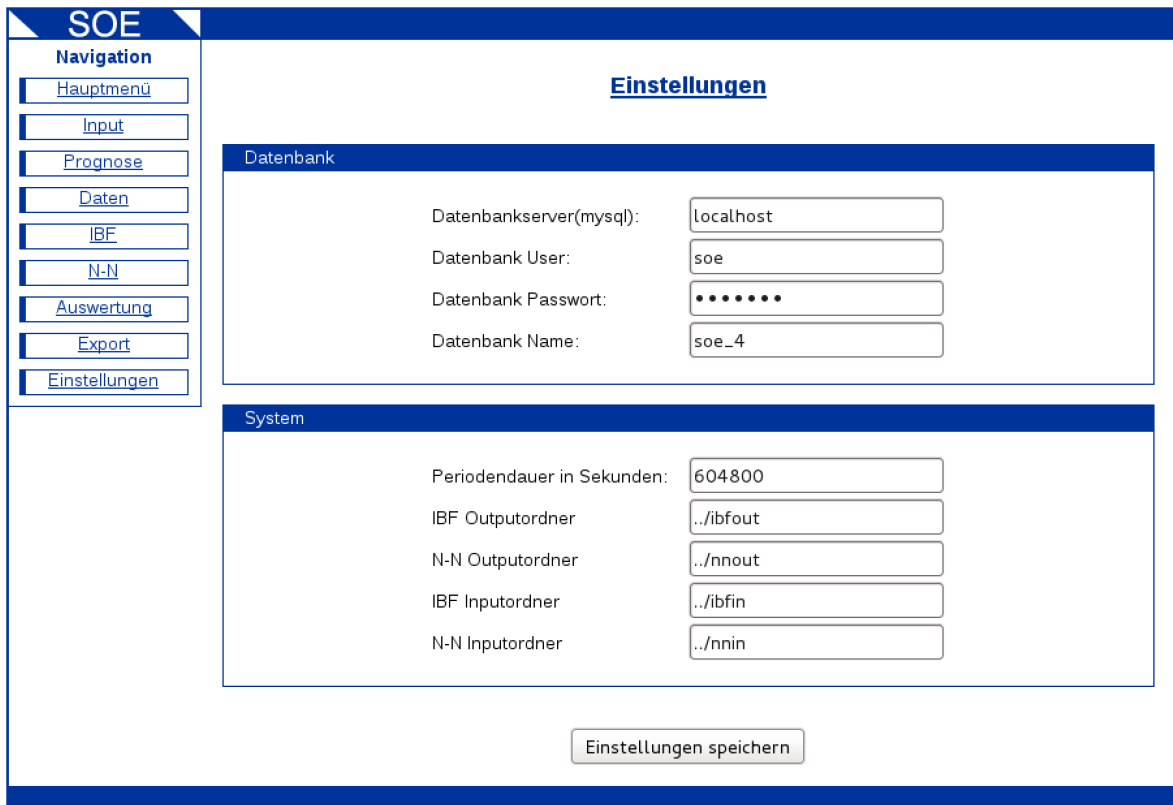


Abbildung 4.23: Screenshot – Einstellungen

Die Sektion *Datenbank* beinhaltet alle Informationen, die für eine Verbindung zur verwendeten MySQL-Datenbank notwendig sind.

Im Bereich *System* wird die Dauer einer Periode in Sekunden festgelegt. Auch die Input- und Output-Verzeichnisse der Konsolenprogramme für Stufe 1 und 2 können hier eingestellt werden. Insbesondere die hier konfigurierbare Periodendauer ist von großer Bedeutung für das System. Der eingestellte Wert in Sekunden gibt das Diskretisierungsintervall bei der Eingabe eines Betrachtungszeitraums an. Der in der Abbildung verwendete Wert von 604800 Sekunden entspricht einer Woche ( $7t \cdot 24h \cdot 60min \cdot 60s = 604800s$ ). Wird also ein Betrachtungszeitraum von 6 Monaten für ein Dataset gewählt, so führt dies zu einer Diskretisierung der Zeitachse in 26 Intervalle mit einer jeweiligen Dauer von eben diesen 604800 Sekunden.



## Ergebnisse

Im Rahmen dieser Arbeit wurde eine Herangehensweise an die Losgrößenproblematik vorgestellt, welche sich auf rechenintensive Heuristiken stützt. Die praktische Umsetzung dieser Idee in einen lauffähigen Software-Prototypen stellt den Großteil dieser Arbeit dar. Das somit entstandene Softwarepaket lässt sich als informationstechnologische Diskussionsgrundlage interpretieren, um die zu Grunde liegenden Gedanken auf Anwendbarkeit testen zu können. Die Software ist somit ein wesentliches Ergebnis dieser Arbeit.

Die Funktionsweise von Stufe 1 der Heuristik (siehe Kapitel 3.1) kann prinzipbedingt mit beliebigen Verläufen der Bedarfe und Kostensätze umgehen. Durch das Simulieren aller Materialbewegungen im Betrachtungszeitraum, um die Kosten für das jeweilige Parameterset zu ermitteln, können alle möglichen Variationen der Kostensätze und Material-Bedarfe mitberücksichtigt werden. In Stufe 1 kann bei der Implementierung des Systems außerdem jede beliebige Dispositionsstrategie nachgebildet werden. Die real stattfindenden Materialbewegungen können somit in der Simulation, welche zur Kostenermittlung herangezogen wird, bestmöglich nachgebildet werden.

Eines der augenscheinlichsten Ergebnisse dieser Arbeit ist die nun mögliche Abschätzbarkeit der notwendigen Schritte bei der Implementierung des thematisierten Losgrößen-Automatismus. Im Rahmen dieser Arbeit wurde ein auf Webtechnologien basierendes GUI (Kapitel 4.4) entwickelt, die beschriebenen Heuristiken (siehe Kapitel 3.1 und 3.2) wurden in Form von Konsolenapplikationen implementiert und diese Komponenten anschließend zu einem lauffähigen prototypischen Gesamtsystem

zusammengefügt. All diese Schritte wurden mit dem für eine Diplomarbeit üblichen Zeitaufwand bewältigt. Dies ermöglicht einem erfahrenen Softwareentwickler eine ungefähre Einschätzung betreffend der notwendigen Ressourcen für die Entwicklung einer professionell einsetzbaren Beat-Version dieses Systems.

Die Kenntnis der technologischen Barrieren, welche sich primär durch die Eigenschaften der Heuristik in Stufe 1 erklären, sind ein weiterer wesentlicher Output dieser Arbeit. Ob der Einsatz des hier vorgeschlagenen Prozederes überhaupt eine Chance auf ein verwertbares Ergebnis mit angemessenem Rechenaufwand hat, lässt sich anhand von Formel 3.1 und den Überlegungen in Kapitel 4.2.5 errechnen. Das exponentielle Wachstum des Rechenaufwandes, welches aus der Hinzunahme von zusätzlichen Parametern resultiert, zeigt eine vergleichsweise harte Grenze für die Beherrschbarkeit von komplexen Dispositionsstrategien.

Die in Kapitel 1.3 gesetzten Ziele für einen Losgrößen-Automatismus, konnten während der Implementierung der Software berücksichtigt werden. Die Frage nach dem Erreichen der in 1.4 beschriebenen Anforderungen für ein Softwarepaket, welches bei der Suche nach der optimalen Losgröße behilflich sein soll, kann mit den bis dato durchgeführten Entwicklungsschritten nicht vollständig beantwortet werden<sup>1</sup>.

## 5.1 Anwendungsszenario

Um das derzeit prototypische Softwarepaket in einem produktiven Szenario zur Anwendung bringen zu können, muss noch vieles an Entwicklungsarbeit geleistet werden. Sollte es zu einer Implementierung in der Softwarelandschaft eines produzierenden Unternehmens kommen, dann können folgende Use-Cases erwartet werden.

### 5.1.1 Kontinuierliche Berechnung

Im produktiven Einsatz kann davon ausgegangen werden, dass beide Stufen den gewählten Betrachtungszeitraum für alle relevanten Artikel täglich berechnen. Dies bedeutet, dass jeden Tag ein aktualisiertes Dataset für den Betrachtungszeitraum vorliegt. Die Informationsenthüllung im Zeitverlauf<sup>2</sup> fließt also über die wiederholte Berechnung der

---

<sup>1</sup>Siehe Kapitel 6.

<sup>2</sup>Der Unterschied zwischen prognostizierten und tatsächlichen Bedarfen.



idealen<sup>3</sup> Parameter für die verwendeten Dispositionsstrategien ein. Der auf diese Weise gewonnene Wert für die Losgröße der einzelnen Artikel kann auf unterschiedliche Art und Weise verwendet werden:

- **Automatismus:** Die Produktionsaufträge werden basierend auf den Ergebnissen von Stufe 1 und 2 verwendet, um eine vollautomatische Grobplanung der Produktion durchzuführen. Die Disponenten beschäftigen sich nicht mehr mit der Grobplanung, sondern nur noch mit den Optimierungen, die der Feinplanung<sup>4</sup> zugeordnet werden können.
- **Parameter-Eintrag:** Werden Strategien verwendet, welche in der jeweilig vorhandenen Unternehmenssoftware abgebildet sind (s-Q, ...), dann können die hierfür ermittelten Parameter direkt in die jeweiligen Dispositionsmodule eingetragen werden.
- **Empfehlung:** Die Disponenten legen Produktionsaufträge über explizit hierfür vorgesehene Funktionen in den vorhandenen Systemen (ERP, ...) an. Es ist denkbar, dass die errechneten Werte für die Losgröße und den Produktionszeitpunkt dem Disponenten als Empfehlung angezeigt werden, wenn dieser einen neuen Produktionsauftrag anlegen möchte. Dies ist für einen ersten Test des Systems in einem produktiven Umfeld die zu bevorzugende Variante.

### 5.1.2 Analyse/Evaluierung

Das gegenständliche Softwarepaket lässt sich auch als Analyse-Werkzeug verwenden. Wird die Betrachtungsperiode komplett in der Vergangenheit gewählt, so sind bereits alle Geschehnisse und Werte bekannt. Es kann nun ein Vergleich gezogen werden zwischen den tatsächlich statt gefundenen Produktionsvorgängen und den von der Software errechneten Vorschlägen. Sollten sich hier nennenswerte Diskrepanzen zeigen, so kann die Untersuchung dieser Ungereimtheiten Aufschlüsse über diverse Fehlermodalitäten bieten. Angefangen von Abweichungen bei den verwendeten Kostensätzen, hin zu Fehlern in der verwendeten Algorithmik zur Simulation der Dispositionsstrategien und bis zu

---

<sup>3</sup>Sofern die Algorithmik im Stande ist, die idealen Parameter zu finden (Mängel in Stufe 2).

<sup>4</sup>Der Unterschied zwischen Grob- und Feinplanung ist hier über die gewählte Periodendauer definiert. Alle Schritte die in einer feineren zeitlichen Auflösung als die Periodendauer erlaubt betrachtet werden müssen, sind der Feinplanung zugeordnet.

Fehlern, die in der Dispositionsabteilung gemacht wurden. Dieser Use-Case lässt auch eine Evaluierung des Systems für einen potentiellen Anwendungsfall zu. Lässt die Analyse der Abweichungen auf eine Kosteneinsparung bei Verwendung des Systems schließen, so kann anhand der erwartbaren Einsparungen und der Implementierungskosten eine Entscheidungsbasis geschaffen werden. Der große Vorteil hierbei ist, dass für diesen Anwendungsfall keine direkte Integration in die bestehenden Systeme (ERP, MES, ...) notwendig ist und mit Export-/Import-Funktionen die notwendigen Daten bereitgestellt werden können.

### **5.1.3 Szenarien und deren Simulation**

Das hier diskutierte Softwarepaket lässt sich der Kategorie der logistischen Planungstools zuordnen. Dies ist zum einen der Tatsache geschuldet, dass es mit der Optimierung eines wesentlichen Parameters der innerbetrieblichen Logistik befasst ist, der Losgröße. Es gibt allerdings noch ein Einsatzszenario, welches diese Zuordnung rechtfertigt. Mit dem Softwarepaket ist es möglich, unterschiedliche Produktionsanlagen und deren logistische Eignung für das jeweilige Produktprogramm zu simulieren. Da die vorhandenen Anlagenkapazitäten und die verdichteten Arbeitspläne als Datengrundlage für die Berechnungen dienen, können dadurch unterschiedliche Konstellationen bzw. Anlagen simuliert und anschließend evaluiert werden.

Die Software ist somit prinzipiell auch in der Lage, bei der Planung bzw. Umgestaltung von Produktionsanlagen als Simulationswerkzeug eingesetzt zu werden. Als nutzbare Outputs aus dem System sind ja nicht nur die Losgröße und der Zeitpunkt für die jeweiligen Produktionsaufträge zu nennen, sondern auch die aufgeschlüsselten Kosten<sup>5</sup> für die Bereitstellung der einzelnen Erzeugnisse.

## **5.2 Problemfelder**

Hier sollen alle im Verlauf dieser Arbeit entdeckten bzw. beschriebenen Schwachstellen der propagierten Vorgehensweise in kompakter Form nochmals einer Betrachtung unterzogen werden.

---

<sup>5</sup>Es werden für jeden Artikel im gewählten Betrachtungszeitraum Rüstkosten, Bearbeitungskosten, Lagerkosten, Kapitalbindungskosten und Fehlmengenkosten unterschieden und getrennt ausgewiesen.

### 5.2.1 Prognose

*Die Zukunft ist ungewiss.* Eine Behauptung, welche sich ohne Quellenangabe und trotzdem guten Gewissens in eine akademische Arbeit schreiben lässt.

Die Ermittlung der Losgrößen für die jeweiligen Erzeugnisse basiert auf prognostizierten Bedarfen. Die Möglichkeit dem tatsächlichen Optimum der Losgröße nahezukommen, ist von der Güte der dem Berechnungsverfahren zugrunde liegenden Bedarfsvorhersage abhängig. Die erreichbare Güte eben dieser Vorhersage ist wiederum in großem Maße durch die jeweilige Situation bedingt.

Für Erzeugnisse deren Nachfrage über entsprechend lange Betrachtungszeiträume annähernd konstant bleibt, kann man mit präzisen Prognosewerten rechnen. Auch Teile deren Bedarfe über die Auftragsbücher für einen relativ langen Zeitraum in die Zukunft bekannt sind, stellen keine größeren Probleme für die Losgrößen-Heuristik dar. Wirklich problematisch sind alle Erzeugnisse deren Nachfrage starken Schwankungen unterlegen ist und sich auch nicht aus den Auftragsbüchern ableiten lässt.

Die Güte der verwendeten Prognose stellt eine prinzipielle Grenze für die Erreichbarkeit des Losgrößen-Optimums dar.

### 5.2.2 Anbindung – Datenintegrität

Der produktive Einsatz der Software ist wohl kaum in der derzeitigen Realisation als Inselsystem möglich. Eine direkte Anbindung, ja vielleicht sogar Einbindung in die bestehenden Systeme ist unerlässlich für den professionellen Einsatz dieses Softwarepaketes. Derartige Bestrebungen werden allerdings durch diverse Hindernisse erschwert. In der nachfolgenden Auflistung werden einige kritische Punkte genannt.

- Redundanzen: Im aktuellen Entwicklungsstand der Software werden die benötigten Daten aus der Datenbank des Unternehmens extrahiert und als Kopie in der Losgrößenoptimierungs-Software verwendet. Es entsteht eine redundante Datenhaltung, welche prinzipiell bei Unternehmenssoftware äußerst unerwünscht ist. Die Tatsache, dass während der Berechnungen keine Änderungen an den Input-Daten zulässig sind, erfordert aber genau diese Redundanz<sup>6</sup>. Im Produktiveinsatz müssen Mechanismen implementiert sein, welche auf der einen Seite die Integrität der

---

<sup>6</sup>Die Berechnungen werden mit einer Momentaufnahme der benötigten Daten durchgeführt.

Inputdaten während der Berechnung sicher stellen und auf der anderen Seite mögliche Veränderungen, die während der Berechnungen auftreten, erkennen, um die Resultate gegebenenfalls zu verwerfen.

- **Haftungsfragen:** In der Regel existieren Wartungs- bzw. Serviceverträge zwischen den Anbietern von ERP-Systemen und den Unternehmen, welche diese Systeme einsetzen. Die Anbindung oder Integration der Software in ein bestehendes System ist somit als kritisch einzustufen, betreffend die in diesen Verträgen vereinbarten Gewährleistungen.
- **Heterogenität:** Die in den Unternehmen eingesetzten ERP- und MES-Lösungen weisen meist einen hohen Anpassungsgrad an die gegebenen Umstände auf. Dies bedeutet wiederum ein relativ heterogenes Umfeld, in welches die hier thematisierte Software eingebettet werden soll. Das Fehlen von standardisierten Schnittstellen lässt erhebliche Aufwände bei der Installation der Software in unterschiedlichen Unternehmen erwarten.

### 5.2.3 Exponentielles Wachstum

Das in Kapitel 4.2.5 detailliert beschriebene Verhalten von Stufe 1 bei der Erhöhung der Parameterzahl, stellt eine unüberwindbare Hürde dar, sofern mit der beherrschbaren<sup>7</sup> Parameterzahl kein Auskommen gefunden werden kann. Das exponentielle Wachstum der notwendigen Rechenschritte, welches aus Formel 3.1 ersichtlich wird, limitiert die Einsetzbarkeit der Heuristik in Stufe 1. Diese Grenze für die maximale Anzahl an sinnvoll verwendbaren Parametern, lässt sich kaum durch mehr Rechenleistung verrücken<sup>8</sup>.

Daraus lässt sich auch die folgende Einschätzung formulieren: Für Strategien die mit den heute verfügbaren Rechnern nicht in vertretbarer Zeit berechnet werden können, wird es wohl auch noch in Jahrzehnten nicht möglich sein, sie mit der in Stufe 1 implementierten Algorithmik zu lösen.

---

<sup>7</sup>Beherrschbar bedeutet in diesem Zusammenhang, dass der Rechenaufwand in einem angemessenen Rahmen bleibt.

<sup>8</sup>Unter der Annahme, dass die Anzahl an Diskretisierungsschritten konstant gehalten wird.

## 5.2.4 Ausnahmefälle

In Situationen, in denen keinerlei Regelmäßigkeiten bei den Bedarfen erkennbar sind, ist eine Prognose de facto nur durch Ableitung aus den Auftragsbüchern möglich. Hierbei kann man jedoch eigentlich nicht mehr von Prognose sprechen, da es sich in diesem Szenario um eine Bestellung mit einer gewissen Lieferzeit handelt. Jeder derartige Bedarf kann grundsätzlich als Ausnahmefall bezeichnet werden, da das Erzeugnis nur auf Bestellung produziert wird. Oft werden in solchen Fällen auch kundenspezifische Ausführungen an dem Erzeugnis vorgenommen.

In einem Szenario, in dem überwiegend kundenspezifisch produziert wird, ist die Frage nach der optimalen Losgröße obsolet. Es sei denn, es werden Lieferzeiten akzeptiert, welche deutlich über der Durchlaufzeit liegen. Dann kann durch die Aggregation von Kundenaufträgen wieder eine Losgrößenproblematik auftreten.

## 5.3 Entwicklungspotentiale

Ob die beschriebene Herangehensweise bei der Bestimmung der Losgröße in einem realen Anwendungsfall vorteilhaft ist, kann in diesem Entwicklungsstadium nicht eindeutig beantwortet werden. Es lassen sich jedoch schon jetzt gewisse Ansatzpunkte ausmachen, an denen sich die Performance des Systems positiv beeinflussen lassen könnte. Nachfolgend werden einige dieser Verbesserungsmöglichkeiten erwähnt.

### 5.3.1 Parallelisierung in Stufe 1

Der Algorithmus in Stufe 1 lässt sich sehr gut parallelisieren. Es ist möglich die unterschiedlichen Artikel eines Datasets gleichzeitig zu berechnen. Dies bedeutet die parallele Ausführung des Konsolenprogramms von Stufe 1, wobei es nun für jede Instanz ein eigenes Input-Verzeichnis gibt. Von der GUI werden nun die XML-Dateien, welche die Daten für die einzelnen Berechnungen beinhalten, gleichmäßig auf die Input-Verzeichnisse aufgeteilt. Die notwendige Rechenzeit für ein gesamtes Dataset ist umgekehrt proportional zu der Anzahl an laufenden Instanzen des Konsolenprogramms<sup>9</sup>. Prinzipiell ist es auch möglich, die Berechnungen betreffend einen Artikel zu parallelisieren. Hierzu ist eine

---

<sup>9</sup>Es handelt sich hierbei tatsächlich um einen annähernd linearen Zusammenhang!

korrekte Aufteilung der Start- und Endwerte des Parametersets für unterschiedliche Instanzen notwendig. Abbildung 4.4 verdeutlicht die Aufgabenstellung. Werden die Parameterkombinationen oberhalb der hervorgehobenen Zeile von einer Instanz berechnet und die Kombinationen unterhalb der Zeile von einer anderen, so wird dadurch die Rechenzeit halbiert.

Diese beiden Varianten der Parallelisierung führen zu annähernd linear wachsenden Einsparungen betreffend die Rechenzeit der Algorithmik in Stufe 1. Begrenzt sind diese Herangehensweisen einmal durch die Anzahl der betrachteten Artikel und zumindest prinzipiell durch die Anzahl an möglichen Kombinationen pro Artikel. Die praxisrelevante Einschränkung bei der Parallelisierung ist allerdings die Anzahl der verfügbaren Prozessorkerne, da jede Instanz des Konsolenprogramms nur dann einen Performancegewinn mit sich bringt, so sie auch einen physikalisch vorhandenen Prozessorkern exklusiv nutzen kann.

### **5.3.2 Alternative Algorithmik für Stufe 2**

Die in Stufe 2 verwendete Algorithmik ähnelt sehr dem N-N-Verfahren, wie schon in Kapitel 4.3 erläutert wurde. Wie bereits in 4.3.5 erwähnt, existieren etliche Algorithmen (Minimum-Spanning-Tree-Heuristik, Algorithmus von Christofides, künstliche neuronale Netze – Hopfield-Netz, ...), welche für die Aufgabenstellung in Stufe 2 wesentlich vielversprechender sind. Hierbei geht es primär jedoch nicht um die Rechenzeit, sondern hauptsächlich um die Qualität der zu erwartenden Ergebnisse.

### **5.3.3 Systemparameter Periodendauer**

Die Software arbeitet mit einer definierten Periodendauer, welche auch als Diskretisierungszeit verstanden werden muss. Es werden bei den Berechnungen alle Vorgänge in zeitliche Schritte unterteilt, deren Dauer eben durch diesen systemweit gültigen Parameter definiert ist. Eine kürzere Periodendauer bedeutet eine höhere zeitliche Auflösung der Abläufe im Unternehmen und somit potentiell präzisere Ergebnisse, allerdings auch einen höheren Rechenaufwand. Das Auffinden eines Optimums für diesen Parameter ist definitiv keine triviale Fragestellung und bedarf einer eigenen Untersuchung.

Es ist festzuhalten, dass Optimierungen der Software wesentlich gewichtiger sind,

als der von der Hardware-Entwicklung zu erwartende Performancegewinn. Dies liegt primär in dem exponentiellen Verhalten, welches in Kapitel 4.2 detailliert diskutiert wurde.





# KAPITEL 6

## Fazit

Die Fragestellung dieser Arbeit hat mir die Verknüpfung der Inhalte meines Studiums mit den Aufgaben aus meiner selbstständigen Tätigkeit als Softwareentwickler ermöglicht. Dieser interdisziplinäre Charakter stellte für mich den Anreiz dar, mich mit diesem Thema zum Abschluss meines Studiums zu befassen. Das Auffinden der optimalen Losgröße stellt in jedem Unternehmen eine Aufgabe dar, welche unterschiedliche Abteilungen beschäftigt (Produktion, Logistik, IT). Es handelt sich hierbei somit nicht nur um eine inhaltlich anspruchsvolle Aufgabenstellung, sondern auch um ein Thema, welches unternehmensintern ein gewisses Konfliktpotential aufweist.

Da sich die Losgrößen-Fragestellung in einem recht komplexen technologischen und sozialen Umfeld abspielt, habe ich für einen potentiell unterstützend wirkenden Automatismus eine äußerst simple Herangehensweise gewählt. Die resultierende Logik bzw. Funktionsweise der Software ist möglichst geradlinig gehalten, um für alle Beteiligten möglichst nachvollziehbar zu sein.

Die Frage nach der Wertigkeit des im Rahmen dieser Arbeit entwickelten Automatismus, ist mit den bis jetzt erfolgten Schritten nicht eindeutig beantwortbar. Ob die Einführung der entwickelten Software in einem Produktionsbetrieb zu einer relevanten Kostenersparnis führen kann, ist von sehr vielen Einflussgrößen abhängig und lässt sich wohl erst nach einem Real-Versuch beantworten. Hier findet sich natürlich auch ein großes Hemmnis für die weitere Entwicklung der in dieser Arbeit vorgestellten Heuristik. Um den praktischen Nutzen einschätzen zu können, ist eine Erprobung unter Realbedingungen unerlässlich. Gerade ein solcher Testlauf ist allerdings mit großem

Aufwand und mit gewissen Risiken verbunden. Einerseits muss das Softwarepaket aus dem Prototypen-Stadium hin zu einer Beta-Version entwickelt werden, andererseits muss ein Unternehmenspartner gefunden werden, welcher bereit ist, die notwendigen Ressourcen für die Implementierung und den anschließenden Testbetrieb zur Verfügung zu stellen.

Wie bei vielen Innovationen in der Produktionstechnik, lässt sich auch hier das Potential der im Rahmen dieser Arbeit geschilderten Vorgehensweise erst nach einer praktischen Erprobung einschätzen.

# Abkürzungen

**CPU** Central Processing Unit. 5, 16, 40, 42

**CSS** Cascading Style Sheets. 29

**CSV** Comma-Separated Values. 32, 61, 66, 72

**ERP** Enterprise Ressource Planning. 5, 6, 15, 32, 38, 63, 77, 78, 80

**GUI** Graphical User Interface. 30, 32, 33, 35, 37, 38, 43, 57, 59, 61

**HTML** Hypertext Markup Language. 29, 59

**IBF** Intelligent Brute Force. 48

**ID** Identifier. 37, 44

**KiB** Kibibyte. 40

**MES** Manufacturing Execution System. 5, 6, 15, 32, 38, 78, 80

**MiB** Mebibyte. 40

**N-N** Nearest-Neighbor. 17, 52, 56, 58, 82

**PHP** Hypertext Preprocessor. 30, 59

**TSP** Travelling Salesman Problem. 17, 18, 26, 55, 56

**XML** Extensible Markup Language. 32, 33, 35, 38, 39, 43–45, 51, 52, 54, 57, 69, 81, 88

# Abbildungsverzeichnis

1.1	Möglicher Einfluss einer Losgrößenvariation auf andere Parameter . . . . .	3
1.2	Zielkonflikt zwischen Lagerkosten und Rüstkosten beim Festlegen der Losgröße	4
2.1	Deterministische Modelle . . . . .	12
2.2	Stochastische Modelle . . . . .	12
2.3	Verlauf des Lagerstandes bei Andler-Harris . . . . .	13
2.4	Grobdarstellung der Ein- und Ausgabedaten der angestrebten Software . .	15
3.1	Brute-Force-Illustration mit drei Parametern: A,B,C. . . . .	21
3.2	Optimallösungen für drei Artikel (Beispiel) . . . . .	23
3.3	Überbelegung der Produktionskapazitäten (Beispiel) . . . . .	24
3.4	Lösungsmenge aus Stufe 1 der Heuristik (Beispiel) . . . . .	25
3.5	Maschinenbelegung nach Heuristik Stufe 2 (Beispiel) . . . . .	26
4.1	Struktur des Software-Prototypen . . . . .	31
4.2	Flussdiagramm von Stufe 1 der Heuristik . . . . .	34
4.3	Daten für Stufe 1 (XML) . . . . .	36
4.4	Schrittweises Erzeugen von Parametersets . . . . .	41
4.5	Quelltext der Funktion kombi zum Erzeugen der Parameterkombinationen	42
4.6	Beispielhafter Output aus Stufe 1 (Auszug) . . . . .	44
4.7	Mögliche Kombinationen in Abhängigkeit zur Parameter-Anzahl (50 Werte pro Parameter) . . . . .	46
4.8	Mögliche Kombinationen in Abhängigkeit zur Auflösung bei der Diskretisie- rung (festgehaltene Parameteranzahl) . . . . .	47
4.9	Flussdiagramm von Stufe 2 der Heuristik . . . . .	51
4.10	Input-Daten für Stufe 2 der Heuristik (Auszug aus XML-Datei) . . . . .	54
4.11	Beispielhafter Output aus Stufe 2 (Auszug) . . . . .	57
4.12	Screenshot – Hauptmenü . . . . .	60
4.13	Screenshot – Input 1 . . . . .	61
4.14	Screenshot – Input 2 . . . . .	62
4.15	Screenshot – Prognose 1 . . . . .	64
4.16	Screenshot – Prognose 2 . . . . .	65
4.17	Screenshot – Daten . . . . .	66
4.18	Screenshot – Stufe 1 . . . . .	68
4.19	Screenshot – Stufe 2 . . . . .	70
4.20	Screenshot – Auswertung Stufe 1 . . . . .	71
4.21	Screenshot – Auswertung Stufe 2 . . . . .	71
4.22	Screenshot – Export . . . . .	72
4.23	Screenshot – Einstellungen . . . . .	73

# Tabellenverzeichnis

2.1	Benötigte Parameter für die Berechnung der optimalen Losgröße nach Andler	14
3.1	Variablen für die Berechnung der Anzahl an Parameter-Kombinationen . .	22
4.1	Beispiel für den Speicherbedarf der Input-Daten von Stufe 1 pro Artikel . .	40
4.2	Wertebereiche einer Beispiel-Strategie mit vier Parametern . . . . .	49



# Literaturverzeichnis

- [1] F. FUCHS, H. M. u. K. T.: *MES – Grundlage der Produktion von morgen*. München : Oldenbourg Industrieverlag GmbH, 2008
- [2] GUDEHUS, T. : *Logistik 1, Grundlagen, Verfahren und Strategien*. 3. Auflage. Berlin/Heidelberg/New York : Springer-Verlag, 2007
- [3] H. KELLERER, U. P. u. D. P.: *Knapsack Problems*. Berlin/Heidelberg/New York : Springer-Verlag, 2004
- [4] LEXIKON: *Aktuelle Fachbegriffe aus Informatik und Kommunikation*. 9. Überarbeitete und aktualisierte Auflage. Hochschulverlag AG an der ETH Zürich, 2007
- [5] M. HESSELER, M. G.: *Basiswissen ERP-Systeme – Auswahl, Einführung und Einsatz betriebswirtschaftlicher Standardsoftware*. Herdecke Witten : W3L GmbH, 2007
- [6] M. J. BECKMANN, D. B.: *Lagerhaltung. Modelle und Methoden*. Berlin/Heidelberg/New York : Springer-Verlag, 1989
- [7] SCHULTE, G. : *Material- und Logistikmanagement*. 2. Auflage. München : Oldenbourg Wissenschaftsverlag GmbH, 2001