Forschungsberichte aus dem
wbk Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)

Alex Maximilian Frey

Datenbasierte Erstellung und Überprüfung von Modellen zur Produkt- und Prozesskonfiguration

Band 302

Forschungsberichte aus dem
wbk Institut für Produktionstechnik
Karlsruher Institut für Technologie (KIT)

Hrsg.: Prof. Dr.-Ing. Jürgen Fleischer
Prof. Dr.-Ing. Gisela Lanza
Prof. Dr.-Ing. habil. Volker Schulze
Prof. Dr.-Ing. Frederik Zanger

Alex Maximilian Frey

Datenbasierte Erstellung und Überprüfung von Modellen zur Produkt- und Prozesskonfiguration

Band 302

wbk
Institut für Produktionstechnik

Datenbasierte Erstellung und Überprüfung von Modellen zur Produkt- und Prozesskonfiguration

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des

Karlsruher Instituts für Technologie (KIT)

angenommene

Dissertation

von

Alex Maximilian Frey, M.Sc.

Tag der mündlichen Prüfung:	27.10.2025
Hauptreferentin:	Prof. Dr.-Ing. Gisela Lanza
Korreferent:	Prof. Dr.-Ing. Dieter Krause

KIT
Karlsruher Institut für Technologie

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: Karlsruhe, Karlsruher Institut für Technologie, Diss., 2025

Print-ISBN 978-3-8191-0504-3
PDF-ISBN 978-3-8191-0460-2
ISSN 2944-6430
eISSN 2944-6449
https://doi.org/10.2370/9783819104602

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Telefon: 02421 / 99 0 11 - 0 • Telefax: 02421 / 99 0 11 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

Vorwort des Herausgebers

Die schnelle und effiziente Umsetzung innovativer, nachhaltiger und wirtschaftlicher Technologien stellt den entscheidenden Wirtschaftsfaktor für produzierende Unternehmen dar. Universitäten können als "Wertschöpfungspartner" einen wesentlichen Beitrag zur Wettbewerbsfähigkeit der Industrie leisten, indem sie wissenschaftliche Grundlagen sowie neue Methoden und Technologien erarbeiten und aktiv den Umsetzungsprozess in die praktische Anwendung unterstützen.

Vor diesem Hintergrund wird im Rahmen dieser Schriftenreihe über aktuelle Forschungsergebnisse des Instituts für Produktionstechnik (wbk) am Karlsruher Institut für Technologie (KIT) berichtet. Unsere Forschungsarbeiten beschäftigen sich mit der Leistungssteigerung von additiven und subtraktiven Fertigungsverfahren, den Produktionsanlagen und der Prozessautomatisierung sowie mit der ganzheitlichen Betrachtung und Optimierung von Produktionssystemen und -netzwerken. Hierbei werden jeweils technologische wie auch organisatorische Aspekte betrachtet.

Prof. Dr.-Ing. Jürgen Fleischer

Prof. Dr.-Ing. Gisela Lanza

Prof. Dr.-Ing. habil. Volker Schulze

Prof. Dr.-Ing. Frederik Zanger

wbk
Institut für Produktionstechnik

Vorwort des Verfassers

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Produktionstechnik (wbk) des Karlsruher Instituts für Technologie (KIT).

Mein Dank gilt Frau Prof. Dr.-Ing. Gisela Lanza für die Betreuung meiner wissenschaftlichen Arbeit als Hauptreferentin, für das mir entgegengebrachte Vertrauen sowie für die fachliche und persönliche Unterstützung. Außerdem danke ich Herrn Prof. Dr.-Ing. Dieter Krause und Frau Prof. Dr.-Ing. Anne Meyer für ihr Interesse an meiner Arbeit und die Übernahme des Korreferats bzw. des Prüfungsvorsitzes. Dem Karlsruhe House of Young Scientists (KHYS) danke ich für die Förderung meines Forschungsaufenthalts an der Technischen Universität Dänemark (DTU). Mein Dank gilt ebenso Herrn Prof. Dr. Lars Hvam für die Möglichkeit dieses Forschungsaufenthaltes und den regelmäßigen Austausch, von dem ich sehr profitiert habe. Meinen ehemaligen Kolleginnen und Kollegen am wbk, insbesondere im Bereich Produktionssysteme, danke ich für die gute Zusammenarbeit und das freundschaftliche Miteinander. Besonders danke ich Louis Schäfer und Rainer Silbernagel für das Lektorat der vorliegenden Arbeit.

Darüber hinaus danke ich von Herzen meinen Eltern und meinen Geschwistern für ihre beständige Begleitung auf meinem Lebensweg, meinem Sohn Clemens, der mein Leben in besonderer Weise bereichert, und besonders meiner Frau Silvia für ihren unentbehrlichen Rückhalt und für die vielen schönen Momente unseres gemeinsamen Lebens – vergangene wie kommende.

Karlsruhe, im November 2025

Alex Maximilian Frey

KIT
Karlsruher Institut für Technologie

Abstract

Configuration systems enable industrial companies to automate and thus rationalize production process planning within the orderfulfilment process. Particularly great potential lies in the automatic configuration of bills of materials and routings, i.e., in product and process configuration. To date, configuration systems have not been implemented comprehensively for this purpose. The main obstacles are the effort required to create the configuration models that define the framework conditions and rules for configuration, as well as the high probability of errors in these models. These challenges can be overcome with data-driven methods such as machine learning techniques. These methods can be used to create models for product and process configuration based on past orders and to validate them through pattern recognition. This can contribute to the wider use of configuration systems in production process planning and thus to greater efficiency in order processing in industrial companies. However, according to the current state of research, data-driven methods in connection with configuration models have only been researched rudimentarily.

The aim of this thesis is therefore to lay the scientific foundations for the use of data-driven methods for creating and validating models for product and process configuration. It answers the questions of which methods are suitable for this purpose and how effective their use is. To this end, suitable methods are developed and evaluated from a technical perspective as part of a demonstration. This includes the data-based creation of the elements of industry-standard models for product and process configuration: super bills of materials, super routings, and dependencies in the form of rules between the variables of the models. In addition, the expansion of the data base and the data-driven validation of rules are considered. The demonstration of the developed methods shows that both the data-driven creation and the data-driven validation of product and process configuration models are fundamentally possible and in many cases lead to accurate results.

Kurzzusammenfassung

Konfigurationssysteme ermöglichen Industrieunternehmen die Automatisierung und damit Rationalisierung der Arbeitsablaufplanung im Rahmen des Auftragsabwicklungsprozesses. Großes Potenzial besteht hierbei insbesondere in der automatischen Konfiguration von Stücklisten und Arbeitsplänen, d. h. für die Produkt- und Prozesskonfiguration. Bisher werden Konfigurationssysteme hierfür nicht umfassend eingesetzt. Wesentliche Hinderungsgründe sind der Aufwand für die Erstellung sowie die hohe Fehleranfälligkeit der hinterlegten Konfigurationsmodelle, die die Rahmenbedingungen und Regeln der Konfiguration festlegen. Diesen Herausforderungen kann mit datenbasierten Methoden, wie z. B. Verfahren des maschinellen Lernens, begegnet werden. Hiermit können Modelle für die Produkt- und Prozesskonfiguration zum einen auf Basis zurückliegender Aufträge erstellt und zum anderen durch Mustererkennung überprüft werden. Dadurch kann ein Beitrag zu einem weitergehenden Einsatz von Konfigurationssystemen in der Arbeitsablaufplanung und damit zu einer höheren Effizienz des Auftragsabwicklungsprozesses in Industrieunternehmen geleistet werden. Nach Stand der Forschung sind jedoch datenbasierte Methoden im Zusammenhang mit Konfigurationsmodellen nur rudimentär erforscht.

Ziel der vorliegenden Arbeit ist es deshalb, die wissenschaftlichen Grundlagen für den Einsatz datenbasierter Methoden zur Erstellung und Überprüfung von Modellen für die Produkt- und Prozesskonfiguration zu legen. Es werden die Fragen beantwortet, welche Methoden hierfür geeignet sind und wie effektiv deren Einsatz ist. Hierfür werden geeignete Methoden entwickelt und im Rahmen einer Demonstration aus technischer Sicht bewertet. Dies umfasst die datenbasierte Erstellung der Bestandteile von industrieüblichen Modellen der Produkt- und Prozesskonfiguration: Maximalstücklisten, Maximalarbeitspläne sowie Abhängigkeiten in Form von Regeln zwischen den Variablen der Modelle. Darüber hinaus werden die Erweiterung der Datenbasis und die datenbasierte Überprüfung von Regeln betrachtet. Die Demonstration der entwickelten Methoden zeigt, dass sowohl die datenbasierte Erstellung, als auch die datenbasierte Überprüfung von Produkt- und Prozesskonfigurationsmodellen grundsätzlich möglich ist und in vielen Fällen zu genauen Ergebnissen führt.

Inhaltsverzeichnis

Abkürzungen

Abkürzung	Bedeutung
A	Anforderung
AVK	Arbeitsvorgangsklasse
AL	Aktives Lernen (engl. Active Learning)
ALS	Approximative Logiksynthese (engl. Approximate Logic Synthesis)
APL	Arbeitsplan
AVO	Arbeitsvorgang
BGK	Baugruppenklasse
CAPP	Computerunterstützte Prozessplanung (engl. Computer Aided Process Planning)
CG	Spaltengenerierung (engl. Column Generation)
CTO	Auftragsbezogene Konfiguration (engl. Configure to Order)
DNF	Disjunktive Normalform
DNN	Tiefes Neuronales Netz (engl. Deep Neural Network)
DP	Dualproblem
DSRP	Design Science Research Process
ERP	Enterprise Resource Planning
ETO	Auftragsbezogene Entwicklung (engl. Engineer to Order)
HLF	High-Level-Formel
HLKM	High-Level-Konfigurationsmodell
HLKS	High-Level-Konfigurationssystem
ILP	Ganzzahlige lineare Optimierung (engl. Integer Linear Programming)
KK	Komponentenklasse
KM	Konfigurationsmodell
KN	Klassennummer
KNF	Konjunktive Normalform
KS	Konfigurationssystem

LLKM	Low-Level-Konfigurationsmodell
LLKS	Low-Level-Konfigurationssystem
LP	Lineares Optimierungsproblem (engl. Linear Program)
MAPL	Maximalarbeitsplan
ML	Maschinelles Lernen (engl. Machine Learning)
MP	Master-Problem
MQS	Membership Query Synthesis
MSTL	Maximalstückliste
QBC	Query by Committee
RMP	Reduziertes Master-Problem
SAP	SAP SE
SAT	Erfüllbarkeit (engl. Satisfiability)
SL	Überwachtes Lernen (engl. Supervised Learning)
SP	Subproblem, auch Pricing-Problem genannt
STA	Strukturalternative
STO	Strukturoption
UL	Unüberwachtes Lernen (engl. Unsupervised Learning)
UML	Unified Modelling Language
VAPL	Variantenbezogener Arbeitsplan
VR	Versionenraum (engl. Version Space)
VSTL	Variantenbezogene Stückliste
XMP	Relaxiertes Master-Problem
XRMP	Relaxiertes reduziertes Master-Problem
ZK	Zukaufkomponente
ZKK	Zukaufkomponentenklasse

Formelzeichen

Kapitel 4.1

Formelzeichen	Typ	Bedeutung
AK_i	Klasse	AVK i in einem MAPL
KK_i^B	Klasse	KK einer Baugruppe i in einer MSTL
KK_i^Z	Klasse	KK einer ZK i in einer MSTL
p_i^{AkZu}	$\{0,1\}$	Aktivitätszustand einer KK i
p_i^{Bez}	String	Bezeichnung einer KK i
PK	Klasse	Klasse des Produkts in einer MSTL
p_i^{Me}	$\mathbb{N}$	Menge in der eine Komponente der KK i in ihre übergeordnete Komponente oder in das Produkt eingeht
$p_{i,j}^{Pa}$	fallabhängig	Allgemeiner Parameter j einer KK i
p_i^{Pos}	$\mathbb{N}$	Position einer KK i in einer MSTL
p_i^{STO}	Menge	Parameter einer KK i oder einer AVK i, der angibt für welche STOs diese instanziiert werden kann
$p_i^{STO,MAPL}$	$\mathbb{N}$	Gültige STOs des MAPL für eine KK i
$p_{PK}^{STO,MSTL}$	$\mathbb{N}$	Gültige STOs für die Konfiguration einer MSTL
$v_{i,j}$	fallabhängig	Eine mögliche Ausprägung j des Produktmerkmals i
$v_{i,j}^{Me}$	$\mathbb{N}$	Eine mögliche Ausprägung j von p_i^{Me}
$v_{i,j,k}^{Pa}$	fallabhängig	Eine mögliche Ausprägung k von $p_{i,j}^{Pa}$
x_i	fallabhängig	Ausprägung von Produktmerkmal i

Kapitel 4.2 und Anhang A3

Formelzeichen	Typ	Bedeutung
A^{GePf}	Matrix	Eine nullbasiert indizierte Matrix deren Einträge $A_{i,j}^{GePf}$ den Ergebnissen der Probleme $P_{i,j}^{GePf}$ entsprechen
b^{Min}	$\mathbb{N}$	Komplexität der komplexitätsminimalen MSTL unter allen bisher gefundenen MSTL in Algorithmus $Alg^{MinMSTL}$
b^{Ob}	$\mathbb{N}$	Obere Schranke für die Komplexität einer resultierenden MSTL in einem Knoten eines Suchbaums in Algorithmus $Alg^{MinMSTL}$

b^{Un}	$\mathbb{N}$	Untere Schranke für die Komplexität einer resultierenden MSTL in einem Knoten eines Suchbaums in Algorithmus Alg^{MinMSTL}
d^{MICD}	$[0,1]$	Mittlere Intraclusterdistanz
$d_{i,j}^{ZK}$	$[0,1]$	Normierte Distanz zweier ZKs i und j basierend auf ihrer Kontextähnlichkeit
I^{Bez}	Menge	Menge der Indizes von Bezeichnungen in S^{Bez}
I_l^{KBez}	Menge	Menge der Indizes von KKs der MSTL, die die Bezeichnung l aufweisen
I^{Pl}	Menge	Menge der Indizes von STO-Platzhaltern; es gilt $I^{Pl} = \{1, \dots, n^{Pl}\}$
I^{VSTL}	Menge	Menge der Indizes von VSTLs aus S^{VSTL}
I^{ZKK}	Menge	Menge der Indizes von ZKKs in einer Maximalstückliste
$I_k^{ZKK,VSTL}$	Menge	Menge der Indizes von ZKKs, die für die Konfiguration der variantenbezogenen Stückliste k aus der MSTL aktiv sind
n^{ClMax}	$\mathbb{N}$	Maximal mögliche Anzahl von Clustern für eine Clustering-Aufgabe
$n_{i,j}^{IdLa}$	$\mathbb{N}$	Anzahl der übereinstimmenden Labels in zwei Knoten i und j zweier Pfade
n_l^{MaxB}	$\mathbb{N}$	Maximale Anzahl von ZKs mit Bezeichnung l in einer VSTL aus einer Menge von VSTLs
n^{NiAn}	$\mathbb{N}$	Anzahl der noch nicht annotierten ZKs in einem Knoten eines Suchbaums in Algorithmus Alg^{MinMSTL}
$n_{i,k}^{PfLa}$	$\mathbb{N}$	Anzahl der Vorkommen eines Labels k im Pfad einer Komponente i zur Wurzel ihrer VSTL
n^{Pl}	$\mathbb{N}$	Anzahl aktivierbarer STO-Platzhalter
n^{PrKl}	$\mathbb{N}$	Prognostizierte Anzahl der Klassen in einer MSTL
n^{STLGr}	$\mathbb{N}$	Größe einer VSTL nach Anzahl der ZKs, die sie enthält
n_i^{KNBez}	$\mathbb{N}$	Anzahl unterschiedlicher KNs, mit denen ZKs mit Bezeichnung i in einem Knoten eines Suchbaums in Algorithmus Alg^{MinMSTL} annotiert wurden
$P_{i,j}^{GePf}$	Problem	Problem zur Bestimmung der Länge des längsten gemeinsamen Pfades aus zwei Pfaden wobei jeweils nur die ersten i bzw. j Knoten betrachtet werden
S^{Bez}	Menge	Menge aller Bezeichnungen von ZKs in einer Menge von VSTLs
$s_{i,j}^{GePf}$	$\mathbb{N}$	Länge des längsten gemeinsamen Pfades zweier ZKs i und j zu den Wurzeln ihrer jeweiligen VSTL

$\hat{s}_{i,j}^{GePf}$	$\mathbb{N}$	Maximal mögliche Länge eines längsten gemeinsamen Pfades zweier ZKs i und j zu den Wurzeln ihrer jeweiligen VSTL
S^{LaZK}	Menge	Menge aller Labels von ZKs in einer Menge von VSTLs
S^{VSTL}	Menge	Menge von VSTLs, die für die datenbasierte Erstellung einer MSTL verwendet werden
$s_{i,j}^{ZK}$	$[0,1]$	Normierte Kontextähnlichkeit zweier ZKs i und j
u_j^{PlAk}	$\{0,1\}$	Entscheidungsvariable, die angibt, ob ein STO-Platzhalter j aktiv ist (1) oder nicht (0)
$u_{k,j}^{VSTL,Pl}$	$\{0,1\}$	Entscheidungsvariable, die angibt, ob eine VSTL k einem STO-Platzhalter j zugeordnet ist (1) oder nicht (0)
$u_{i,j}^{ZKK,Pl}$	$\{0,1\}$	Entscheidungsvariable, die angibt, ob eine ZKK i der MSTL einem STO-Platzhalter j zugeordnet ist (1) oder nicht (0)

Kapitel 4.3

Formelzeichen	Typ	Bedeutung
$d_{i,j}^{AVO}$	$[0,1]$	Normierte Distanz zweier AVOs i und j basierend auf ihrer Kontextähnlichkeit
S_i^{E0}	Menge	Menge der nicht vorhandenen Kanten in einem VAPL i, die jeweils nicht vorhandene Vorrangbeziehungen darstellen
$S_i^{E0,L}$	Menge	Menge der nicht vorhandenen Kanten in einem VAPL i, die jeweils nicht vorhandene Vorrangbeziehungen darstellen, jeweils angegeben als Labels der inzidenten Knoten
$S^{E0,MAPL}$	Menge	Menge der nicht vorhandenen Kanten in einem MAPL, die jeweils nicht vorhandene Vorrangbeziehungen darstellen
$S^{E0,MAPL,L}$	Menge	Menge der nicht vorhandenen Kanten in einem MAPL, die jeweils nicht vorhandene Vorrangbeziehungen darstellen, jeweils angegeben als Labels der inzidenten Knoten
S_i^{E1}	Menge	Menge der Kanten in einem VAPL i, die jeweils Vorrangbeziehungen darstellen
$S_i^{E1,L}$	Menge	Menge der Kanten in einem VAPL i, die jeweils Vorrangbeziehungen darstellen, jeweils angegeben als Labels der inzidenten Knoten
$S^{E1,MAPL}$	Menge	Menge der Kanten in einem MAPL, die jeweils Vorrangbeziehungen darstellen

$S^{E1,MAPL,L}$	Menge	Menge der Kanten in einem MAPL, die jeweils Vorrangbeziehungen darstellen, jeweils angegeben als Labels der inzidenten Knoten
S_i^{Nach}	Menge	Menge der Labels von AVOs, die einem AVO i in einem VAPL nachfolgen
S_i^{OB}	Menge	Menge der Labels von AVOs, die zu einem AVO i in einem VAPL in keiner Beziehung stehen
S_i^{V}	Menge	Menge der Knoten in einem VAPL i, die jeweils AVOs darstellen
S^{VAPL}	Menge	Menge der VAPLs die für die datenbasierte Erstellung eines MAPL verwendet werden
S_i^{Vor}	Menge	Menge der Labels von AVOs, die einem AVO i in einem VAPL vorausgehen
$S_i^{V,L}$	Menge	Menge der Labels von Knoten in einem VAPL i, die jeweils AVOs darstellen
$S^{V,MAPL}$	Menge	Menge der Knoten in einem MAPL, die jeweils AVKs darstellen
$S^{V,MAPL,L}$	Menge	Menge der Labels von Knoten in einem MAPL, die jeweils AVKs darstellen

Kapitel 4.4 und Anhang A5

Formelzeichen	Typ	Bedeutung
a_{i,n^M+1}	$\{0,1\}$	Entscheidungsvariable, die angibt, ob das neu hinzuzufügende Monom n^M+1 den positiven Datenpunkt i akzeptiert (1) oder nicht (0)
$a_{m,i}$	$\{0,1\}$	Parameter, der angibt, ob Monom m den positiven Datenpunkt i im Trainingsdatensatz akzeptiert (1) oder nicht (0)
$\hat{b}^o$	$\mathbb{N}$	Obere Schranke im Algorithmus Alg$^{B\&P}$ für die minimale Komplexität des zu ermittelnden booleschen Ausdrucks
c_m	$\mathbb{N}$	Anzahl der Literale in Monom m. Entspricht dem entsprechenden Zielfunktionskoeffizienten des MP
$L^{ExklMonome}$	Liste	Liste von Monomen, die einem RMP nicht hinzugefügt werden dürfen
L^{Monome}	Liste	Liste der Monome in einem RMP
n^{Vz}	$\mathbb{N}$	Index der Entscheidungsvariable eines RMP in der verzweigt wird
n^{Dn}	$\mathbb{N}$	Anzahl der negativen Datenpunkte im Trainingsdatensatz
n^{Dp}	$\mathbb{N}$	Anzahl der positiven Datenpunkte im Trainingsdatensatz

n^F	$\mathbb{N}$	Anzahl der Features in einem Trainingsdatensatz
n^M	$\mathbb{N}$	Anzahl der berücksichtigten Monome in einem RMP
S^{NGanzZ}	Menge	Menge aller nichtganzzahligen Entscheidungsvariablen in der optimalen Lösung eines XRMP
$T^{Training}$	Tabelle	Ein Trainingsdatensatz
u_m	$\{0,1\}$	Entscheidungsvariable des RMP, die festlegt, ob Monom m Teil des zu lernenden booleschen Ausdrucks ist (1) oder nicht (0)
u_m^{XMP*}	$\{0,1\}$	Wert der Entscheidungsvariable m in der optimalen Lösung des XMP
v_i	$\mathbb{R}$	Entscheidungsvariable des DP, welche der Nebenbedingung i des XRMP zugeordnet ist
v_i^*	$\mathbb{R}$	Wert einer Entscheidungsvariablen v_i in der optimalen Lösung eines DP
$w_f^{n\mu}$	$\{0,1\}$	Stelle f des negativen Anteils eines Monoms μ in Dual-Rail-Darstellung
w_f^n	$\{0,1\}$	Entscheidungsvariable, die angibt, ob das neu hinzuzufügende Monom die dem Feature f entsprechende Variable als negatives Literal enthält (1) oder nicht (0)
$w_f^{p\mu}$	$\{0,1\}$	Stelle f des positiven Anteils eines Monoms μ in Dual-Rail-Darstellung
w_f^p	$\{0,1\}$	Entscheidungsvariable, die angibt, ob das neu hinzuzufügende Monom die dem Feature f entsprechende Variable als positives Literal enthält (1) oder nicht (0)
x_{if}^p	$\{0,1\}$	Wahrheitswert des i-ten positiven Datenpunkts hinsichtlich Feature f
$\overline{x_{if}^p}$	$\{0,1\}$	Negierter Wahrheitswert des i-ten positiven Datenpunkts hinsichtlich Feature f
x_{if}^n	$\{0,1\}$	Wahrheitswert des i-ten negativen Datenpunkts hinsichtlich Feature f
$\overline{x_{if}^n}$	$\{0,1\}$	Negierter Wahrheitswert des i-ten negativen Datenpunkts hinsichtlich Feature f
z^{MP*}	$\mathbb{N}$	Optimaler Zielfunktionswert eines MP
z^{RMP*}	$\mathbb{N}$	Optimaler Zielfunktionswert eines RMP
z^{SP*}	$\mathbb{R}$	Optimaler Zielfunktionswert eines SP
z^{XMP*}	$\mathbb{R}$	Optimaler Zielfunktionswert eines XMP
z^{XRMP*}	$\mathbb{R}$	Optimaler Zielfunktionswert eines XRMP

Kapitel 4.5

Formelzeichen	Typ	Bedeutung
$B_{l,j}(\boldsymbol{x})$	Bool. Ausdr.	Boolescher Ausdruck, der dem j-ten Modell in VR l entspricht
$B^{HL}(\boldsymbol{x})$	Bool. Ausdr.	Boolescher Ausdruck, der der HLF entspricht
$B^{MS}_{l,j,k}(\boldsymbol{x})$	Bool. Ausdr.	Boolescher Ausdruck, der der MSF jk des VR l entspricht
d^{Dist}	$\mathbb{N}$	Kleinster Abstand einer Lösung, d. h. Variante, zu allen bereits im Trainingsdatensatz befindlichen Varianten
d_i^{Dist}	$\mathbb{N}$	Hamming-Abstand einer Lösung von einem bereits im Datensatz vorhandenen Datenpunkt i
$\hat{d}^{Komp}$	$\mathbb{N}$	Maximale Differenz der Komplexität eines Modells zur Komplexität des komplexitätsminimalen Modells in einem VR, bei der es noch berücksichtigt wird
$d^{Komp}_{l,j}$		Differenz der Komplexität eines Modells j zur Komplexität des komplexitätsminimalen Modells im VR l
$d^{MS}_{l,j,k}$	$\{0,1\}$	Entscheidungsvariable, die angibt, ob sich die Vorhersagen der Modelle j und k des VR des Labels l für eine Lösung, d. h. eine Variante, unterscheiden (1) oder nicht (0)
$L^{ExklMonome}$	Liste	Liste von Monomen, die einem RMP nicht hinzugefügt werden dürfen
n^D	$\mathbb{N}$	Anzahl der Datenpunkte in einem Trainingsdatensatz
n^F	$\mathbb{N}$	Anzahl der Features eines Trainingsdatensatzes
$n^{Kl,HL}$	$\mathbb{N}$	Anzahl der Klauseln in einer HLF
$n^{Kl,MS}_{l,j,k}$	$\mathbb{N}$	Anzahl der Klauseln in der MSF jk eines VR l
n^L	$\mathbb{N}$	Anzahl der Labels in einem Trainingsdatensatz
n^{MW}	$\mathbb{N}$	Anzahl der Ausprägungen eines Produktmerkmals
n^{VR}	$\mathbb{N}$	Definierte Größe der VRs für Methode 5
$r^{MS}_{l,j,k}$	$[0,1]$	Gewichtung der MSF, die den Modellen j und k im VR l zugeordnet ist
$u^{Vorh}_{k,m}$	$\{0,1\}$	Parameter, der festlegt, ob Monom m Teil des k-ten booleschen Ausdrucks in einem Versionenraum ist (1) oder nicht (0)
u_m	$\{0,1\}$	Entscheidungsvariable des RMP, die festlegt, ob Monom m Teil des zu lernenden booleschen Ausdrucks ist (1) oder nicht (0)
w^{MS}	$[0,1]$	Gewichtung des Kriteriums Modellseparation

$w_{f,m}^{p,HL}$	$\{0,1\}$	Parameter, der angibt, ob Klausel m der HLF das Feature f als positives Literal enthält (1) oder nicht (0)
$w_{f,l,j,k,m}^{p,MS}$	$\{0,1\}$	Parameter, der angibt, ob Klausel m der MSF jk des VR des Labels l das Feature f als positives Literal enthält (1) oder nicht (0)
$w_{f,m}^{n,HL}$	$\{0,1\}$	Parameter, der angibt, ob Klausel m der HLF das Feature f als negatives Literal enthält (1) oder nicht (0)
$w_{f,l,j,k,m}^{n,MS}$	$\{0,1\}$	Parameter, der angibt, ob Klausel m der MSF jk des VR l das Feature f als positives Literal enthält (1) oder nicht (0)
x_f	$\{0,1\}$	Ausprägung von Feature f in einer zu wählenden Variante
$x_{i,f}^{Vorh}$	$\{0,1\}$	Ausprägung von Feature f in der bereits im Trainingsdatensatz vorhandenen Variante i

Kapitel 4.6

Formelzeichen	Typ	Bedeutung
S^{Reg}	Menge	Menge von Regeln eines LLKM
$T^{Ausreißer}$	Tabelle	Tabelle, die für jeden Eintrag einer Literal- oder Monomtabelle den Ausreißerwert angibt
$T^{Alternativ}$	Tabelle	Tabelle, die für jeden Eintrag einer Literal- oder Monomtabelle den alternativen Eintrag mit der höchsten Vorhersagewahrscheinlichkeit enthält

Kapitel 5.1 und Anhang A7

Formelzeichen	Typ	Bedeutung
δ	$[0,1]$	Parameter des Approximate Model Countings, der das Signifikanzniveau des resultierenden Intervalls festlegt
ε	$[0,1]$	Parameter des Approximate Model Countings, der die Größe des resultierenden Intervalls festlegt

Kapitel 5.2 und Anhang A8

Formelzeichen	Typ	Bedeutung
b^{min}	$\{0,1\}$	Variable, die angibt ob für ein Experiment eine minimale MSTL erstellt wurde (1) oder nicht (0)
b^{ZB1}	$\{0,1\}$	Variable, die angibt ob für ein Experiment die Zeitbeschränkung für Schritt 1 der Methode 2 erreicht wurde (1) oder nicht (0)

b^{ZB2}	$\{0,1\}$	Variable, die angibt ob für ein Experiment die Zeitbeschränkung für Schritt 2 der Methode 2 erreicht wurde (1) oder nicht (0)
d^{MSTL}	$[0,1]$	Normierte Distanz der Ergebnis- und der Referenz-MSTL
$\hat{d}^{\mathrm{MSTL}}$	$\mathbb{N}$	Maximal mögliche Distanz der Ergebnis- und der Referenz-MSTL
$d^{MSTL,abs}$	$\mathbb{N}$	Absolute Distanz der Ergebnis- und der Referenz-MSTL
d^{STO}	$\mathbb{N}$	Differenz der Anzahlen von STOs in einer Ergebnis- und einer Referenz-MSTL
$d_{k,l}^{STO}$	$\mathbb{N}$	Distanz zweier STOs k und l einer MSTL
n^{Abh}	$\mathbb{N}$	Anzahl der ZKKs in einer Referenz-MSTL, die von der gültigen STO abhängen
$n_{i,j}^{AnzPart}$	$\mathbb{N}$	Anzahl möglicher Partitionen einer Zahl i deren kleinster Summand größer gleich j ist
$n_{i,j}^{AnzPart,gen}$	$\mathbb{N}$	Anzahl möglicher Partitionen einer Zahl i deren kleinster Summand j ist
n^{BGK}	$\mathbb{N}$	Anzahl der BGKs in einer Referenz-MSTL
n^{ClMin}	$\mathbb{N}$	Anzahl der Cluster derjenigen STO mit der geringsten Clusteranzahl in der Ergebnis-MSTL, falls diese weniger Cluster aufweist als die Referenz-MSTL, ansonsten in der Referenz-MSTL
n^{Mult}	$\mathbb{N}$	Anzahl der Multipositionen je STO einer Referenz-MSTL
n^{Sing}	$\mathbb{N}$	Anzahl der Positionen je STO einer Referenz-MSTL mit singulären Bezeichnungen
n^{STO}	$\mathbb{N}$	Anzahl der STOs in einer MSTL
n^{Sum}	$\mathbb{N}$	Ein Summand einer Ganzzahlpartition
n^{SumMax}	$\mathbb{N}$	Größter Summand einer Ganzzahlpartition
n^{VSTL}	$\mathbb{N}$	Anzahl der VSTLs in der Menge S^{VSTL}
n^{ZKK}	$\mathbb{N}$	Anzahl der ZKKs, die einer STO einer Referenz-MSTL zugeordnet sind
$n^{ZKK,MSTL}$	$\mathbb{N}$	Anzahl von ZKKs in einer Referenz-MSTL
$n^{ZuTeilen}$	$\mathbb{N}$	Natürliche Zahl für die eine Ganzzahlpartition generiert werden soll
r^{Abh}	$[0,1]$	Anteil von ZKKs einer MSTL, die von der gültigen STO abhängen
r^{BGK}	$[0,1]$	Verhältnis der Anzahl von BGKs zu ZKKs in einer Referenz-MSTL

r^{Mult}	[0,1]	Anteil mehrfach auftretender Bezeichnungen von ZKKs, die einer STO einer Referenz-MSTL zugeordnet sind
r^{RadSTO}	[0; 0,5)	Radius eines Intervalls aus dem die Auftretenswahrscheinlichkeiten für STOs gleichmäßig zufällig gewählt werden
r^{RadZKK}	[0; 0,5)	Radius eines Intervalls aus dem die Auftretenswahrscheinlichkeiten für ZKKs gleichmäßig zufällig gewählt werden
S_k^{Cl}	Menge	Menge an Clustern zu BGKs in einer MSTL k
S^{STO}	Menge	Menge aller STOs einer Ergebnis- und einer Referenz-MSTL
S^{VSTL}	Menge	Menge von VSTLs, die für die datenbasierte Erstellung einer MSTL verwendet werden
δ	[0,1]	Parameter des Uniform Model Samplings, der das Signifikanzniveau des Intervalls festlegt, in dem eine Gleichverteilung angenähert wird
ε	[0,1]	Parameter des Uniform Model Samplings, der das Intervall festlegt, in dem eine Gleichverteilung angenähert wird

Kapitel 5.3 und Anhang A9

Formelzeichen	Typ	Bedeutung
b^{min}	{0,1}	Variable, die angibt ob für ein Experiment ein minimaler MAPL erstellt wurde (1) oder nicht (0)
b^{ZB1}	{0,1}	Variable, die angibt ob für ein Experiment die Zeitbeschränkung für Schritt 1 der Methode 3 erreicht wurde (1) oder nicht (0)
b^{ZB2}	{0,1}	Variable, die angibt ob für ein Experiment die Zeitbeschränkung für Schritt 2 der Methode 3 erreicht wurde (1) oder nicht (0)
d^{MAPL}	$\mathbb{N}$	Normierte Distanz des Ergebnis- und des Referenz-MAPL
$\hat{d}^{MAPL}$	$\mathbb{N}$	Maximal mögliche Distanz des Ergebnis- und des Referenz-MAPL
$d^{MAPL,abs}$	$\mathbb{N}$	Absolute Distanz des Ergebnis- und des Referenz-MAPL
d^{STO}	$\mathbb{N}$	Differenz der Anzahlen der STOs in einem Ergebnis- und einem Referenz-MAPL
$d_{k,l}^{STO}$	$\mathbb{N}$	Distanz zweier STOs k und l eines MAPL
n^{Abh}	$\mathbb{N}$	Anzahl der AVKs eines Referenz-MAPL, die von der gültigen STO abhängen

n^{AK}	$\mathbb{N}$	Anzahl der AVKs, die einer STO eines Referenz-MAPL zugeordnet sind
n^{AKMin}	$\mathbb{N}$	Anzahl der AVKs je STO eines Referenz-MAPL sofern dieser weniger STOs aufweist als ein zugehöriger Ergebnis-MAPL, ansonsten Anzahl der AVKs je STO des Ergebnis-MAPL
$n^{AK,MAPL}$	$\mathbb{N}$	Anzahl der AVKs in einem Referenz-MAPL
n^{Mult}	$\mathbb{N}$	Anzahl der Multipositionen je STO eines Referenz-MAPL
n^{Sing}	$\mathbb{N}$	Anzahl der Positionen je STO eines Referenz-MAPL mit singulären Bezeichnungen
n^{STO}	$\mathbb{N}$	Anzahl der STOs in einem Referenz-MAPL
n^{VAPL}	$\mathbb{N}$	Anzahl der VAPLs in der Menge S^{VAPL}
n^{VBZ}	$\mathbb{N}$	Anzahl der Vorrangbeziehungen in einem MAPL
r^{Abh}	$[0,1]$	Anteil von AVKs eines Referenz-MAPL, die von der gültigen STO abhängen
r^{Mult}	$[0,1]$	Anteil mehrfach auftretender Bezeichnungen von AVKs, die einer STO eines Referenz-MAPL zugeordnet sind
r^{VBZ}	$[0,1]$	Anteil von gültigen Vorrangbeziehungen in einem MAPL an allen möglichen Vorrangbeziehungen in diesem Referenz-MAPL
r^{RadAK}	$[0;\ 0{,}5)$	Radius eines Intervalls aus dem die Auftretenswahrscheinlichkeiten für AVKs gleichmäßig zufällig gewählt werden
r^{RadSTO}	$[0;\ 0{,}5)$	Radius eines Intervalls aus dem die Auftretenswahrscheinlichkeiten für STOs gleichmäßig zufällig gewählt werden
$S_k^{E1,Bez}$	Menge	Menge der Vorrangbeziehungen in einer STO k eines Referenz-MAPL bezogen auf die Bezeichnungen der AVOs
S^{STO}	Menge	Menge aller STOs eines Ergebnis- und eines Referenz-MAPL
S^{VAPL}	Menge	Menge von VAPLs, die für die datenbasierte Erstellung eines MAPL verwendet werden
$t^{M3,1}$	$\mathbb{N}$	Zeitbeschränkung in Sekunden für Schritt 1 der Methode 3 für ein Experiment
$t^{M3,2}$	$\mathbb{N}$	Zeitbeschränkung in Sekunden für Schritt 2 der Methode 3 für ein Experiment
δ	$[0,1]$	Parameter des Uniform Model Samplings, der das Signifikanzniveau des Intervalls festlegt, in dem eine Gleichverteilung angenähert wird

ε	$[0,1]$	Parameter des Uniform Model Samplings, der das Intervall festlegt, in dem eine Gleichverteilung angenähert wird

Kapitel 5.4 und Anhang A10

Formelzeichen	Typ	Bedeutung
$b_{i,j}^{korrL}$	$\{0,1\}$	Variable die angibt, ob das i-te Label des j-ten Datenpunkts eines Testdatensatzes durch ein trainiertes Modell korrekt wiedergegeben wird (1) oder nicht (0).
b_i^{korrM}	$\{0,1\}$	Variable die angibt, ob ein auf dem i-ten Label eines Datensatzes trainiertes Modell dem tatsächlich gültigen Modell für dieses Label entspricht (1) oder nicht (0).
n^L	$\mathbb{N}$	Anzahl der Labels in einem Datensatz
$n^{MinFeatures}$	$\mathbb{N}$	Einstellparameter des Algorithmus DK-XTSD, der angibt, ab wie vielen Features in einem Teiltrainingsdatensatz der Algorithmus von Chatterjee (2018) eingesetzt wird
n^{Test}	$\mathbb{N}$	Anzahl der Datenpunkte in einem Testdatensatz
$n^{Training}$	$\mathbb{N}$	Anzahl der Datenpunkte in einem Trainingsdatensatz
$\hat{n}^{Training}$	$\mathbb{N}$	Größte Anzahl von Datenpunkten in einem Trainingsdatensatz über alle Trainingsdatensätze einer Experimentreihe
r^{GenEx}	$[0,1]$	Mittlere Genauigkeit von datenbasiert erstellten Regeln auf einem Testdatensatz ohne Berücksichtigung von Standardpositionen
r^{GenIn}	$[0,1]$	Mittlere Genauigkeit von datenbasiert erstellten Regeln auf einem Testdatensatz unter Berücksichtigung von Standardpositionen
r^{ModEx}	$[0,1]$	Anteil der logisch übereinstimmenden Regeln zwischen einem datenbasiert erstellten und einem tatsächlichen LLKM ohne Berücksichtigung von Standardpositionen
r^{ModIn}	$[0,1]$	Anteil der logisch übereinstimmenden Regeln zwischen einem datenbasiert erstellten und einem tatsächlichen LLKM unter Berücksichtigung von Standardpositionen
t^{Reg}	$\mathbb{N}$	Zeitbeschränkung in Sekunden für die datenbasierte Erstellung einer Regel im Rahmen eines Experiments

Kapitel 5.5 und Anhang A11

Formelzeichen	Typ	Bedeutung
$n^{Training}$	$\mathbb{N}$	Anzahl der Datenpunkte in einem Trainingsdatensatz
$\hat{n}^{Training}$	$\mathbb{N}$	Größte Anzahl von Datenpunkten in einem Trainingsdatensatz über alle Trainingsdatensätze einer Experimentreihe
n^{VR}	$\mathbb{N}$	Größe der VRs für Methode 5
r^{GenEx}	$[0,1]$	Mittlere Genauigkeit von datenbasiert erstellten Regeln auf einem Testdatensatz ohne Berücksichtigung von Standardpositionen
w^{MS}	$[0,1]$	Gewichtung des Kriteriums Modellseparation angibt

Kapitel 5.6 und Anhang A12

Formelzeichen	Typ	Bedeutung
$k^{Datenpunkte}$	$\{0{,}5;\ 1\}$	Verwendetes Verfahren zur Auswahl von Datenpunkten für den Random-Forest-Algorithmus
$k^{Features}$	$\{log2, sqrt\}$	Verwendetes Verfahren zur Auswahl von Features für den Random-Forest-Algorithmus
n^{Fehler}	$\mathbb{N}$	Anzahl der eingebrachten Fehler in einem LLKM
$k^{Fehlerart}$	$\{Gleichv., \ldots\}$	Art der eingebrachten Fehler in einem LLKM
$k^{Gewichtung}$	$\{N.gesetzt, Ausgegl.\}$	Verwendetes Verfahren zur Gewichtung von Datenpunkten für den Random-Forest-Algorithmus
k^{KM}	$\{A, B, C\}$	Betrachtetes Konfigurationsmodell
n^{DT}	$\mathbb{N}$	Anzahl der verwendeten Entscheidungsbäume für den Random-Forest-Algorithmus
$v_{i,j}$	fallabhängig	j-te mögliche Ausprägung eines Produktmerkmals i in einem HLKM des Industriepartners
x_i	$\{0,1\}$	Variable zur Darstellung des i-ten Produktmerkmals in One-Hot-Codierung für boolesche Merkmale
$x_{i,j}$	$\{0,1\}$	j-te Variable zur Darstellung des i-ten Produktmerkmals in One-Hot-Codierung für kategorische und mehrwertige Merkmale
x_i^{init}	fallabhängig	Produktmerkmal in einem Konfigurationsmodell
y_j	$\{0,1\}$	Ausprägung eines abhängigen Parameters i in einem Konfigurationsmodell

Anhang A1

Formelzeichen	Typ	Bedeutung
a_{ij}	$\mathbb{R}$	Koeffizienten der Nebenbedingungen des MP oder RMP
b_i	$\mathbb{R}$	Parameter der rechten Seite des MP oder RMP
c_i	$\mathbb{R}$	Zielfunktionskoeffizient des MP oder RMP
M	$\mathbb{N}$	Anzahl der Zeilen des RMP
N	$\mathbb{N}$	Anzahl der Spalten des RMP
u_i	$\mathbb{R}$	Entscheidungsvariable des MP oder RMP
u_i^b	$\mathbb{R}$	Entscheidungsvariable des RMP
v_j	$\mathbb{R}$	Entscheidungsvariable des DP
v_j^*	$\mathbb{R}$	Wert der zugehörigen Entscheidungsvariable für eine optimale Lösung des DP

1 Einleitung

Im Folgenden wird zunächst der Forschungsgegenstand der vorliegenden Arbeit motiviert und dessen Kontext dargestellt (Kapitel 1.1). Anschließend werden hieraus Probleme und zugehörige Forschungsfragen abgeleitet (Kapitel 1.2). Abschließend wird die Methodik zur Beantwortung der Forschungsfragen vorgestellt (Kapitel 1.3).

1.1 Motivation

Industrieunternehmen, die auftragsbezogen entwickelte Produkte (engl. Engineer-to-Order, ETO) anbieten, stehen unter einem hohen Wettbewerbsdruck und sehen sich zunehmend mit steigenden Anforderungen hinsichtlich Produktqualität und Lieferzeiten konfrontiert (Cannas et al. 2022, S. 974). Die **auftragsbezogene Konfiguration** (engl. Configure-to-Order, CTO) unter Einsatz von Konfigurationssystemen bietet Unternehmen die Möglichkeit, diese Anforderungen zu erfüllen. Einerseits können Produkte in hoher Variantenanzahl angeboten und damit individuelle Kundenwünsche adressiert werden (ElMaraghy et al. 2013, S. 632). Andererseits bestehen gegenüber der auftragsbezogenen Entwicklung deutliche Vorteile hinsichtlich Produktqualität, Lieferzeit und Kosten (Haug et al. 2019b, S. 134). Damit gewinnt der Wandel hin zu CTO und damit die Einführung von Konfigurationssystemen für ETO-Unternehmen an Bedeutung (Cannas et al. 2022, S. 980).

Konfigurationssysteme unterstützen oder automatisieren dispositive Aufgaben im Auftragsabwicklungsprozess, wie insbesondere das Verfassen der Anforderungsspezifikation[1] sowie die Erstellung von Stücklisten[2] und Arbeitsplänen (Zhang et al. 2015, S. 58). Die Erstellung von Stücklisten und Arbeitsplänen ist in Industrieunternehmen eine zentrale Aufgabe der Arbeitsplanung (Wiendahl 2019, S. 190), welche im Fokus der vorliegenden Arbeit steht. Eine Studie von Myrodia et al. (2018) zeigt, dass von 59 betrachteten Unternehmen, die Konfigurationssysteme einsetzen, 56 % diese ausschließlich für den Vertrieb und nicht für technische Aufgaben nutzen. Damit bleibt Potenzial für den Einsatz von Konfigurationssystemen für technische Aufgaben im Rahmen der Arbeitsplanung wie insbesondere die Erstellung von **Stücklisten** mittels **Produktkonfiguration** und **Arbeitsplänen** mittels **Prozesskonfiguration** teilweise

[1] Siehe hierzu z.B. Bender & Gericke (2021, S. 198–199).
[2] Wie in Kapitel 2.1 ausgeführt wird, existieren in Unternehmen typischerweise verschiedene Arten von Stücklisten. Die vorliegende Arbeit betrachtet jedoch ausschließlich die Konfiguration von Fertigungsstücklisten, weshalb im Folgenden sofern nicht anders angegeben mit Stücklisten Fertigungsstücklisten gemeint sind.

ungenutzt. Dies ist insbesondere vor dem Hintergrund relevant, dass in einer Studie von Prote et al. (2017) 52 % von 91 befragten Unternehmen gerade den Aufwand für die Erstellung von Arbeitsplänen als hoch oder zu hoch einschätzten. Somit besteht auch für Unternehmen, die bereits konfigurierbare Produkte anbieten, Potenzial für die Einführung von Konfigurationssystemen zur Produkt- und Prozesskonfiguration.

Den Vorteilen der Einführung eines Konfigurationssystems stehen **Herausforderungen** gegenüber. Typischerweise handelt es sich bei der Einführung eines Konfigurationssystems um ein Projekt, an dem Experten verschiedener Disziplinen beteiligt sind (Haug et al. 2012, S. 476), das mehrere Monate dauert (Haug et al. 2019b, S. 139) und mehrere tausend Mitarbeiterstunden in Anspruch nimmt (Kristjansdottir et al. 2018a, S. 66; Shafiee et al. 2020, S. 14). Die Nutzung eines Konfigurationssystems setzt ein konfigurierbares Produkt voraus. Dessen Varianten bilden eine Produktfamilie (Tiihonen et al. 1998, S. 30–35). Varianten eines Produkts werden im Konfigurationsprozess durch Entscheidungen im Rahmen zuvor festgelegter Möglichkeiten spezifiziert (Abbasi et al. 2013, S. 162). Dadurch ergibt sich im Gegensatz zu einem auftragsbezogen entwickelten Produkt ein definierter und beschränkter Produktraum. Dieser kann jedoch mit typischen Variantenanzahlen von z. B. 10^{24} im Anlagenbau (Blumöhr et al. 2019, S. 36) eine unüberschaubare Größe aufweisen. Diese **Komplexität** setzt sich in den Konfigurationsmodellen, die die Wissensbasis für eine Konfiguration darstellen, fort: In der Literatur sind Konfigurationsmodelle mit 18.000 hinterlegten Regeln alleine für die Erstellung von Stücklisten beschrieben (Sinz 2004, S. 3). Vor diesem Hintergrund kann erklärt werden, warum Wissensbereitstellung und Wissensmodellierung zu den größten Herausforderungen bei der Einführung und Nutzung von Konfigurationssystemen zählen (Haug et al. 2019a, S. 121; Kristjansdottir et al. 2018b, S. 203).

Werden Konfigurationssysteme für eine **bestehende Produktlinie** eingeführt, sind im Unternehmen technische Produktdokumentationen wie insbesondere Stücklisten und Arbeitspläne für Produktvarianten aus zurückliegenden Aufträgen vorhanden[3]. Diese sind insofern korrekt, als sie bereits für eine erfolgreiche Auftragsabwicklung verwendet wurden. Da diese Dokumente i. d. R. von Domänenexperten erstellt wurden, sind sie das Produkt desselben Domänenwissens, das auch der Erstellung von

[3] Aufgrund des Betrachtungsrahmens der vorliegenden Arbeit sind im Folgenden mit technischer Produktdokumentation (kurz: Dokumentation, auch: Dokumente) sofern nicht anders angegeben Fertigungsstücklisten und Arbeitspläne gemeint. Eine technische Produktdokumentation kann in der industriellen Praxis jedoch weitere Dokumente umfassen, wie z. B. von Bender & Gericke (2021, S. 905–906) beschrieben.

Konfigurationsmodellen zugrunde liegt. Es ist deshalb davon auszugehen, dass hieraus mittels **datenbasierter Methoden** Muster extrahiert werden können, aus denen sich ein **Konfigurationsmodell erstellen** lässt, das das zugrundeliegende Domänenwissen repräsentiert. In Fällen in denen eine ausreichende Anzahl von Daten in Form von technisch dokumentierten Produktvarianten zur Verfügung steht, kann die Erstellung von Konfigurationsmodellen potenziell vollständig automatisiert werden.

Es ist jedoch davon auszugehen, dass nicht in allen Fällen initial genügend Daten zur Verfügung stehen, um für eine bestehende Produktlinie ein Konfigurationsmodell mit ausreichender Genauigkeit automatisch zu erstellen. Ggf. ist zum einen eine sukzessive datenbasierte Verfeinerung im laufenden Betrieb möglich: Die mittels Konfigurationssystemen erstellten Stücklisten und Arbeitspläne werden manuell überprüft und das Konfigurationsmodell wird ggf. angepasst, bis eine ausreichende Genauigkeit vorliegt. Eine manuelle Überwachung und Anpassung im Betrieb ist in der Industrie auch für manuell erstellte Konfigurationsmodelle üblich. Dadurch kann jedoch bei der Auftragsabwicklung zusätzlicher Aufwand anfallen, der zu einer Erhöhung der Lieferzeit führen kann. Zum anderen kann die **Erweiterung der Datenbasis** unabhängig vom Auftragsabwicklungsprozess erfolgen. Dafür werden aus dem Raum aller zulässig spezifizierbaren Produktvarianten diejenigen ausgewählt, die einen hohen Informationsgewinn für eine datenbasierte Erstellung des Konfigurationsmodells erwarten lassen. Für diese werden Dokumente, d. h. Stücklisten und Arbeitspläne, erstellt, so dass sich zusätzliche Daten für eine datenbasierte Erstellung von Konfigurationsmodellen ergeben. Prinzipiell ermöglicht ein solches Vorgehen auch Unternehmen mit geringer Expertise in der Wissensmodellierung eine Erstellung von Konfigurationsmodellen durch die Vorgabe von Beispielen. Die **Auswahl geeigneter Varianten**[4] stellt hierbei jedoch eine Herausforderung dar.

Aufgrund ihrer Komplexität sind Konfigurationsmodelle nicht nur aufwändig zu erstellen, sondern auch fehleranfällig (Voronov 2013, S. 4). **Fehler** können bei der Pflege der Modelle oder bereits bei deren Erstellung auftreten (Shafiee et al. 2020, S. 14). Die Überführung von Domänenwissen in ein Modell erfolgt i. d. R. nicht durch Domänenexperten selbst, sondern durch Experten für Wissensrepräsentation (engl. Knowledge Representation Experts) in Abstimmung mit den Domänenexperten (Haug et al. 2012, S. 475–478). Diese Arbeitsteilung geht mit typischen Schnittstellenproblemen, wie

[4] Hier und im Folgenden wird der Begriff Variante als Kurzform für Produktvariante verwendet.

insbesondere Missverständnissen oder einer unzureichenden Wissensweitergabe, einher, aus denen Fehler resultieren können (Haug et al. 2012, S. 478; Shafiee et al. 2017, S. 987–988). Darüber hinaus zeigen Felfernig et al. (2015) in einer Studie mit zehn erfahrenen Experten für Wissensrepräsentation, dass auch diesen in nicht unerheblichem Maße Fehler bei der Formalisierung von Wissen unterlaufen. Fehler in Konfigurationsmodellen können jedoch auch bei einer datenbasierten Erstellung und darüber hinaus durch manuelle Änderungen des Konfigurationsmodells im Betrieb entstehen. Werden mit Hilfe von Konfigurationssystemen im Rahmen der Arbeitsplanung Stücklisten und Arbeitspläne erstellt, haben diese Fehler einen unmittelbaren Einfluss auf die Produktionsdurchführung. In Folge kann es zu schwerwiegenden Unterbrechungen des Materialflusses, zu Mängeln in der Produktqualität oder sogar zu Mängeln in der Produktsicherheit kommen (Küchlin 2020, S. 9). Der Überprüfung von Konfigurationsmodellen kommt damit eine herausragende praktische Bedeutung zu. In der Literatur zu maschinellem Lernen werden Methoden der **Anomalieerkennung** erfolgreich eingesetzt, um Hinweise auf Fehler in Datensätzen zu erhalten (Aggarwal 2017, S. 399–422). Konfigurationsmodelle können zu einem gewissen Teil in Daten umgewandelt werden, die sich für eine solche Anomalieerkennung eignen[5]. Auch für die **Überprüfung von Konfigurationsmodellen** erscheint daher der Einsatz von datenbasierten Methoden vielversprechend[6].

Es lässt sich festhalten, dass in der Industrie **Potenzial** für die Einführung und weitergehende Verwendung von Konfigurationssystemen, insbesondere für die Produkt- und Prozesskonfiguration, besteht. Diesem Potenzial stehen jedoch Herausforderungen gegenüber, wie insbesondere ein hoher Erstellungsaufwand und eine hohe Fehleranfälligkeit von Konfigurationsmodellen. Datenbasierte Methoden bieten Möglichkeiten, diesen Herausforderungen zu begegnen, sind aber – wie in Kapitel 3 dargelegt wird – noch nicht ausreichend erforscht, um in der industriellen Praxis eingesetzt werden zu können. Deshalb befasst sich die vorliegende Arbeit mit der datenbasierten Erstellung und

[5] Diese Daten sind formalisierte Repräsentationen eines Konfigurationsmodells und damit insbesondere nicht mit Daten in Form von spezifizierten Produktvarianten zu verwechseln, die für die Erstellung von Konfigurationsmodellen verwendet werden können.
[6] Auch für die Überprüfung von Stücklisten und Arbeitsplänen, die als Eingangsdaten für die Erstellung von Konfigurationsmodellen dienen kann prinzipiell Anomalieerkennung eingesetzt werden. Dies liegt jedoch nicht im Betrachtungsrahmen der vorliegenden Arbeit, auch wenn sich die in Kapitel 4.6 dargestellte Methode prinzipiell auf diesen Anwendungsfall übertragen lässt.

Überprüfung von Modellen zur Produkt- und Prozesskonfiguration. Abbildung 1.1 veranschaulicht das entsprechende Konzept schematisch.

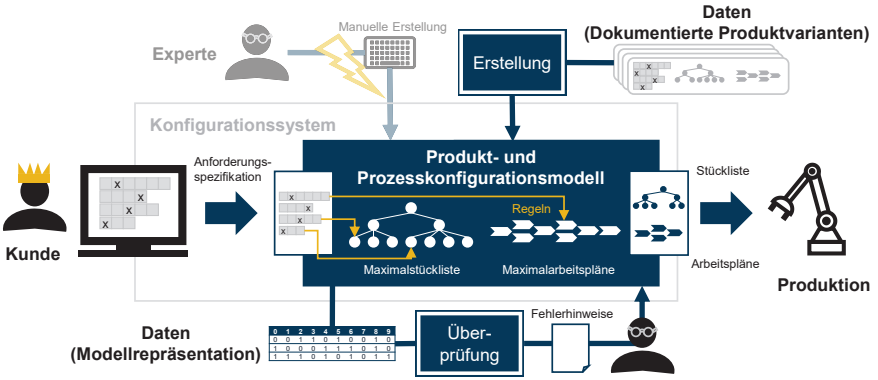


Abbildung 1.1: Konzept der datenbasierten Erstellung und Überprüfung von Modellen zur Produkt- und Prozesskonfiguration

1.2 Probleme und Forschungsfragen

Um die industrielle Relevanz der vorliegenden Arbeit zu gewährleisten, wird von Konfigurationssystemen und zugehörigen Konfigurationsmodellen ausgegangen, wie sie gegenwärtig in der Industrie eingesetzt werden. Diese bestehen aus

- **Maximalstücklisten**, die die potenziellen Komponenten des konfigurierbaren Produkts enthalten,
- **Maximalarbeitsplänen**, die die potenziellen Arbeitsvorgänge seines Herstellprozesses enthalten sowie
- **Regeln**, die die Elemente der Maximalstückliste bzw. des Maximalarbeitsplans und deren Ausprägungen auf Basis der Anforderungsspezifikation festlegen,

wie in Kapitel 2.2.2.4 ausgeführt wird. Um Maximalstücklisten und Maximalarbeitspläne von den individuell für jede Variante erstellten Stücklisten und Arbeitsplänen abzugrenzen, werden letztere im Folgenden als variantenbezogene Stücklisten bzw. Arbeitspläne bezeichnet. Die datenbasierte Erstellung von Produkt- und Prozesskonfigurationsmodellen basiert auf der **Prämisse**, dass im Unternehmen bereits entwickelte und vertriebene Varianten eines Produkts vorliegen und die konfigurierbaren Merkmale des Produkts bereits definiert sind. Der Fokus der vorliegenden Arbeit liegt auf

Anwendungsfällen, in denen das Unternehmen bereits ein Konfigurationssystem für den Vertrieb einsetzt, aber nicht für die Arbeitsablaufplanung[7]. Die entwickelten Methoden können auf Anwendungsfälle übertragen werden, in denen das Unternehmen zwar kein Konfigurationssystem für den Vertrieb einsetzt, aber dennoch eine Anforderungsspezifikation auf Basis definierter Produktmerkmale erstellt. Da das Produkt bereits vertrieben wird, wird davon ausgegangen, dass im Unternehmen bereits technisch dokumentierte Varianten – d. h. durch Merkmale spezifizierte Varianten mit zugehörigen variantenbezogenen Stücklisten und variantenbezogenen Arbeitsplänen – vorliegen. Deren Anzahl kann ausreichend groß sein, um ein hinreichend genaues Konfigurationsmodell zu erstellen oder zu gering, so dass eine Erweiterung der Datenbasis notwendig ist.

Unter diesen Prämissen und vor dem Hintergrund des in Kapitel 1.1 gegebenen Kontextes lassen sich die folgenden **Probleme** als Gegenstand der vorliegenden Arbeit identifizieren.

- Die datenbasierte **Erstellung** von Modellen zur Produkt- und Prozesskonfiguration (Problem 1),
    - die datenbasierte Erstellung von Maximalstücklisten (Problem 2),
    - die datenbasierte Erstellung von Maximalarbeitsplänen (Problem 3),
    - die datenbasierte Erstellung von Regeln (Problem 4),
    - die Auswahl von repräsentativen Varianten zur Erweiterung der Datenbasis (Problem 5),
- und die datenbasierte **Überprüfung** von Regeln (Problem 6).

Die Probleme 1 und 6 ergeben sich unmittelbar aus den in Kapitel 1.1 beschriebenen Herausforderungen in der industriellen Praxis. Die Probleme 2, 3 und 4 ergeben sich für die in der industriellen Praxis überwiegend eingesetzten und zuvor beschriebenen Konfigurationsmodelle als Teilprobleme von Problem 1. Problem 5 ist ein Teilproblem von Problem 1 für Anwendungsfälle mit unzureichender Datenbasis. Es wird betrachtet, um die industrielle Relevanz der vorliegenden Arbeit für eine große Anzahl von Anwendungsfällen zu gewährleisten. Durch Problem 6 wird die Überprüfung von Konfigurationsmodellen im Rahmen der vorliegenden Arbeit auf die Überprüfung von Regeln

[7] Dieser Fall wird u. a. von Haug et al. (2019a, S. 126) und Bredahl Rasmussen et al. (2021, S. 8) für verschiedene Industrieunternehmen beschrieben. Seine Einordnung gegenüber den Idealtypen ETO und CTO wird in Kapitel 2.2.1 thematisiert.

beschränkt. Insbesondere diese werden in der Literatur als fehleranfällig angesehen (z. B. von Küchlin 2020, S. 1).

Die Probleme 1 bis 6 werden im Rahmen der vorliegenden Arbeit gelöst. Dadurch ist es abschließend möglich, die folgenden **Forschungsfragen** zu beantworten:

- Wie und wie zuverlässig können Produkt- und Prozesskonfigurationsmodelle datenbasiert **erstellt** werden? (Forschungsfrage 1)
    - Wie und wie zuverlässig können Maximalstücklisten datenbasiert erstellt werden? (Forschungsfrage 2)
    - Wie und wie zuverlässig können Maximalarbeitspläne datenbasiert erstellt werden? (Forschungsfrage 3)
    - Wie und wie zuverlässig können Regeln datenbasiert erstellt werden? (Forschungsfrage 4)
    - Wie und wie effektiv kann die Datenbasis für die datenbasierte Erstellung von Regeln erweitert werden? (Forschungsfrage 5)
- Wie und wie zuverlässig können Regeln in Produkt- und Prozesskonfigurationsmodellen datenbasiert **überprüft** werden? (Forschungsfrage 6)

1.3 Methodik und Aufbau der Arbeit

Die vorliegende Arbeit lässt sich den **Technikwissenschaft**en im Sinne der Deutschen Akademie der Technikwissenschaften (acatech) zuordnen. „Technikwissenschaften schaffen kognitive Voraussetzungen für Innovation in der Technik und Anwendung technischen Wissens und legen die Grundlagen für die Reflexion ihrer Implikationen und Folgen“ (acatech (Hrsg.) 2013, S. 18). Das übergeordnete Ziel der vorliegenden Arbeit entspricht dem übergeordneten Ziel der Technikwissenschaften „erweiterte[ ] Möglichkeitsräume[ ] für das technische Handeln“ zu schaffen (acatech (Hrsg.) 2013, S. 19). Dabei folgt diese Arbeit dem Paradigma des Design Science Research. „[Design Science Research] seeks to enhance technology and science knowledge bases via the creation of innovative artifacts that solve problems and improve the environment in which they are instantiated” (vom Brocke et al. 2020, S. 1). Konkret entspricht die vorliegende Arbeit dem **Design Science Research Process** nach Peffers et al. (2007), der in Abbildung 1.2 (1) dargestellt ist.

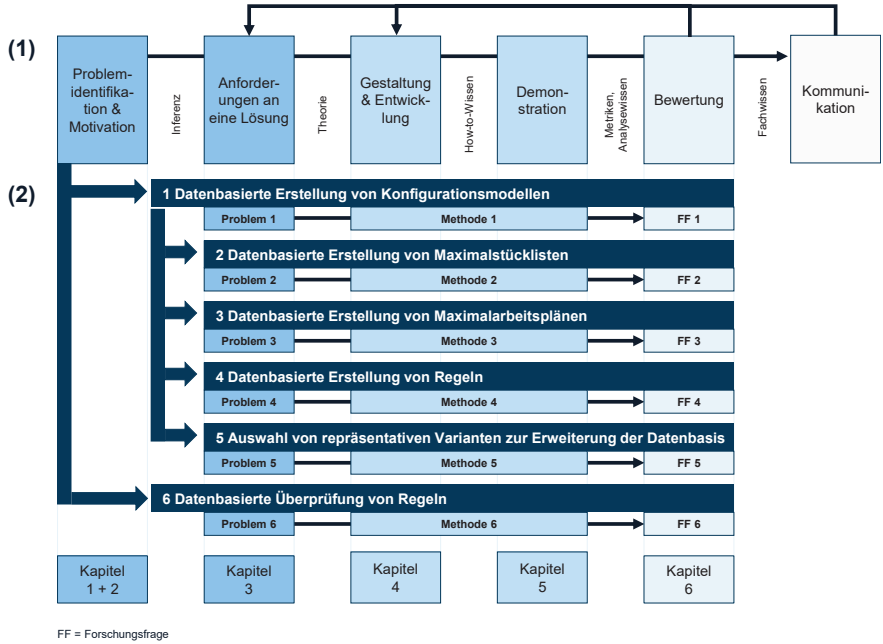


Abbildung 1.2: Methode und Aufbau der Arbeit (Methode nach Peffers et al. 2007, S. 93)

Auch der **Aufbau** der vorliegenden Arbeit entspricht dem Design Science Research Process, wie in Abbildung 1.2 (2) veranschaulicht. In Kapitel 1 wurden die in der vorliegenden Arbeit betrachteten Probleme 1 bis 6 und zugehörige Forschungsfragen aus der industriellen Praxis abgeleitet. In Kapitel 2 werden die für diese Arbeit relevanten Grundlagen vermittelt. Damit wird einerseits im Sinne des Design Science Research Process der Kontext der definierten Probleme beschrieben. Andererseits wird das notwendige Vorwissen zum Verständnis der im Rahmen der Arbeit entwickelten Artefakte vermittelt. In Kapitel 3 werden die Probleme 1 bis 6 konkretisiert indem Anforderungen an die zu deren Lösung erforderlichen Artefakte aus der übergeordneten Problemstellung abgeleitet werden. Außerdem wird untersucht, inwieweit bestehende Ansätze nach Stand der Forschung diese Anforderungen bereits erfüllen (entsprechend Peffers et al. 2007, S. 90). Aus der Differenz ergibt sich ein Lösungsdefizit, das durch die Entwicklung geeigneter Artefakte geschlossen wird. In Kapitel 4 werden die im Rahmen der vorliegenden Arbeit entwickelten Methoden 1 bis 6 zur Lösung der Probleme 1 bis

6, d. h. Artefakte im Sinne der obigen Definition, vorgestellt. In Kapitel 5 werden die Methoden auf industrienahe Anwendungsfälle angewandt. Hierdurch wird ihre grundsätzliche Anwendbarkeit demonstriert und ihre Effektivität quantifiziert. In Kapitel 6 werden diese Ergebnisse diskutiert und auf Basis dessen die Forschungsfragen 1 bis 6 beantwortet. Darüber hinaus wird ein Ausblick auf den weiteren Entwicklungsbedarf gegeben, der im Sinne des Design Science Research Process als Grundlage für weitere Entwicklungszyklen dient. Abschließend wird die Arbeit in Kapitel 7 zusammengefasst. Die Kommunikation der aus der Arbeit hervorgegangenen Erkenntnisse im Sinne des Design Science Research Process erfolgte bereits durch Veröffentlichung eines ersten Ansatzes (Frey et al. 2023) und wird durch die vorliegende Arbeit, welche öffentlich zugänglich ist, komplementiert.

2 Grundlagen

Wie zuvor motiviert, wird durch die vorliegende Arbeit ein Beitrag zur Unterstützung der Arbeitsplanung durch die datenbasierte Erstellung und Überprüfung von Modellen zur Produkt- und Prozesskonfiguration geleistet. Zu diesem Zweck werden u.a. Verfahren des Maschinellen Lernens (ML) eingesetzt. Dementsprechend werden im Folgenden die wichtigsten Grundlagen der Arbeitsplanung (Kapitel 2.1), der Konfiguration (Kapitel 2.2) und des ML (Kapitel 2.3) dargestellt.

2.1 Arbeitsplanung

Die Arbeitsplanung in Industrieunternehmen hat die Aufgabe, „die erforderlichen Verfahren, Betriebsmittel und Abläufe [festzulegen], um ein Erzeugnis zu fertigen oder eine Dienstleistung auszuführen“ (Wiendahl 2019, S. 189). Sie stellt neben der Arbeitssteuerung, die die planmäßige Durchführung der Produktion sicherstellt, eine Teilaufgabe der **Arbeitsvorbereitung** dar (Eversheim 2002, S. 1–7). Die Arbeitsplanung kann wiederum in die Arbeitssystemplanung und die Arbeitsablaufplanung unterteilt werden (Westkämper 2006, S. 155). Abbildung 2.1 zeigt die Gliederung der relevanten Begriffe.

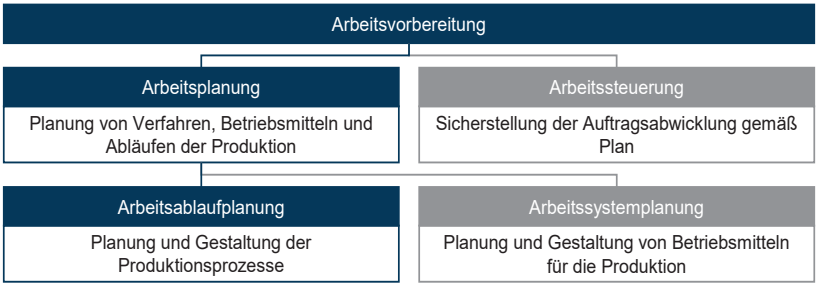


Abbildung 2.1: Aufgaben der Arbeitsvorbereitung (Eigene Darstellung nach Wiendahl 2019, S. 189, Westkämper 2006, S. 155 und Eversheim 2002, S. 1–7)

Die Aufgaben der Arbeitsplanung lassen sich darüber hinaus nach ihrer Fristigkeit in kurz-, mittel- und langfristig einteilen (Wiendahl 2019, S. 189–190). Die kurzfristigen Aufgaben der Arbeitsplanung können weitgehend der Arbeitsablaufplanung zugeordnet werden. Sie umfassen die Arbeitsplanerstellung, die Erstellung von Fertigungsstücklisten (sog. Stücklistenverarbeitung), die NC-Programmierung sowie die Fertigungshilfsmittelplanung (Wiendahl 2019, S. 190). Der Einsatz von Konfigurationssystemen im Rahmen der Arbeitsplanung ist im Wesentlichen für die Erstellung von **Stücklisten**

(STLs) und **Arbeitsplänen (APLs)** und damit in der Arbeitsablaufplanung relevant. Es kann somit präzisiert werden, dass die vorliegende Arbeit sich mit Modellen zur Produkt- und Prozesskonfiguration im Rahmen der **Arbeitsablaufplanung** befasst. Die beiden hierfür zentralen Begriffe STL und APL werden im Folgenden näher erläutert.

STLs enthalten die eindeutig identifizierten Komponenten einer Baugruppe oder eines montierbaren Produkts mit Menge und Einheit (ISO 7573:2008-11, S. 2–4). In der vorliegenden Arbeit werden sowohl Einzelteile, als auch montierte Gruppen von Einzelteilen zusammenfassend als Komponenten bezeichnet. Komponenten, die das betrachtete Unternehmen zukauft, werden als Zukaufkomponenten (ZKs) bezeichnet. Wird explizit der Begriff Baugruppe verwendet, ist damit eine Baugruppe gemeint, die im Unternehmen montiert wird und damit keine ZK ist. Neben ZKs und Baugruppen, die keine bzw. mehrere untergeordnete Komponenten in der STL aufweisen, können in der Praxis auch Komponenten mit genau einer untergeordneten Komponente in STLs auftreten. Dies kann z. B. der Fall sein, wenn eine bestimmte Komponente in verschiedenen Zuständen, wie z. B. vor und nach einer Wärmebehandlung, in der Stückliste erfasst wird. Dieser Fall wird jedoch in der vorliegenden Arbeit nicht betrachtet. STLs können mehrstufige Hierarchien aufweisen. Gängige Formate für STLs sind, wie in Abbildung 2.2 dargestellt, Mengenübersichtsstücklisten, Strukturstücklisten und Baukastenstücklisten (Wiendahl 2019, S. 159–164). Hinsichtlich ihrer Funktion sind u. a. Konstruktionsstücklisten, die im Rahmen der Produktentwicklung erstellt werden und Fertigungsstücklisten, die als Grundlage für die Produktionsdurchführung geeignet sind, zu unterscheiden (Eversheim 2002, S. 23). Aufgrund ihrer unterschiedlichen Funktion können sich Konstruktionsstücklisten und Fertigungsstücklisten u. a. in ihrer Struktur unterscheiden, d. h. in der Weise wie ZKs zu Baugruppen zusammengefasst werden

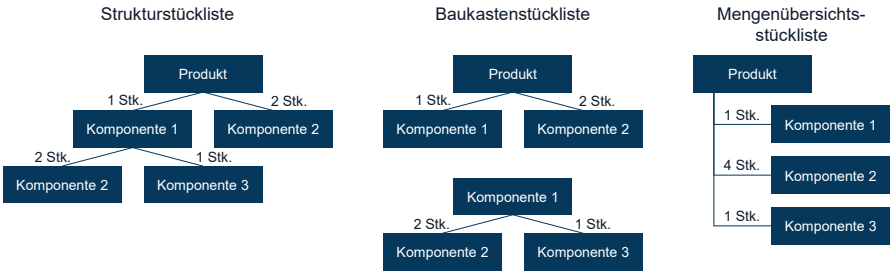


Abbildung 2.2: Gängige Formate für Stücklisten nach Wiendahl (2019, S. 159–164)

(Wiendahl 2019, S. 194). Baugruppen in Konstruktionsstücklisten fassen i. d. R. ZKs zusammen, die gemeinsam eine Produktfunktion abbilden wohingegen Fertigungsstücklisten Baugruppen definieren, die als Zwischenzustände in der Montage auftreten (Wiendahl 2019, S. 194). Bei den in der vorliegenden Arbeit betrachteten variantenbezogenen Stücklisten (VSTLs) handelt es sich um Fertigungsstücklisten.

Der **APL** ist das Ergebnis der Arbeitsplanerstellung und beschreibt „die Vorgangsfolge zur Herstellung eines Teils, einer Baugruppe oder eines Erzeugnisses“ (Wiendahl 2019, S. 191). Ein APL besteht i. d. R. aus mehreren Arbeitsvorgängen (AVOs), für die jeweils mindestens das verwendete Material, der Arbeitsplatz, die Betriebsmittel und die Vorgabezeiten hinterlegt sind (Wiendahl 2019, S. 191). Ein AVO ist i. d. R. einem Arbeitsplatz zugeordnet und umfasst alle Arbeitsinhalte, die für einen Auftrag an diesem Arbeitsplatz ausgeführt werden (Eversheim 2002, S. 24). Da Arbeitsplätze in der industriellen Praxis mehr oder weniger umfassend abgegrenzt werden, können AVOs in der Industrie mehr oder weniger großen Teilen des gesamten Herstellprozesses entsprechen. Die Arbeitsinhalte eines AVOs können weiter in Operationen unterteilt werden, wodurch der APL verfeinert wird (Wiendahl 2019, S. 195). Der Detaillierungsgrad der Arbeitsplanung ist das Ergebnis einer Aufwand-Nutzen-Abwägung und kann von Fall zu Fall unterschiedlich sein (Eversheim 2002, S. 1). In der vorliegenden Arbeit wird nicht zwischen AVOs und Operationen unterschieden, sondern jeweils der Begriff AVO verwendet. Je nach Anwendungsfall kann ein AVO somit mehr oder weniger Arbeitsinhalt umfassen.

2.2 Konfiguration

Im Folgenden werden zunächst die zentralen Begriffe zu Konfiguration im industriellen Kontext, die für die vorliegende Arbeit relevant sind, eingeführt (Kapitel 2.2.1). Anschließend wird auf Systeme und Modelle der Vertriebs-, Produkt- und Prozesskonfiguration sowie auf integrierte Konfigurationsmodelle näher eingegangen (Kapitel 2.2.2). Zuletzt wird entsprechend des Rahmens der vorliegenden Arbeit die Erstellung und Überprüfung dieser Modelle thematisiert (Kapitel 2.2.3).

2.2.1 Zentrale Begriffe der Konfiguration

Wie Oddsson & Ladeby (2014, S. 419–422) zeigen, gibt es in der Literatur ähnliche, wenn auch z .T. nicht übereinstimmende, Definitionen der zentralen Begriffe Konfiguration, Konfigurationsmodell (KM) und Konfigurationssystem (KS). Auf Basis einer

umfassenden Literaturanalyse erarbeiten sie Begriffsdefinitionen, an denen sich die vorliegende Arbeit orientiert. Der Begriff Konfiguration meint zum einen den Prozess des Konfigurierens und zum anderen das Ergebnis dieses Prozesses (Felfernig et al. 2014, S. 6). In der vorliegenden Arbeit werden dementsprechend überall dort die Begriffe Konfigurationsprozess und Konfigurationsergebnis verwendet, wo dies für das Verständnis erforderlich ist.

Der Konfigurationsprozess löst die **Konfigurationsaufgabe**. Darunter verstehen Oddsson & Ladeby (2014, S. 420) die Kombination vordefinierter Elemente sowie das Festlegen von deren Eigenschaften unter Berücksichtigung von Beschränkungen und zulässigen Schnittstellen, sodass gegebene Anforderungen erfüllt werden. Wie Oddsson & Ladeby (2014, S. 420) ausführen, kann das Ergebnis des Konfigurationsprozesses auch Beziehungen zwischen den Elementen enthalten. Je nach betrachteter Konfigurationsaufgabe können Elemente im Sinne der Definition z. B. Komponenten und Beziehungen z. B. **meronymische Beziehungen**, d. h. Ist-Bestandteil-von-Beziehungen, sein. Die Eigenschaften der vordefinierten Elemente, im Folgenden auch als Parameter bezeichnet, und deren Definitionsbereiche sind i. d. R. ebenfalls vordefiniert (siehe Beispiel von Felfernig et al. 2014, S. 56). In der vorliegenden Arbeit wird auch das Vorhandensein oder Nichtvorhandensein eines Elements in einer Kombination ebenfalls als Parameter des Elements verstanden. Beschränkungen bestehen somit immer zwischen Parametern, können jedoch auch ausdrücken, dass sich das Vorhandensein verschiedener Elemente gegenseitig bedingt oder ausschließt (angelehnt an Hvam et al. 2008, S. 214–215). Abweichend von Oddsson & Ladeby (2014) wird in der Literatur nicht nur der Begriff Beschränkungen verwendet, sondern zwischen Beschränkungen und Regeln unterschieden (beispielsweise von Hvam et al. 2008, S. 211–215)[8]. Eine Regel legt die Ausprägung eines Parameters auf Basis der Ausprägung eines bestimmten anderen Parameters fest und ist somit unidirektional (Hvam et al. 2008, S. 211–214). Regeln besitzen für die vorliegende Arbeit eine herausragende Bedeutung. Im Folgenden wird deshalb ebenfalls zwischen **bidirektionale**n **Beschränkungen** und **unidirektionalen Regeln** unterschieden und übergeordnet der Begriff Abhängigkeiten verwendet. Vor dem gegebenen Hintergrund ist der Konfigurationsprozess in der vorliegenden Arbeit auf Basis der Definition einer Konfigurationsaufgabe nach Oddsson &

[8] Kapitel 2.2.2.2 geht auf diese Begriffe im Zusammenhang mit Produktkonfiguration genauer ein.

Ladeby (2014) wie folgt definiert: Die Konfiguration im Sinne des **Konfigurationsprozesses** umfasst

- die Auswahl von Elementen aus einer vordefinierten Menge von Elementen,
- die Festlegung der Ausprägungen der vordefinierten Parameter dieser Elemente im Rahmen vordefinierter Definitionsbereiche,
- und die Festlegung der Beziehungen zwischen den Elementen aus einer vordefinierten Menge möglicher Beziehungen

unter Berücksichtigung von Abhängigkeiten, sodass gegebene Anforderungen erfüllt werden. Entsprechend der üblichen Begriffsverwendung in der Literatur (beispielsweise Zhang et al. 2010, S. 213) wird in der vorliegenden Arbeit davon gesprochen, dass durch den Konfigurationsprozess der betrachtete Gegenstand wie insbesondere ein Produkt oder ein Prozess **konfiguriert** wird. Die Konfiguration im Sinne des **Konfigurationsergebnisses** besteht aus ausgewählten Elementen mit festgelegten Parameterausprägungen, die zueinander in festgelegten Beziehungen[9] stehen.

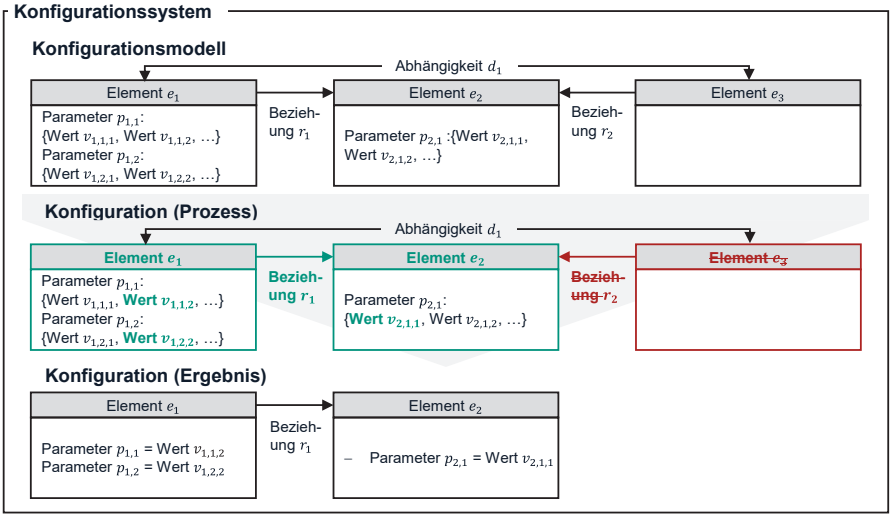


Abbildung 2.3: Schematische Darstellung der relevanten Begriffe im Kontext industrieller Konfigurationssysteme

[9] Beziehungen sind damit in der vorliegenden Arbeit von Abhängigkeiten zu unterscheiden, auch wenn die beiden Begriffe in der Literatur nicht scharf voneinander abgegrenzt sind. Beziehungen zwischen Elementen des

Angelehnt an die Definition von Produktkonfigurationsmodellen nach Oddsson & Ladeby (2014, S. 420) enthält ein **Konfigurationsmodell** eine Menge vordefinierter Elemente mit vordefinierten Parametern mit vordefinierten Definitionsbereichen. Darüber hinaus enthält es die möglichen Beziehungen zwischen den Elementen sowie die gültigen Abhängigkeiten zwischen deren Parametern. Angelehnt an die Definition von Produktkonfigurationssystemen nach Oddsson & Ladeby (2014, S. 422) werden in der vorliegenden Arbeit software-basierte Systeme, die den Nutzer dabei unterstützen auf Basis eines hinterlegten KM eine Konfiguration vorzunehmen als **Konfigurationssysteme** bezeichnet. Der Begriff **Konfigurator** wird synonym hierfür verwendet. Abbildung 2.3 stellt die eingeführten Begriffe anhand eines abstrahierten KM dar[10]. Auf die konkrete Ausgestaltung von KMs wird in den folgenden Kapiteln eingegangen.

Nach Hvam et al. (2008, S. 15–27) sind in Industrieunternehmen u. a. Spezifikationen von Angeboten, Produkten und Produktionsprozessen zu erstellen. Unternehmen, die dem Prinzip der auftragsbezogenen Entwicklung (**Engineer-to-Order**, ETO) folgen, erstellen diese Spezifikationen nach Auftragseingang (Hvam et al. 2008, S. 28–30). Unternehmen, die dem Prinzip der auftragsbezogenen Konfiguration (**Configure-to-Order**, CTO) folgen, erstellen hingegen auftragsunabhängige Spezifikationen, die als Basis für die Erstellung von auftragsbezogenen Spezifikationen mittels Konfiguration dienen (Hvam et al. 2008, S. 28–30). Die auftragsunabhängig erstellten Spezifikationen entsprechen hierbei den oben beschriebenen Elementen und Abhängigkeiten des Konfigurationsmodells. Im Gegensatz zur ETO existiert damit für die CTO ein definierter und geschlossener Lösungsraum (Kourtis et al. 2024, S. 154). Zwischen den Idealtypen ETO und CTO findet sich in der Praxis ein Kontinuum für den Grad der auftragsunabhängig erstellten Spezifikationen (Hvam et al. 2008, S. 28–30). Beispiele hierfür sind Unternehmen, die, wie in Kapitel 1.1 und 1.2 beschrieben, Konfigurationssysteme ausschließlich im Vertrieb nutzen.

2.2.2 Typen von Konfigurationssystemen und -modellen

KSs können nach ihrem Einsatzzweck in Vertriebs-, Produkt-, Prozess- und Instandhaltungskonfigurationssysteme eingeteilt werden (Zhang et al. 2020, S. 1–3).

Konfigurationsmodells werden teilweise oder vollständig in das Konfigurationsergebnis übernommen. Abhängigkeiten hingegen schränken die Möglichkeiten im Konfigurationsprozess ein und liegen im abschließenden Konfigurationsergebnis nicht vor.

[10] Die Größen d, e, p, r und v stehen als Platzhalter für bestimmte Abhängigkeiten (d), Elemente (e), Parameter (p), Beziehungen (r) bzw. Parameterausprägungen (v).

Instandhaltungskonfigurationssysteme sind nicht Gegenstand der vorliegenden Arbeit. Vertriebs-, Produkt- und Prozesskonfigurationssysteme werden im Folgenden erläutert, wobei der Fokus auf den zugehörigen KMs liegt.

2.2.2.1 Vertriebskonfigurationssysteme und -modelle

Vertriebskonfigurationssysteme werden häufiger als **Vertriebskonfiguratoren** (engl. sales configurator oder commercial configurator) bezeichnet. Sie unterstützen einen Kunden oder einen Vertriebsmitarbeiter, der mit dem Kunden interagiert, dabei ein Produkt vollständig und korrekt mittels Konfiguration zu spezifizieren, d. h. zu konfigurieren (Trentin et al. 2014, S. 694). Die Konfiguration erfolgt i. d. R. durch das Ausprägen von **Produktmerkmale**n[11], wobei unzulässige Kombinationen von Merkmalausprägungen durch hinterlegte Beschränkungen ausgeschlossen sind (Abbasi et al. 2013, S. 166–170). Insbesondere bei der Gestaltung von Vertriebskonfiguratoren für die direkte Interaktion mit dem Kunden ist darauf zu achten, dass die abgefragten Merkmale dem Nutzer ermöglichen, seine Anforderungen zu artikulieren (Salvador & Forza 2007, 117ff). Die **Vertriebskonfiguration** stellt damit nicht zwingend eine technische Spezifikation des Produkts dar, sondern erfolgt u. U. auf Ebene der **Kundenanforderungen** oder der **Produktfunktionen** (Salvador & Forza 2007, S. 117–119). Für eine konfigurierbare Pumpe kann z. B. aus Sicht des Kunden ein Produktmerkmal sinnvoll sein, das das geförderte Medium beschreibt. Dieses Produktmerkmal kann sich in vielfältiger Weise auf die technische Spezifikation des Produkts – wie z. B. auf die Auswahl der verwendeten Werkstoffe für verschiedene Komponenten – niederschlagen. Die Übersetzung einer Vertriebskonfiguration in eine technische Spezifikation ist insbesondere in diesen Fällen nicht trivial (siehe auch Anwendungsfall von Salvador & Forza 2007, S. 125). Durch die definierten Merkmale und deren Definitionsbereiche sowie die hinterlegten Beschränkungen, definiert das in Vertriebskonfiguratoren hinterlegte Vertriebskonfigurationsmodell einen **Konfigurationsraum**. Jede mögliche Vertriebskonfiguration aus dem Konfigurationsraum entspricht einer möglichen Variante des konfigurierbaren Produkts. Die Vertriebskonfiguration ist die Basis für den in der vorliegenden Arbeit betrachteten Produkt- und Prozesskonfigurationsprozess. In manchen Fällen schließt die Vertriebskonfiguration in der Praxis auch bereits die Beschreibung von

[11] Im Folgenden bezeichnen Produktmerkmale (kurz: Merkmale) Eigenschaften des Produkts aus Sicht des Kunden und sind Teil der Angebotsspezifikation. Mit Parametern werden demgegenüber im Folgenden Eigenschaften des Produkts oder seiner Komponenten bezeichnet, die im Rahmen der technischen Spezifikation festgelegt werden (siehe Kapitel 2.2.1).

Komponenten des Produktes ein, wodurch sich eine Vertriebsstückliste ergibt (Brinkop et al. 2012, S. 10). Somit kann ein fließender Übergang zur Produktkonfiguration bestehen. Vertriebskonfiguratoren verfügen i. d. R. über zusätzliche Funktionen, die den Vertrieb unterstützen, wie insbesondere die Ermittlung von Preisen und Lieferkonditionen sowie eine Angebotserstellung (Trentin et al. 2012b, S. 47). Diese sind jedoch für die vorliegende Arbeit nicht relevant.

2.2.2.2 Produktkonfigurationssysteme und -modelle

Der Begriff Produktkonfiguration wird in der Literatur in einem engen und einem weiten Sinne verwendet. Autoren wie z. B. Trentin et al. (2012a) nutzen den Begriff Produktkonfiguration in einem weiten Sinne für Konfigurationsaufgaben in verschiedenen Bereichen von Industrieunternehmen einschließlich Vertrieb und Produktion. In der vorliegenden Arbeit wird jedoch in einem engen Sinne in Anlehnung an Zhang et al. (2013, S. 465–466) unter **Produktkonfiguration** die Konfiguration eines Produkts aus technischer Sicht und damit insbesondere die Auswahl und Parametrierung seiner Komponenten verstanden[12]. Damit ist die Produktkonfiguration von der Vertriebskonfiguration und der Prozesskonfiguration zu unterscheiden.

Gängige Produktkonfigurationsmodelle in der Literatur sind **objektorientiert**[13] und lassen sich als Verbindung einer Strukturdarstellung und einer zugehörigen Semantik beschreiben. Für die Strukturdarstellung werden insbesondere Klassendiagramme in Unified Modelling Language (UML), Produktvarianten-Master und Featuremodelle (Felfernig et al. 2014, S. 52–59; Haug et al. 2010, S. 410–412) verwendet. Abbildung 2.4 zeigt beispielhaft eine Strukturdarstellung eines Produktmodells in Form eines UML-Klassendiagramms (1) und eines Produktvarianten-Masters (2). Für Featuremodelle sei auf Felfernig et al. (2014, S. 52–54) verwiesen. Die Semantik kann sich einerseits aus der Strukturdarstellung selbst ergeben. Die Darstellung in Abbildung 2.4 impliziert z. B., dass sich Scheiben- und Felgenbremsen als Instanzen derselben Klasse gegenseitig ausschließen. Andererseits kann sie in Form von Abhängigkeiten beschrieben werden (siehe Kapitel 2.2.1 und Beispiel von Felfernig et al. 2014, S. 52–59).

---

[12] In der vorliegenden Arbeit ist die Produktkonfiguration auf die Konfiguration von Stücklisten beschränkt. Die Konfiguration weiterer technischer Dokumente, wie insbesondere technischer Zeichnungen, wird nicht betrachtet.
[13] Auch wenn Objektorientierung in kommerziellen KS oftmals nicht vollständig umgesetzt ist, lässt sich ein objektorientiertes Modell darin weitgehend realisieren (siehe Rasmussen et al. 2020), weshalb der Fokus an dieser Stelle auf objektorientierten Modellen liegt.

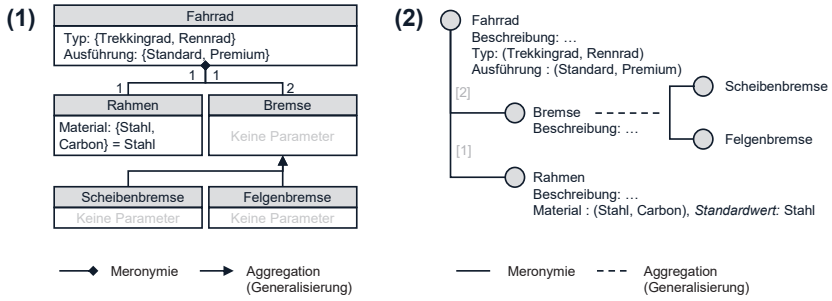


Abbildung 2.4: Beispielhafte Strukturdarstellung eines Produktkonfigurationsmodells als UML-Klassendiagramm (1) und Produktvarianten-Master (2)

Eine Abhängigkeit in Form einer **Regel** könnte z. B.

$$\left(Fahrrad.Typ^{14} = Rennrad\right) \vee \left(Fahrrad.Ausführung = Premium\right) \Rightarrow \left(Rahmen.Material = Carbon\right) \quad 2.1$$

und eine Abhängigkeit in Form einer **Beschränkung** z. B.

$$\left(Fahrrad.Typ = Rennrad\right) \vee \left(Fahrrad.Ausführung = Premium\right) \leftrightarrow \left(Rahmen.Material = Carbon\right) \quad 2.2$$

lauten. Im ersten Fall müssten zwingend zunächst die Parameter Typ und Ausführung festgelegt werden. Anschließend würde ein Rahmen aus Carbon gewählt werden, sofern ein Rennrad oder ein Fahrrad in Premiumausführung gewählt wurden. Andernfalls wäre das Rahmenmaterial nach Standardwert Stahl. Im zweiten Fall können die Parameter Typ, Ausführung und Rahmenmaterial in beliebiger Reihenfolge beliebig gewählt werden. Es muss nur gewährleistet sein, dass abschließend die Beschränkung eingehalten wird. Regelbasierte Modelle werden in der Literatur zum Teil kritisch gesehen (beispielsweise von Mailharro 1998, S. 384). Sie sind aber in der Praxis gebräuchlich, wie in Kapitel 2.2.2.4 ausgeführt wird. Deshalb werden sie in der vorliegenden Arbeit betrachtet. Abhängigkeiten, die im Konfigurationsmodell hinterlegt sind, können in die Strukturdarstellung des Konfigurationsmodells aufgenommen (siehe Beispiel von Felfernig et al. 2014, S. 52–59) oder in separaten Dateien, wie z. B. Class-Responsibility-

[14] Hier und im Folgenden bezeichnet die Schreibweise [Klasse/Objekt].[Attribut] eine Referenz auf das Attribut einer Klasse oder eines Objekts wie in der objektorientierten Programmierung üblich.

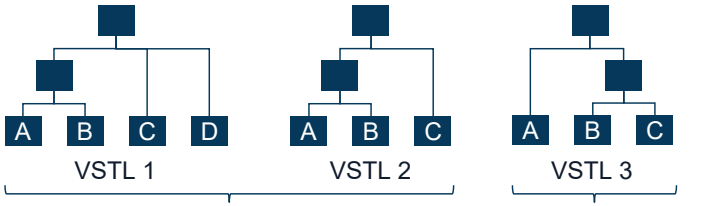
Collaboration-Karten[15], abgebildet werden.

Modelle zur Konfiguration von VSTLs folgen in ihrer Strukturdarstellung i. d. R. dem Aufbau einer STL. Es existieren verschiedene Strukturdarstellungen in Form von Stücklisten wie z. B. modulare Stücklisten, Variantenstücklisten, generische Stücklisten und Maximalstücklisten für deren vollständige Erläuterung auf Jiao et al. (2000, S. 299–300) verwiesen sei. Nach Jiao et al. (2000, S. 299–300) enthalten generische Stücklisten ausschließlich Generalisierungen – d. h. **Aggregationen** von Komponentenklassen (KKs) wie die Klasse Bremse im Beispielfall – und meronymische Beziehungen. Demgegenüber enthalten Maximalstücklisten (MSTLs) nach Jiao et al. (2000, S. 299–300) mit Verweis auf Kneppelt (1984) bestimmte Komponenten, die in den VSTLs auftreten können. Diesen wird je nach gewählter Variante eine Liste mit hinzuzufügenden oder zu entfernenden Komponenten beigefügt. Abweichend hiervon folgt die vorliegende Arbeit für MSTLs dem Verständnis[16] von Ram Babu et al. (2014, S. 96) wonach eine **Maximalstückliste** alle Komponenten enthält, die über alle Varianten hinweg auftreten können. In der vorliegenden Arbeit werden Maximalstücklisten objektorientiert beschrieben. Sie bestehen aus einer Klasse je Komponente, die in einer möglichen Variante auftreten kann, sowie einer Klasse, die das Produkt selbst repräsentiert. Eine Klasse wird im Rahmen des Konfigurationsprozesses als Objekt **instanziiert**, falls sie für eine bestimmte Variante benötigt wird; andernfalls wird sie nicht instanziiert (angelehnt an Felfernig et al. 2014, S. 55–59). KKs definieren somit, welche Komponenten – und damit welche Elemente im Sinne der in Kapitel 2.2.1 gegebenen Definition – in einer VSTL auftreten können. Generalisierungen sind möglich und werden jeweils durch eine ihrer aggregierten Klassen instanziiert. Die in Abbildung 2.4 gezeigte Strukturdarstellung entspricht einer Maximalstückliste. In diesem Beispiel könnte ein Fahrrad konfiguriert werden, indem ein Produkt der Klasse Fahrrad mit Typ Rennrad und Ausführung Premium, eine Komponente der Klasse Rahmen mit Material Carbon und eine Komponente der Klasse Scheibenbremse instanziiert würden. Ihre Instanzen werden Teil der VSTL. In diesem Fall könnte die Klasse Felgenbremse nicht instanziiert werden, da sie derselben Superklasse wie die Klasse Scheibenbremse zugeordnet ist. Wie in Kapitel 4.1.1 ausgeführt wird, besteht ein Produktkonfigurationsmodell in der

[15] Für Class-Responsibility-Collaboration-Karten sei auf Hvam et al. (2008, S. 186–195) verwiesen.
[16] Dieses Verständnis liegt auch den Konfigurationssystemen der SAP SE zu Grunde. Siehe https://help.sap.com/docs/SAP_S4HANA_ON-PREMISE/a73402f511734e6eac56063e631bf24e/c162b6531de6b64ce10000000a174cb4.html (zuletzt überprüft am 07.06.2025)

vorliegenden Arbeit aus einer MSTL sowie aus Regeln, die die Elemente der VSTL und deren Ausprägungen festlegen.

Meronymische Beziehungen zwischen Klassen der MSTL spiegeln sich in Komponenten der VSTLs wider (siehe Beispiel von Felfernig et al. 2014, S. 55–59), d. h. durch das Produktkonfigurationsmodell sind die Baugruppen und damit die Fügereihenfolge festgelegt. Die Fügereihenfolge eines Produkts kann jedoch innerhalb eines Unternehmens von Fall zu Fall unterschiedlich sein (Romanowski & Nagi 2004, S. 317–318). Unterschiede können sich z. B. für verschiedene Produktionssysteme aufgrund unterschiedlicher Fügemittel, für verschiedene Varianten aufgrund unterschiedlicher Zugänglichkeit der Fügestelle (Jiménez 2013, S. 240–242) und zu verschiedenen Zeitpunkten aufgrund von Änderungen ergeben. Dementsprechend können VSTLs für das gleiche Produkt von unterschiedlichen Fügereihenfolgen ausgehen und somit alternative Strukturen aufweisen, wie in Abbildung 2.5 veranschaulicht. VSTL 1 und VSTL 2 unterscheiden sich in den benötigten ZKs, weisen jedoch keine Widersprüche in der Fügereihenfolge auf. VSTL 3 hingegen weist eine andere Struktur auf, da die ZKs B und C vor den ZKs A und B gefügt werden. In der vorliegenden Arbeit wird in einem solchen Fall der Begriff **Strukturalternative** (STA) verwendet. VSTL 1 und 2 entsprechen somit einer anderen STA des Produkts als VSTL 3. VSTLs die verschiedenen STAs entsprechen, können sich in mehreren meronymischen Beziehungen zugleich unterscheiden. Entsprechend der oben genannten Ursachen können zwei STAs in VSTLs einerseits technisch begründet sein, womit beide zugleich gültig sind. Andererseits können sie durch inkonsistente Planungsstände oder schlicht menschliche Fehler



VSTL = Variantenbezogene Stückliste

Abbildung 2.5: Strukturalternativen in variantenbezogenen Stücklisten

begründet sein, womit evtl. nur eine oder keine der beiden aktuell gültig ist. Werden VSTLs für die datenbasierte Erstellung von Produktkonfigurationsmodellen verwendet, müssen STAs in VSTLs erkannt und, falls sie technisch begründet sind, im KM abgebildet werden.

2.2.2.3 Prozesskonfigurationssysteme und -modelle

Die Entwicklung von Systemen für die Produktkonfiguration reicht bis in die 1970er Jahre zurück (Hotz et al. 2014, S. 9). Demgegenüber wurde das Konzept der Prozesskonfiguration in der Literatur erst Anfang der 2000er Jahre durch Schierholt (2001) umfassend beschrieben[17]. Als Grund hierfür sieht Schierholt, dass lange Zeit bei der Produktkonfiguration von Montageprodukten ausgegangen worden sei, bei denen die Montageprozesse durch die STL determiniert seien (Schierholt 2001, S. 412). Er weist darauf hin, dass dies bei der **Teilefertigung** nicht der Fall sei und betrachtet beispielhaft die Fertigung von Blechen und Coils (Schierholt 2001, S. 412–423). Die durchzuführenden Teilprozesse hängen in diesem Anwendungsfall u. a. von der Breite des zu fertigenden Blechs sowie von dessen Werkstoff ab, d. h. es wird eine Prozesskonfiguration auf Basis der Merkmale des Endprodukts durchgeführt. Es sei an dieser Stelle angemerkt, dass sich auch für **Montageprodukte** die Prozesskonfiguration nicht zwangsläufig trivial aus der Produktkonfiguration ergibt. Auch wenn durch eine Baugruppe der VSTL i. d. R. festgelegt ist, dass ein Fügevorgang für die Komponenten der Baugruppe erforderlich ist, ist damit noch nicht das Fügeverfahren[18] determiniert. Darüber hinaus können im Rahmen der Montage weitere Verfahren erforderlich sein, die sich nicht aus den Komponenten der Baugruppen ergeben. Beispiele hierfür sind das Füllen, Beschichten, Justieren, Prüfen (DIN 8580:2022-12, S. 13) oder Aufspielen von Software. Zuletzt können auch Fälle auftreten, in denen auch die Durchführung eines Fügevorgangs nicht durch die Baugruppe alleine determiniert ist. Beispielsweise kann die Vertriebskonfiguration vorsehen, dass auf Wunsch des Kunden ein bestimmtes Bauteil in der Montage der Baugruppe nur beigelegt, aber nicht gefügt wird, weil der Kunde selbst eine entsprechende Montage vornimmt. In der vorliegenden Arbeit wird deshalb sowohl die Produkt- als auch die Prozesskonfiguration für Produkte der Teilefertigung und der

[17] Die Möglichkeit Arbeitspläne in Abhängigkeit der gewählten Variante zu konfigurieren wird jedoch bereits einige Jahre zuvor von Haag (1998, S. 78) als Funktion von SAP R/3 erwähnt.
[18] Für Fügeverfahren sei auf Deutsches Institut für Normung e.V. (2022, S. 13) verwiesen

Montage betrachtet. Die beiden Fälle werden im Folgenden nur insoweit explizit unterschieden, als dies notwendig ist.

Ausgehend von der Arbeit von Schierholt (2001) werden in der Literatur Prozesskonfigurationsmodelle analog zu bestehenden Produktkonfigurationsmodellen beschrieben. Die Elemente dieser Modelle entsprechen den Teilprozessen des Produktionsprozesses. Sie sind entweder Fertigungstechnologien wie im Falle von Schierholt (2001, S. 412) oder AVOs wie in Kapitel 2.1 beschrieben (beispielsweise auch bei Wang et al. 2017). Entsprechend sind die **Ergebnisse der Prozesskonfiguration** Technologieketten bzw. variantenbezogene Arbeitspläne (VAPLs). Die vorliegende Arbeit beschränkt sich auf Prozesskonfiguration im Rahmen der Arbeitsablaufplanung zur Erstellung von VAPLs. Prozesskonfigurationsmodelle können analog zu Produktkonfigurationsmodellen als Verbindung einer objektorientierten Strukturdarstellung und einer Semantik beschrieben werden. Abbildung 2.6 zeigt ein einfaches Beispiel für ein Prozesskonfigurationsmodell. Für ein umfassendes Prozesskonfigurationsmodell nach Stand der Forschung sei auf Wang et al. (2017, S. 957) verwiesen.

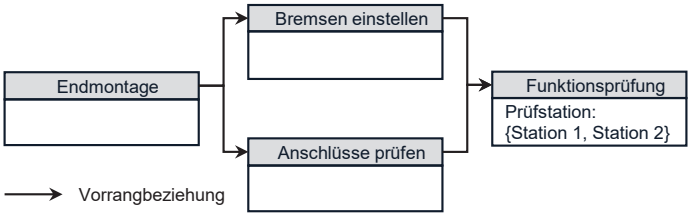


Abbildung 2.6: Beispielhafte Darstellung eines Prozesskonfigurationsmodells

Analog zu MSTLs liegt der vorliegenden Arbeit das Verständnis von **Maximalarbeitsplän**en (MAPLs) nach Ram Babu et al. (2014, S. 96) zugrunde, wonach MAPLs alle AVOs, die für eine zulässige Variante auftreten können, enthalten. Im Sinne der Objektorientierung werden in der vorliegenden Arbeit AVOs in MAPLs als Klassen – im Folgenden Arbeitsvorgangsklassen (AVKs) genannt – beschrieben. Diese werden im Zuge der Konfiguration in Abhängigkeit der Variante ggf. instanziiert. Durch die Klassen sind die AVOs und deren Parameter als wählbare Elemente im Sinne der Definition aus Kapitel 2.2.1 vorgegeben. Zwischen AVKs können Vorrangbeziehungen bestehen, welche sich in den VAPLs widerspiegeln und parallele Ausführungen von AVOs – wie in Abbildung 2.6 zu sehen –zulassen können. Wie in Kapitel 4.1.1 ausgeführt wird, besteht

ein Prozesskonfigurationsmodell in der vorliegenden Arbeit aus einem MAPL sowie aus Regeln, die die Elemente des VAPL und deren Ausprägungen festlegen.

Ebenso wie in VSTLs können auch in VAPLs **STAs** auftreten und müssen bei der datenbasierten Erstellung von Prozesskonfigurationsmodellen berücksichtigt werden. Beispielsweise unterscheiden sich die VAPLs 1 und 2 in Abbildung 2.7 in den benötigten AVOs, weisen jedoch keine Widersprüche in ihrer Struktur auf. VAPL 3 hingegen sieht vor, dass B zwingend vor C ausgeführt werden muss und entspricht damit einer anderen STA.

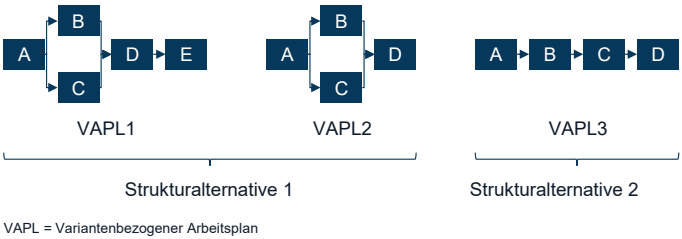


Abbildung 2.7: Strukturalternativen in variantenbezogenen Arbeitsplänen

2.2.2.4 Integrierte Vertriebs-, Produkt- und Prozesskonfigurationssysteme und -modelle

In der Praxis können die vorgestellten Vertriebs-, Produkt- und Prozesskonfigurationsmodelle als integrierte Modelle auftreten, auf die im Folgenden eingegangen wird. Aufbauend auf bestehenden KMs entwickeln Zhang et al. (2013; 2020) integrierte Vertriebs-, Produkt- und Prozesskonfigurationsmodelle. In den Modellen existieren u. U. Beschränkungen innerhalb des Vertriebskonfigurationsmodells, d. h. bestimmte Ausprägungen der Produktmerkmale lassen sich nicht miteinander kombinieren. Je Klasse des Produktkonfigurationsmodells existiert ein Prozesskonfigurationsmodell, das die Montage und Bearbeitung des Produkts oder der entsprechenden Komponente beschreibt. Die Ausprägungen der Produktmerkmale beschränken das Produkt- und Prozesskonfigurationsmodell. Im Rahmen dieser Beschränkungen werden im Anschluss an den Vertriebskonfigurationsprozess automatisch mittels mathematischer Optimierung ein herstellkostenoptimales Produkt und herstellkostenoptimale Prozesse konfiguriert. Es findet somit keine vollständig integrierte Vertriebs-, Produkt- und Prozesskonfiguration statt, sondern zunächst eine Vertriebskonfiguration und anschließend eine

integrierte Produkt- und Prozesskonfiguration. Diese **sequenzielle Trennung** findet sich auch in kommerziell verfügbaren Systemen für die Vertriebs-, Produkt- und Prozesskonfiguration. Derjenige Teil des KS, der die Funktionen des Vertriebskonfigurators (siehe Kapitel 2.2.2.1) bereitstellt wird in diesem Kontext als **High-Level-Konfigurationssystem (HLKS)** bezeichnet (Haag 1998, S. 78). Derjenige Teil des KS, der die Funktionen eines integrierten Produkt- und Prozesskonfigurationssystems bereitstellt, wird demgegenüber als **Low-Level-Konfigurationssystem (LLKS)** bezeichnet (Haag 1998, S. 78). Außerdem werden in der vorliegenden Arbeit für die entsprechenden KMs die Begriffe **High-Level-Konfigurationsmodell (HLKM)** und **Low-Level-Konfigurationsmodell (LLKM)** verwendet.

Für LLKMs besteht ein Unterschied zwischen Stand der Forschung und Stand der Technik. Dies wird im Folgenden anhand der kommerziell verfügbaren Konfigurationssysteme LO-VC und AVC[19] der SAP SE (SAP) erläutert. In diesen Systemen werden für die Klasse des Produkts im HLKM und ggf. auch für bestimmte KKs sog. Konfigurationsprofile angelegt, die Abhängigkeiten, i. d. R. Beschränkungen, zwischen den Klassenmerkmalen enthalten. Auf Basis der Konfigurationsprofile kann die High-Level-Konfiguration vorgenommen werden, welche als Ausgangspunkt für die Low-Level-Konfiguration dient. Das LLKM ist durch eine MSTL sowie einen APL je Komponente definiert. Der APL je Komponente beschreibt den Herstellprozess dieser Komponente, ggf. aus ihren Subkomponenten. Er kann ein MAPL sein, sofern der Herstellprozess der Komponente von der gewählten Produktvariante abhängt. Die Elemente der MSTL und der MAPLs können von der High-Level-Konfiguration oder von anderen Elementen der MSTL und der MAPLs abhängen. Als Abhängigkeiten können jedoch im LLKM keine Beschränkungen auftreten, sondern ausschließlich Regeln. Diese treten entweder in Form von Auswahlbedingungen oder in Form von Prozeduren auf. **Auswahlbedingungen** werden genutzt, um direkt oder indirekt in Abhängigkeit von der High-Level-Konfiguration zu entscheiden, ob eine Komponente oder ein Arbeitsvorgang Teil einer bestimmten VSTL bzw. eines bestimmten VAPL ist. **Prozeduren** werden genutzt, um Merkmale von Elementen auszuprägen. (Blumöhr et al. 2019, S. 153–259)[20]

---

[19] LO-VC (Logistik-Variant Configuration) ist ein Konfigurationssystem im Rahmen des SAP ERP; AVC (Advanced Variant Configuration) ist dessen Nachfolgeprodukt. Die Ausführungen gelten für beide Systeme. (zu SAP LO-VC siehe https://help.sap.com/docs/SAP_S4HANA_ON-PREMISE/a73402f511734e6eac56063e631bf24e/1a40b953495bb44ce10000000a174cb4.html?locale=de-DE (zuletzt überprüft am 07.06.2025)

[20] Steht in der vorliegenden Arbeit eine Quellenangabe hinter dem Satzzeichen wird der gesamte Absatz bis zu dieser Stelle nach dieser Quelle indirekt zitiert.

Damit entspricht das LLKM der SAP-Systeme nicht den von Zhang et al. (2020) beschriebenen Produkt- und Prozesskonfigurationsmodellen, welche beschränkungsbasiert sind und eine Konfiguration durch Optimierung vornehmen. Durch in der Literatur beschriebene Anwendungsfälle (siehe Anwendungsfälle von Braun 2021, S. 83–92 und Chatras et al. 2016, S. 5641) sowie online verfügbare Produktinformationen[21] kann bestätigt werden, dass dies auch für andere kommerziell verfügbare, integrierte KSs gilt. Gegenwärtig ist nicht bekannt, ob kommerziell verfügbare KSs zukünftig beschränkungsbasierte LLKMs, wie von Zhang et al. (2013; 2020) beschrieben, bereitstellen werden. Dazu wäre u. a. die Implementierung eines optimierungsbasierten Low-Level-Konfigurationsprozesses notwendig. Die vorliegende Arbeit orientiert sich deshalb an den gegenwärtig in der Praxis eingesetzten KSs mit **beschränkungsbasierte**n **HLKMs** und **regelbasierte**n **LLKMs**. Das datenbasiert zu erstellende KM ist damit in der vorliegenden Arbeit konkret ein LLKM nach Stand der Technik. Der Fokus hinsichtlich der datenbasierten Erstellung von Regeln liegt auf Auswahlbedingungen und nicht auf Prozeduren. Ein HLKM mit definierten Produktmerkmalen und Definitionsbereichen wird vorausgesetzt (siehe Kapitel 1.2).

2.2.3 Erstellung und Überprüfung von Konfigurationsmodellen

2.2.3.1 Erstellung von Konfigurationsmodellen

Die Erstellung von KMs ist von der Entwicklung von KSs abzugrenzen. Letzteres ist nicht Gegenstand der vorliegenden Arbeit. Es wird jedoch nicht ausgeschlossen, dass die Erstellung eines KM parallel zur Entwicklung eines KS erfolgt. Dies kann der Fall sein, wenn als KS keine Standardsoftware eingesetzt wird. Soll ein KS gemeinsam mit einer neuentwickelten Produktfamilie eingeführt werden, kann die Entwicklung der Produktfamilie und des KM integriert erfolgen. Für eine Beschreibung eines Entwicklungsprozesses für Produktfamilien, der das KM berücksichtigt sei auf Gauss et al. (2021) verwiesen. Darüber hinaus zeigen Hanna et al. (2023), wie Modelle, die für die modellbasierte Produktentwicklung eingesetzt werden, auch für die Produktkonfiguration genutzt werden können. In der vorliegenden Arbeit, ebenso wie in den existierenden

[21] Siehe z. B. Oracle CPQ (ehemals BigMachines): https://help.bigmachines.com/BMIHelp/Content/BOM_Mapping/BOM_Overview.htm?TocPath=Configuration%7CBOM%20Mapping%7C_____0 (zuletzt überprüft am 07.06.2025)
Configit Ace: https://configit.com/learn/tech-talks/bom-validation-delivering-on-the-configuration-promise/ (zuletzt überprüft am 07.06.2025)

Vorgehensmodellen zur Erstellung von KMs (siehe Shafiee et al. 2017, S. 988), wird die Erstellung von KMs jedoch unabhängig von einer Produktentwicklung betrachtet.

Auf Basis der existierenden **Vorgehensmodelle** zur Erstellung von KMs teilen Shafiee et al. (2017, S. 990–995) den Erstellungsprozess in vier Schritte ein (Abbildung 2.8, 1): die Festlegung des Umfangs des KM, die Erfassung von Wissen, die Modellierung und Validierung von Wissen sowie die Dokumentation und Instandhaltung. Der Umfang des KM wird hinsichtlich der zu erfüllenden Funktionen, der Funktionsweise und der einzuschließenden Produkte mittels Stakeholderanalyse festgelegt (Shafiee et al. 2017, S. 992–993). Für die Wissenserfassung wird das benötigte Wissen kategorisiert und es werden die relevanten Quellen und Ressourcen ermittelt (Shafiee et al. 2017, S. 991). Für die Wissensmodellierung schlagen Shafiee et al. (2017) die Verwendung von Produktvarianten-Mastern in Verbindung mit Class Responsibility-Collaboration-Karten zur standardisierten Beschreibung von Abhängigkeiten vor. Hvam et al. (2008, S. 34) beschreiben den Vorgang der Wissensmodellierung in Anlehnung an Duffy & Andreasen (1995, S. 30–31) ausführlicher als mehrstufigen Prozess (Abbildung 2.8, 2). In diesem Prozess wird, ausgehend von der realen Welt, zunächst ein **Phänomenmodell** in Form eines Produktvarianten-Masters erstellt, das die relevanten Aspekte des zu modellierenden Produkts enthält (Hvam et al. 2008, S. 34). Das Phänomenmodell muss zunächst keiner strengen Form genügen und kann damit so gestaltet werden, dass es für Domänenexperten leicht verständlich ist (Hvam et al. 2019, S. 4436). Für den Fall eines bestehenden und historisch gewachsenen Produktprogramms, von dem Hvam et al. (2008) ausgehen, wird das Phänomenmodell als Beschreibung des bestehenden Produktprogramms erstellt (Hvam et al. 2008, S. 139–170). Als nächstes wird das Phänomenmodell formalisiert, woraus ein **Informationsmodell** (Hvam et al. 2008, S. 34) als verfeinerter Produktvarianten-Master oder als UML-Diagramm resultiert (Hvam et al. 2008, S. 54). Abschließend wird das Informationsmodell in einem **Computermodell** umgesetzt (Hvam et al. 2008, S. 34), so dass ein ausführbares KS entsteht. Wie von

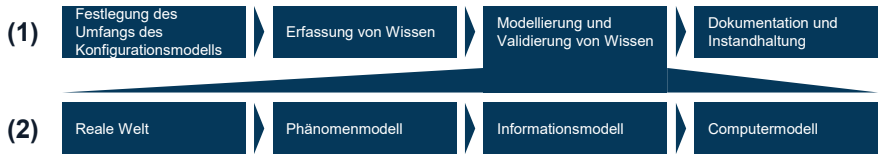


Abbildung 2.8: Vorgehensmodell zur Erstellung von Konfigurationsmodellen nach Shafiee et al. (2017, S. 990–995) und Duffy und Andreasen (1995, S. 30–31)

Shafiee et al. (2017, S. 991) beschrieben, kann im Zuge der Modellierung eine Überprüfung des Modells stattfinden (siehe Kapitel 2.2.3.2). Die im Rahmen des Erstellungsprozess entstehenden Modelle dienen abschließend als Grundlage für die Dokumentation des KM sowie seine Instandhaltung im Betrieb (Shafiee et al. 2017, S. 995).

Die vorliegende Arbeit stellt eine datenbasierte Alternative zu dem in der Literatur beschriebenen manuellen Prozess dar. Das zu erstellenden KM in Form einer MSTL, eines MAPL und der zugehörigen Regeln ist ein objektorientiertes Informationsmodell, das eine unmittelbare Umsetzung in ein Computermodell ermöglicht. Im Gegensatz zum klassischen Vorgehen wird jedoch nicht von Wissen über ein reales System, sondern von Beobachtungen in Form von Daten des realen Systems ausgegangen.

2.2.3.2 Überprüfung von Konfigurationsmodellen

Die Ansätze zur Überprüfung von KMs können in **Verifikation und Validierung** eingeteilt werden. Die Zuordnung bestehender Ansätze zur Überprüfung von KMs ist jedoch in der Literatur nicht eindeutig. Z. B. nutzen sowohl Voronov (2013, S. 185) als auch Braun (2021, S. 58–59) formale Prüfverfahren um automatisch zu überprüfen, ob es in einem KM Komponenten gibt, die überhaupt nicht ausgewählt werden können. Voronov (2013) ordnet dies der Verifikation und Braun (2021) der Validierung zu. Die vorliegende Arbeit verwendet ein Unterscheidungskriterium für Ansätze zur Überprüfung von KMs, das sich an Sinz (2004, S. 108–109) anlehnt: Wenn ein Ansatz Domänenwissen erfordert ist er der Validierung zuzuordnen, ansonsten der Verifikation. In diesem Sinne ist die im Rahmen der vorliegenden Arbeit entwickelte Methode zur Überprüfung von KMs (siehe Kapitel 4.6) der Validierung zuzuordnen. Neben der Einteilung in Verifikation und Validierung existiert eine Einteilung für die Ansätze zur Überprüfung von wissensbasierten Systemen nach Meseguer & Preece (1995, S. 337–339). Diese unterscheidet:

- **Inspektion**: Manuelle Überprüfung des Systems durch einen Experten
- **Statische Verifikation**: Automatische Überprüfung des Systems auf logische Widersprüche
- **Empirisches Testen**: Überprüfung des Systems durch Anwendung auf ausgewählte Beispiele und manuelle Überprüfung der Ergebnisse
- **Evaluation**: Untersuchung inwiefern das System die Anforderungen der Nutzer im Betrieb erfüllt

Diese Einteilung dient als Grundlage für die Einordnung des Stands der Forschung in Kapitel 3.6.2. In diesem Sinne ist die im Rahmen der vorliegenden Arbeit entwickelte

Methode zur Überprüfung von KMs (siehe Kapitel 4.6) als Methode zur Unterstützung der Inspektion einzuordnen.

2.3 Maschinelles Lernen

Im Folgenden werden zunächst die zentralen Begriffe des maschinellen Lernens eingeführt, die für die vorliegende Arbeit relevant sind (Kapitel 2.3.1). Auf Basis dessen werden die beiden Problemstellungen des überwachten (Kapitel 2.3.2) und unüberwachten Lernens (Kapitel 2.3.3) erläutert, die für die vorliegende Arbeit relevant sind.

2.3.1 Zentrale Begriffe des maschinellen Lernens

Nach DIN EN ISO/IEC 22989:2023-04 bezeichnet **maschinelles Lernen** einen „Prozess der Optimierung von Modellparametern durch computergestützte Verfahren, so dass das Verhalten des Modells den Daten oder Erfahrungen entspricht“ (DIN EN ISO/IEC 22989:2023-04, S. 16). Dieser Prozess wird auch als **Training** eines Modells bezeichnet und die hierfür verwendeten Daten als **Trainingsdaten** (DIN EN ISO/IEC 22989:2023-04, S. 17). Nach Definition setzt das maschinelle Lernen voraus, dass ein Modelltyp und damit die bestehenden Anpassungsmöglichkeiten definiert sind, dass eine Metrik – in der Literatur u. a. als Evaluations- oder Verlustfunktion bezeichnet – existiert, nach der optimiert werden kann und dass ein Vorgehen für die Optimierung festgelegt ist (Domingos 2012, S. 79). Bestehende Verfahren des ML unterscheiden sich im Wesentlichen in diesen drei Aspekten (Domingos 2012, S. 79). Für eine umfassende Übersicht über Verfahren des ML sei auf Li (2024) verwiesen. Für Optimierungsverfahren im Zusammenhang mit ML sei im Allgemeinen auf Aggarwal (2020) verwiesen. In der vorliegenden Arbeit wird für die Optimierung im Speziellen Spaltengenerierung (engl. Column Generation, CG) genutzt, ein Prinzip, um lineare Optimierungsprobleme mit einer großen Anzahl von Variablen effizient zu lösen. Hierauf wird in Anhang A1 eingegangen.

Die für das ML verwendeten **Daten** liegen als **Datenpunkte** mit identischem Format vor, die zu einem **Datensatz** zusammengefasst werden (DIN EN ISO/IEC 22989:2023-04, S. 14; Jung 2022, S. 19–20). Datenpunkte stellen atomare Datenelemente dar, die von ML-Algorithmen in großer Anzahl verarbeitet werden (DIN EN ISO/IEC 22989:2023-04, S. 15). Was bei einer gegebenen Problemstellung konkret als Datenpunkt betrachtet wird, muss vom Modellierer in Abhängigkeit des Anwendungsfalls entschieden werden (Jung 2022, S. 19–20). Bei der Erstellung von LLKMs ist ein

Datenpunkt eine Variante. Eigenschaften von Datenpunkten, die aus den Datenpunkten automatisch abgeleitet werden können, werden als **Features** bezeichnet (Jung 2022, S. 204). Für eine Variante, die bereits durch Produktmerkmale beschrieben ist, liegt es z. B. nahe, die Produktmerkmale selbst als Feature des Datenpunkts zu betrachten[22]. Datenpunkte in einem Datensatz sind i. d. R. mit denselben Features beschrieben (Mahalle 2022, S. 17). Jedes Feature hat einen definierten Definitionsbereich. Dieser kann **numerisch** sein – d. h. ein Zahlenbereich wie z. B. für eine Abmessung eines Produkts – oder **kategorisch** – d. h. eine Menge definierter Symbole wie z. B. für die Farbe eines Produkts (Qamar & Raza 2020, S. 64). Bei **binär**en Features handelt es sich um einen speziellen Fall kategorischer Features, die lediglich zwei verschiedene Werte annehmen können (Mahalle 2022, S. 16), wie z. B. die Eigenschaft einer Variante einer bestimmten Norm zu genügen. Neben Features können Datenpunkte weitere Eigenschaften besitzen, die sich nicht automatisch aus dem Datenpunkt selbst ergeben. Diese Eigenschaften werden den Datenpunkten durch Annotation zugeordnet und als **Labels** bezeichnet (Jung 2022, S. 26). Bei der datenbasierten Erstellung von LLKMs kann ein Label für eine Variante z. B. angeben, ob zur Herstellung der Variante ein bestimmter AVO benötigt wird. Labels können ebenso wie Features numerisch oder kategorisch und im Speziellen binär sein (Jung 2022, S. 26). Ob für einen verwendeten Datensatz Labels vorhanden sind oder nicht, ist ein entscheidendes Kriterium für die Einteilung von Problemstellungen des maschinellen Lernens: Wenn Labels vorhanden sind, liegt **überwachtes Lernen** (engl. Supervised Learning, SL) vor, andernfalls **unüberwachtes Lernen** (engl. Unsupervised Learning, UL)[23] (Jung 2022, S. 12–15).

2.3.2 Überwachtes Lernen

SL dient dazu, auf Basis des Datensatzes ein Modell zu erstellen, das den tatsächlich bestehenden Zusammenhang zwischen Features und Labels möglichst genau wiedergibt (Qamar & Raza 2020, S. 45). Sowohl prädiktive Modelle des SL als auch LLKMs, wie sie in der Industrie eingesetzt werden, stellen eine Beziehung zwischen Eingangsdaten in Form von Datenpunkten bzw. Produktmerkmalen und Ausgangsdaten in Form

[22] Der Begriff Feature wird in der vorliegenden Arbeit ausschließlich im Sinne des ML verwendet und nicht als Synonym für Produktfeature im Sinne der Produktentwicklung, auch wenn Features in weiten Teilen der vorliegenden Arbeit Produktmerkmalen entsprechen.
[23] Darüber hinaus existiert mit dem bestärkenden Lernen (engl. Reinforcement Learning) eine weitere Kategorie, die für die vorliegende Arbeit jedoch nicht relevant ist; für eine umfassende Darstellung des bestärkenden Lernens sei auf Ris-Ala (2023) verwiesen.

von Labels bzw. VSTLs und VAPLs her. Aufgrund dieser Analogie ist das SL von Bedeutung für die datenbasierte Erstellung von LLKMs.

Die Probleme des SL können in Klassifikation und Regression unterteilt werden. Eine **Klassifikation** liegt vor, falls die betrachteten Labels kategorisch sind und eine **Regression,** falls sie numerisch sind (Jung 2022, S. 26). Bei einem Klassifikationsproblem werden die Ausprägungen, die das kategorische Label annehmen kann als **Klassen**[24] bezeichnet (Jung 2022, S. 26). Klassifikationsprobleme können weiter unterteilt werden in **binäre Probleme** bei denen es zwei Klassen gibt, d. h. binäre Labels, und **Multiklassen-Probleme** mit mehr als zwei Klassen (Jung 2022, S. 26–27). Darüber hinaus können Probleme des SL vorliegen, bei denen einem Datenpunkt mehrere Labels zugeordnet sind. Diese Probleme werden als **Multi-Label-Probleme** bezeichnet (Jung 2022, S. 27). Abbildung 2.9 veranschaulicht die zentralen Begriffe des SL an einem Beispiel. Im Beispiel liegt mehr als ein Label und für zwei der Labels jeweils mehr als eine Klasse vor. Deshalb handelt es sich dabei um ein Multi-Label- und Multiklassen-Problem. Multi-Label-Probleme sind für die vorliegende Arbeit relevant, da VSTLs und VAPLs i. d. R. durch mehr als eine numerische oder kategorische Eigenschaft beschrieben werden. Ein Multi-Label-Problem mit n Labels kann in n Singlelabel-Probleme zerlegt werden, die jeweils in ihren Features übereinstimmen. Dieses Vorgehen wird als **Binary-Relevance**-Ansatz[25] bezeichnet (Tidake & Sane 2018, S. 1046–1047).

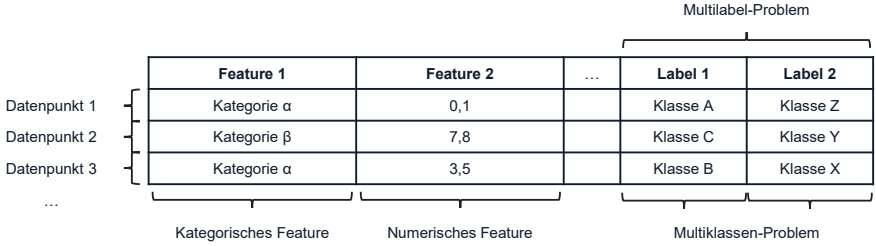


Abbildung 2.9: Beispielhafte Darstellung von Begriffen des überwachten Lernens (Eigene Darstellung auf Basis von Jung 2022, S. 26–27)

[24] Diese sind nicht zu verwechseln mit Klassen in objektorientierten Modellen wie in Kapitel 2.2.2 vorgestellt.
[25] Es existieren weitere Ansätze bei denen z. B. mehrere Labels zu einem Label zusammengefasst werden Tidake & Sane (2018, S. 1046–1049). Die resultierenden Modelle sind jedoch nicht in Regeln überführbar, wie sie in LLKM vorliegen und damit für die vorliegende Arbeit nicht relevant.

Der Trainingsdatensatz, auf Basis dessen das Modell optimiert wird, enthält i. d. R. nur einen Teil aller grundsätzlich möglichen Datenpunkte. Die Herausforderung besteht deshalb darin, ein Modell zu trainieren, das dennoch korrekte Vorhersagen über die Ausprägungen der Labels von allgemeinen Datenpunkten treffen kann. Diese Fähigkeit wird als **Generalisierungsfähigkeit** bezeichnet (Jung 2022, S. 129). Die Generalisierungsfähigkeit eines SL-Modells wird i. d. R. überprüft, indem Vorhersagen des Modells für Datenpunkte außerhalb des Trainingsdatensatzes betrachtet werden. Diese werden mit den tatsächlichen Labelausprägungen für diese Datenpunkte verglichen (DIN EN ISO/IEC 22989:2023-04, S. 32). Dieses Vorgehen wird als **Testen** bezeichnet und die dafür verwendeten Daten als **Testdaten** (DIN EN ISO/IEC 22989:2023-04, S. 32). Von den Testdaten sind die **Validierungsdaten** zu unterscheiden. Diese werden zur Auswahl eines Modelltyps oder zur Einstellung von Hyperparametern eines Modells verwendet (DIN EN ISO/IEC 22989:2023-04, S. 32).

Für eine Diskrepanz zwischen den Modellvorhersagen und den tatsächlichen Ausprägungen der Labels beim Testen kann es drei Ursachen geben: Verzerrung, Varianz und Rauschen (Aggarwal 2021, S. 442–443). **Verzerrung** (engl. Bias) beschreibt den Teil der Abweichung, der darauf zurückzuführen ist, dass das verwendete Modell vereinfachende und unzutreffende Annahmen über bestehende Zusammenhänge impliziert (Aggarwal 2021, S. 442–443). Wenn z. B. ein lineares Regressionsmodell zur Modellierung eines Datensatzes angewandt wird, dem ein nichtlinearer Zusammenhang zwischen Features und Labels zugrunde liegt, bleibt ein Vorhersagefehler bestehen, der unabhängig von der Größe des Trainingsdatensatzes ist. **Varianz** (engl. Variance) bezeichnet den Anteil der Abweichung, der darauf zurückzuführen ist, dass das gelernte Modell nicht statistisch robust bzgl. der verwendeten Datenpunkte ist (Aggarwal 2021, S. 443). Je kleiner der Trainingsdatensatz ist, desto stärker hängt das gelernte Modell davon ab, welche der möglichen Datenpunkte für das Training verwendet werden, d. h. die Varianz kann durch eine Vergrößerung des Trainingsdatensatzes reduziert werden. **Rauschen** (engl. Noise) bezeichnet den Anteil der Abweichung, der auf Fehler in den Daten zurückzuführen ist, wie z. B. Messfehler bei der Datenerhebung (Aggarwal 2021, S. 443). Selbst für ein Modell, das die Gesetzmäßigkeit hinter den betrachteten Daten vollständig korrekt abbildet, verbleibt diese Abweichung, da sie sich unsystematisch in den Trainings- und Testdaten widerspiegelt. Varianz und Verzerrung sind abhängig von der Komplexität des verwendeten Modells (Jung 2022, S. 129) wobei unter **Modellkomplexität** die Anzahl einstellbarer Modellparameter verstanden wird (Spiegelhalter

et al. 2002, S. 584). Modelle mit hoher Komplexität können im Zuge der Optimierung eng an den Trainingsdatensatz angepasst werden (Jung 2022, S. 126–130). Deshalb wird im Allgemeinen davon ausgegangen, dass die Generalisierungsfähigkeit von Modellen bei steigender Modellkomplexität zunächst aufgrund abnehmender Verzerrung zunimmt und anschließend aufgrund zunehmender Varianz abnimmt (Jung 2022, S. 130). Dazwischen liegt im Allgemeinen eine optimale Modellkomplexität (siehe Abbildung 2.10)[26]. Für Modelle in Form von booleschen Ausdrücken, auf die in Kapitel 3.4 näher eingegangen wird, ist eine Beschränkung der Modellkomplexität wichtig, um eine hohe Generalisierungsfähigkeit zu erreichen (Liu et al. 2016, S. 134–143). Verzerrung und Varianz hängen eng mit den Effekten **Unteranpassung** (engl. Underfitting) und **Überanpassung** (engl. Overfitting) zusammen, bei denen eine zu geringe bzw. zu hohe Modellkomplexität zu schlechter Generalisierungsfähigkeit führt (Emmert-Streib & Dehmer 2019, S. 533–534). Unteranpassung geht mit hoher Verzerrung und geringer Varianz einher, Überanpassung mit geringer Verzerrung und hoher Varianz (Rocks & Mehta 2022, S. 1).

Insbesondere in Anwendungsfällen in denen die Vorhersagen eines Modells weitreichende Konsequenzen haben können, besteht neben der Anforderung einer hohen Generalisierungsfähigkeit häufig auch die Anforderung der **Interpretierbarkeit**. Hierunter versteht man, dass der Prozess zum Schlussfolgern einer Vorhersage für Menschen verständlich ist (Rudin et al. 2022, S. 11). Dies trifft auf viele prädiktive Modelle des ML, wie z. B. gängige Tiefe Neuronale Netze (engl. Deep Neural Networks, DNN), nicht zu

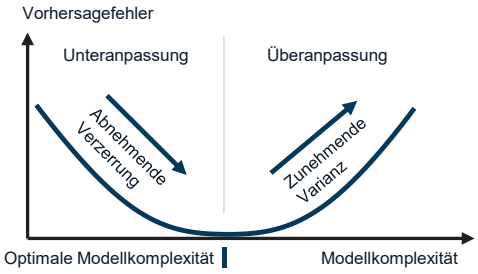


Abbildung 2.10: Schema der optimalen Modellkomplexität in Anlehnung an Jung (2022, S. 129)

[26] Tiefe neuronale Netze (engl. Deep Neural Networks) können Ausnahmen von diesem Verhalten zeigen (Jung (2022, S. 130)), werden jedoch in der vorliegenden Arbeit nicht betrachtet.

(Rudin 2019, S. 206). In der Industrie eingesetzte LLKMs müssen im Allgemeinen interpretierbar sein, da sie manuell gepflegt werden. Methoden zum ML interpretierbarer Modelle sind deshalb für die vorliegende Arbeit besonders relevant. Interpretierbarkeit unterscheidet sich von **Erklärbarkeit**. Unter Erklärbarkeit wird verstanden, dass die Vorhersagen eines nichtinterpretierbaren Modells post-hoc durch ein anderes Modell erklärt werden können (Rudin 2019, S. 206). Z. B. werden Vorhersagen von DNNs zur Bilderkennung durch die Relevanz von Pixeln für eine bestimmte Vorhersage erklärt, wobei jedoch die Modelle selbst nicht interpretierbar sind (Rudin 2019, S. 208). Aufgrund der Interpretierbarkeit von LLKMs die in der Industrie genutzt werden, ist Erklärbarkeit für diese Modelle nicht relevant.

Wie in Kapitel 1.2 erläutert, werden in der vorliegenden Arbeit Anwendungsfälle berücksichtigt, bei denen für eine datenbasierte Erstellung von LLKMs nicht genügend annotierte Datenpunkte – d. h. Varianten mit zugehörigen VSTLs und VAPLs – zur Verfügung stehen. In diesen Fällen soll der bestehende Datensatz erweitert werden. Dieses Vorgehen entspricht im Kontext des ML dem **aktiven Lernen** (engl. Active Learning, AL). Beim AL liegen initial einige annotierte Datenpunkte vor und die Anfrage von Labels für weitere Datenpunkte geht mit Kosten einher. Abbildung 2.11 stellt das Konzept des AL schematisch dar. Auf Basis der annotierten Datenpunkte wird ein Modell trainiert. Anschließend werden bei einem sog. **Orakel**, wie z. B. einem Domänenexperten, ein oder mehrere Labels für einen oder mehrere weitere Datenpunkte angefragt. Nach Erhalt der Labels wird das Modell erneut trainiert, wobei aufgrund abnehmender Varianz eine höhere Generalisierungsfähigkeit zu erwarten ist. Dieser Vorgang wird so lange wiederholt bis ein Budget aufgebraucht, eine festgelegte Generalisierungsfähigkeit erzielt oder ein anderes Abbruchkriterium erreicht ist. (Tharwat & Schenck 2023, S. 820–840)

Die Herausforderung des AL besteht darin, mit der Anfrage weniger Labels eine möglichst große Steigerung der Generalisierungsfähigkeit zu erreichen. Hierfür existieren verschiedene Strategien. **Informationsbasierte Anfragestrategien** ermitteln den Informationsgewinn eines Labels auf Basis bereits trainierter Modelle. Es werden z. B. Labels für Datenpunkte angefragt, für die eine hohe Vorhersageunsicherheit besteht oder die einen großen erwarteten Einfluss auf das zuletzt trainierte Modell oder dessen Vorhersagen erwarten lassen. Alternativ hierzu werden beim Query by Committee (QBC) mehrere Modelle trainiert und anschließend Datenpunkte ausgewählt, für die die Vorhersagen des Komitees heterogen sind. Bei **repräsentationsbasierte**n

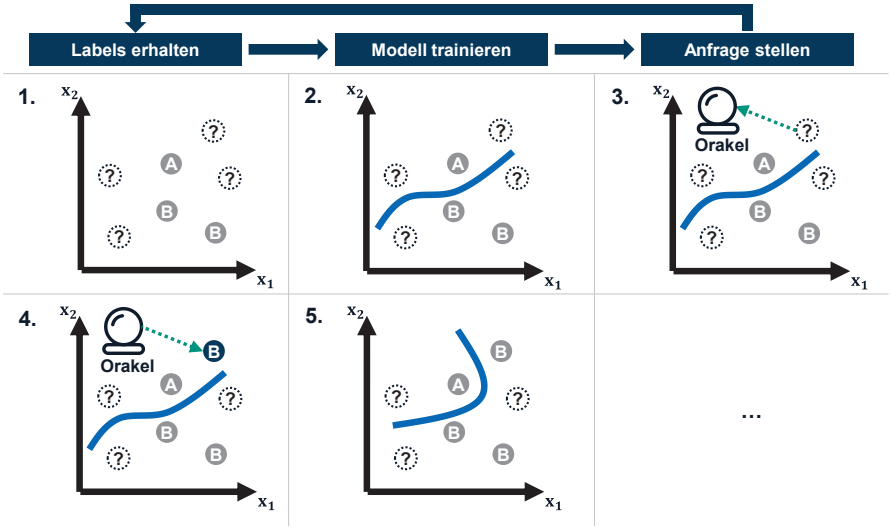


Abbildung 2.11: Schema des aktiven Lernens (Eigene Darstellung auf Basis von Tharwat und Schenck 2023, S. 820–840)

Anfragestrategien ist die Anfrage hingegen von der Verteilung der Datenpunkte im Feature-raum abhängig. Der **Featureraum** ist derjenige – i. d. R. mehrdimensionale – Raum, der durch die Wertebereiche der Features aufgespannt wird. Es werden entweder Datenpunkte gewählt, die weit entfernt von anderen Datenpunkten, in Bereichen mit hoher Dichte oder im Zentrum von Clustern liegen. (Tharwat & Schenck 2023, S. 828–836)

Grundsätzlich können drei verschiedene Anwendungsszenarien des AL unterschieden werden. Beim **poolbasierten AL** liegt die Gesamtheit der möglichen Datenpunkte als endliche diskrete Menge vor. Für eine Labelanfrage wird aus diesem Pool ein nichtannotierter Datenpunkt ausgewählt. Beim **streambasierten AL** werden die möglichen Datenpunkte nacheinander betrachtet und es wird jeweils entschieden, ob für den betrachteten Datenpunkt ein Label angefragt werden soll. Bei der **Membership Query Synthesis** (MQS) werden Datenpunkte nicht aus einer endlichen Menge von Datenpunkten ausgewählt, sondern unter Berücksichtigung bestimmter Bedingungen generiert. Poolbasiertes AL ist das in der Literatur überwiegend betrachtete Szenario. MQS wird für viele Anwendungsfälle als ungeeignet angesehen, weil nicht sichergestellt werden

kann, dass die generierten Datenpunkte vom Orakel sinnvoll annotiert werden können. (Tharwat & Schenck 2023, S. 826–827) Wie in Kapitel 4.5 ausgeführt wird, ist MQS für den Anwendungsfall der vorliegenden Arbeit jedoch geeignet und vorteilhaft.

2.3.3 Unüberwachtes Lernen

UL dient dazu, Muster zu erkennen, die jeweils die Datenpunkte untereinander oder die Features untereinander in Beziehung setzen (Aggarwal 2021, S. 299). UL wird zum Clustern von Datenpunkten, zum Aggregieren von Features und zur Reduktion der Dimension des Datensatzes eingesetzt (Aggarwal 2021, S. 299). Darüber hinaus wird UL zum Erkennen von **Ausreißern** in Datensätzen verwendet. Nach einer vielzitierten Definition von Hawkins (1980, S. 1) handelt es sich bei Ausreißern um Beobachtungen, die so sehr von anderen Beobachtungen abweichen, dass der Verdacht naheliegt, dass sie durch einen anderen Mechanismus generiert wurden. Der Begriff **Anomalie** wird in der vorliegenden Arbeit, wie auch in der Literatur (Aggarwal 2017, S. 1), synonym hierzu verwendet. In der Literatur finden sich zahlreiche Arbeiten die Anomalieerkennung nutzen um Hinweise auf Fehler zu erhalten. Dabei kann es sich um technische, aber auch menschenverursachte Fehler handeln, wie z. B. um Fehler in der industriellen Montage (Rijayanti et al. 2023), bei der Steuerung von Flugzeugen (Igenewari et al. 2019) oder bei der Erstellung von medizinischen Behandlungsplänen (Sipes et al. 2014). Der Einsatz von Anomalieerkennung für das Ermitteln von Fehlern in LLKMs erscheint deshalb aussichtsreich.

Existierende Ansätze zur Anomalieerkennung auf Datensätzen ohne Labels lassen sich in vier Kategorien einteilen: dichte-, distanz-, wahrscheinlichkeits- und abhängigkeitsbasierte Ansätze (Li & van Leeuwen 2023, S. 2518–2519). Bei **dichte- und distanzbasierten Ansätze**n werden Datenpunkte als Anomalien identifiziert, wenn sie sich in einem Bereich des Featureraums mit geringer Dichte befinden bzw. einen großen Abstand zu anderen Datenpunkten aufweisen (Li & van Leeuwen 2023, S. 2518). In **wahrscheinlichkeitsbasierte**n **Ansätze**n wird die plausibelste Wahrscheinlichkeitsverteilung für das Zustandekommen der Datenpunkte ermittelt. Es werden diejenigen Datenpunkte als Anomalien identifiziert, die nach dieser Verteilung eine geringe Auftretenswahrscheinlichkeit aufweisen (Li & van Leeuwen 2023, S. 2518).

Abhängigkeitsbasierte Ansätze nutzen Methoden des SL um Abhängigkeiten[27] zwischen Features zu ermitteln und damit Muster im Datensatz zu erkennen. Datenpunkte werden als Anomalien identifiziert, wenn sie dem ermittelten Muster nicht entsprechen (Li & van Leeuwen 2023, S. 2519–2520). Wie in Kapitel 4.6 ausgeführt wird, sind für die vorliegende Arbeit abhängigkeitsbasierte Ansätze relevant.

[27] Bei Abhängigkeiten im Sinne abhängigkeitsbasierter Ansätze des UL handelt es sich um Muster in Daten. Sie sind nicht zu verwechseln mit Abhängigkeiten die in einem KM definiert sind und in Kapitel 2.2.1 erläutert werden.

3 Stand der Forschung

Im Folgenden werden für jedes der in Kapitel 1.2 definierten Probleme 1 bis 6 die Anforderungen an die entsprechenden Methoden 1 bis 6 zur Lösung dieses Problems präzisiert. Dies entspricht dem zweiten Schritt des Design Science Research Process (DSRP, siehe Kapitel 1.3). Die folgenden Unterkapitel entsprechen den Problemen 1 bis 6. Je Problem werden zunächst auf Basis des in Kapitel 1.1 beschriebenen Kontextes und der Motivation der vorliegenden Arbeit Anforderungen (A) an die zugehörige Methode abgeleitet. Anschließend wird jeweils untersucht, inwieweit Methoden nach Stand der Forschung diese Anforderungen erfüllen. Schließlich wird je Problem das bestehende Lösungsdefizit ermittelt, das durch die Entwicklung eigener Methoden behoben werden soll.

3.1 Problem 1: Datenbasierte Erstellung von Konfigurationsmodellen

3.1.1 Anforderungen

Wie in Kapitel 1.1 beschrieben, liegt das Potenzial von Konfigurationssystemen (KSs) für die Arbeitsablaufplanung in der automatischen Erstellung von variantenbezogenen Stücklisten (VSTLs) und variantenbezogenen Arbeitsplänen (VAPLs). Entsprechend der Definition einer Konfiguration in Kapitel 2.2.1 beinhaltet dies jeweils die Festlegung der Elemente und ihrer Parameterausprägungen sowie der Struktur der VSTL bzw. des VAPL. Die datenbasiert erstellten Low-Level-Konfigurationsmodelle (LLKMs) müssen daher in der Lage sein, die Komponenten der VSTL auszuwählen **(Anforderung A1a: Auswahl von Komponenten)**, die Ausprägungen derer Parameter zu definieren **(Anforderung A1b: Ausprägung der Komponentenparameter)** und die Struktur der VSTL festzulegen **(Anforderung A1c: Festlegung der Stücklistenstruktur)**. Hinsichtlich des Arbeitsplans (APL) müssen sie in der Lage sein, die Arbeitsvorgänge (AVOs) des VAPL auszuwählen **(Anforderung A1d: Auswahl von Arbeitsvorgängen)**, die Ausprägungen derer Parameter zu definieren **(Anforderung A1e: Ausprägung der Arbeitsvorgangsparameter)** und die Struktur des VAPL festzulegen **(Anforderung A1f: Festlegung der Arbeitsplanstruktur)**. Damit diese Modelle nach ihrer Erstellung von Experten gepflegt werden können, müssen sie – ebenso wie LLKMs, die derzeit in der Industrie verwendet werden – interpretierbar sein **(Anforderung A1g: Interpretierbares Modell)**. Zuletzt werden auch Fälle berücksichtigt, in denen die Anzahl der

verfügbaren Datenpunkte zu gering ist, um ein ausreichend genaues LLKM datenbasiert zu erstellen (siehe Kapitel 1.1). Dazu muss es möglich sein, den Datensatz möglichst effizient zu erweitern **(Anforderung A1h: Erweiterung der Datenbasis)**.

3.1.2 Relevante Arbeiten

Für die manuelle Erstellung von Konfigurationsmodellen (KMs) existieren Vorgehensmodelle wie in Kapitel 2.2.3.1 beschrieben und von Shafiee et al. (2017) zusammenfassend dargestellt. Diese sind grundsätzlich geeignet, um vollständige Modelle für die Produkt- und Prozesskonfiguration zu erstellen. Allerdings sind diese Vorgehensmodelle nicht datenbasiert und weisen deshalb die in Kapitel 1.1 beschriebenen Nachteile auf. Im Folgenden werden Ansätze zur datenbasierten Erstellung von KMs sowie weiterer Modelle betrachtet, die eine Konfiguration von VSTLs oder VAPLs ermöglichen. Entsprechend der herausragenden Bedeutung der Produktkonfiguration in der Literatur beschränken sich bestehende Ansätze zur datenbasierten Erstellung auf die Produktkonfiguration. Im Kontext der Arbeitsablaufplanung existieren jedoch darüber hinaus Ansätze, die der Erstellung von Modellen dienen, mit denen VAPLs in einem weiten Sinne generiert werden können. Auch wenn diese Modelle keine KMs im Sinne der vorliegenden Arbeit sind, werde diese Arbeiten aufgrund ihrer ähnlichen Funktion im Folgenden ebenfalls betrachtet.

Datenbasierte Erstellung von Konfigurationsmodellen

Es existieren drei verwandte Ansätze von Wang et al. (2023), He et al. (2021) und Shao et al. (2006), die sich mit der datenbasierten Erstellung von Produktkonfigurationsmodellen befassen[28]. Ansätze, die sich explizit mit der datenbasierten Erstellung von Prozesskonfigurationsmodellen befassen existieren hingegen nicht. Die Ansätze von Wang et al. (2023) und Shao et al. (2006) gehen davon aus, dass bereits eine generische Stückliste definiert ist. Für jede generische Klasse existieren verschiedene mögliche Instanzen, die in Abhängigkeit der Produktmerkmale oder der Parameter übergeordneter Klassen ausgewählt werden. Der Ansatz von He et al. (2021) geht ebenfalls von generischen Klassen aus, berücksichtigt jedoch keine meronymischen Beziehungen zwischen den Klassen. Wang et al. (2023) ordnen den generischen Klassen

[28] Darüber hinaus existiert eine Arbeit von Chen & Wang (2009), die sich mit der datenbasierten Erstellung eines Modells befasst, dass als Teil eines Produktkonfigurationssystems beschrieben wird. Dieses dient jedoch der Übertragung von Produktmerkmalen auf Designparameter des Produkts und nicht der Erstellung von Stücklisten. Deshalb wird es an dieser Stelle nicht näher betrachtet.

Parameter zu. Diese hängen allerdings nicht unmittelbar von den Produktmerkmalen ab, sondern werden durch die Auswahl einer Instanz ausgeprägt. Dies beschränkt die Mächtigkeit des KM. Die beiden anderen Ansätze berücksichtigen keine Parameter der Komponenten. Der Fokus der drei Arbeiten ist die datenbasierte Erstellung der Abhängigkeiten über die die Instanzen der generischen Klassen ausgewählt werden. Dafür werden automatische Featureselection und heuristische Entscheidungsbaumklassifikation (Wang et al. 2023) bzw. Association Rule Mining (He et al. 2021; Shao et al. 2006) eingesetzt. Die datenbasierte Erstellung der verwendeten generischen Klassen oder deren Beziehungen ist nicht Gegenstand der existierenden Arbeiten.

Datenbasierte Erstellung prädiktiver Modelle für die Arbeitsablaufplanung

Die existierenden Ansätze zur datenbasierten Erstellung von prädiktiven Modellen stellen Alternativen zu klassischen Ansätzen des Computer Aided Process Plannings (CAPP) dar. Bei den klassischen Ansätzen werden typischerweise APLs auf Basis von Regeln, die durch Experten definiert werden, erstellt (Hussong et al. 2021, S. 648; Schenk 2014, S. 754). Die entsprechenden Arbeiten von Hussong et al. (2021), Natarajan & Gokulachandran (2020), Schuh et al. (2019), Schuh et al. (2017) und Amaitik (2012) gehen, wie das klassische CAPP, von Bauteilen aus, die durch Fertigungsfeature[29] beschrieben sind. Abweichend davon verwendet der Ansatz von Hashimoto & Nakamoto (2021) eine Voxel-Darstellung eines CAD-Modells als Ausgangsbasis und der Ansatz von Joo et al. (2001) Metadaten, die das CAD-Modell beschreiben. Die Ansätze nutzen historische Daten, um tiefe neuronale Netze (DNNs) – oder im Fall von Schuh et al. (2019) und Schuh et al. (2017) Entscheidungsbäume – zu trainieren, um AVOs und z. T. auch zugehörige Maschinen, Werkzeuge und Prozessparameterausprägungen vorherzusagen. Hussong et al. (2021), Schuh et al. (2019) und Schuh et al. (2017) schlagen darüber hinaus vor, Methoden des maschinellen Lernens (ML) auch für die Bestimmung der Reihenfolge von AVOs einzusetzen, wobei Hussong et al. (2021) hierfür konkret auf DNNs mit einer Long Short Term Memory-Architektur[30] verweist. Die entsprechenden Vorschläge werden jedoch nicht weiter ausgeführt und es bleibt offen, wie mit einem solchen Ansatz allgemeine APLs – d. h. auch solche mit nichtlinearer Struktur – berücksichtigt werden können.

[29] Fertigungsfeature entsprechen geometrischen Elementen mit zugehöriger Semantik (Hussong et al. 2021, S. 649).
[30] Für eine Erläuterung eines solchen Ansatzes sei auf Aggarwal (2021, S. 290–294) verwiesen.

3.1.3 Lösungsdefizit

Tabelle 3.1: Stand der Forschung zur datenbasierten Erstellung von Konfigurationsmodellen

● = Vollständig erfüllt ◐ = Teilweise erfüllt ○ = Nicht erfüllt	A1a Auswahl von Komponenten	A1b Ausprägung der Komponentenparameter	A1c Festlegung der Stücklistenstruktur	A1d Auswahl von Arbeitsvorgängen	A1e Ausprägung der Arbeitsvorgangsparameter	A1f Festlegung der Arbeitsplanstruktur	A1g Interpretierbares Modell	A1h Erweiterung der Datenbasis
Datenbasierte Erstellung von Konfigurationsmodellen								
Wang et al. 2023	●	◐	◐	○	○	○	●	○
He et al. 2021	●	○	○	○	○	○	●	○
Shao et al. 2006	●	○	◐	○	○	○	●	○
Datenbasierte Erstellung prädiktiver Modelle für die Arbeitsablaufplanung								
Hashimoto & Nakamoto 2021	○	○	○	●	●	○	○	○
Hussong et al. 2021	○	○	○	●	○	◐	○	○
Natarajan & Gokulachandran 2020	○	○	○	●	○	○	○	○
Schuh et al. 2019, Schuh et al. 2017	○	○	○	●	●	◐	●	○
Amaitik 2012	○	○	○	●	●	○	○	○
Joo et al. 2001	○	○	○	●	●	○	○	○

Tabelle 3.1 gibt einen Überblick über Ansätze zur datenbasierten Erstellung von KMs nach Stand der Forschung sowie deren Anforderungserfüllung. Derzeit existieren nur wenige Ansätze, die explizit für die datenbasierte Erstellung von KMs entwickelt wurden. Diese beschränken sich auf die Produktkonfiguration und vernachlässigen die datenbasierte Erstellung von Strukturen der MSTL und des MAPL. Im thematisch angrenzenden Umfeld des CAPP existieren Ansätze zum Lernen prädiktiver Modelle für die Inferenz von APLs. Die gelernten prädiktiven Modelle sind typische Modelle des ML und nicht mit industrieüblichen KMs vergleichbar, die aus Maximalstücklisten (MSTLs), Maximalarbeitsplänen (MAPLs) und Regeln bestehen. Darüber hinaus bleibt die Struktur von APLs weitgehend unberücksichtigt. Zusammenfassend lässt sich festhalten, dass die datenbasierte Erstellung industrieller KMs nur rudimentär erforscht ist. Insbesondere besteht kein integrierter Ansatz zur datenbasierten Erstellung von LLKMs. Die zu entwickelnde Methode 1 soll in der Lage sein, LLKMs datenbasiert zu erstellen und dabei die Anforderungen A1a bis A1h berücksichtigen. Auf die Anforderungen A1b und A1e wird in der vorliegenden Arbeit dabei nur am Rande eingegangen, weil der Fokus auf der Auswahl von Elementen liegt (siehe Kapitel 2.2.2.4). In den folgenden Kapiteln

wird untersucht, inwieweit in der Literatur Ansätze zur Adressierung der Teilprobleme von Problem 1 existieren.

3.2 Problem 2: Datenbasierte Erstellung von Maximalstücklisten

3.2.1 Anforderungen

Bei der datenbasierten Erstellung von MSTLs sollen keine Informationen die in den Daten – d. h. den verwendeten VSTLs – vorhanden sind verloren gehen. Damit muss es möglich sein, jede zur Erstellung verwendete VSTL aus der resultierenden MSTL zu konfigurieren **(Anforderung A2a: Informationserhaltung)**. Die Menge an verfügbaren Daten kann von Fall zu Fall unterschiedlich sein und auch Fälle mit hoher Datenverfügbarkeit sollen berücksichtigt werden können **(Anforderung A2b: Skalierbarkeit)**. Wie in Kapitel 2.2.2.2 erläutert, können Strukturalternativen (STAs) in VSTLs auftreten. Diese können technisch begründet sein oder Inkonsistenzen im Datensatz darstellen. Sind sie technisch begründet, müssen die entsprechenden alternativen Strukturen als Optionen im KM abgebildet werden. Wenn sie Inkonsistenzen im Datensatz darstellen, ist es wichtig, dass sie in einer datenbasiert erstellten MSTL sichtbar sind, damit sie von einem Domänenexperten als solche erkannt werden können. Eine Methode zur datenbasierten Erstellung von MSTLs muss daher in der Lage sein, STAs in den eingehenden VSTLs zu erkennen und in der MSTL darzustellen **(Anforderung A2c: Strukturalternativen)**. Zuletzt soll vermieden werden, dass die Methoden Aspekte industrieüblicher Stücklisten nicht berücksichtigen und damit nicht praktisch anwendbar sind. VSTLs mit mehreren identisch bezeichneten Komponenten an verschiedenen Positionen der VSTL, im Folgenden Multikomponenten genannt, sind im Sinne der Gleichteilverwendung in der Industrie üblich. Liegen Multikomponenten in VSTLs vor und werden nicht explizit berücksichtigt, können sie fälschlicherweise als STA erkannt werden. Wenn z. B. eine Komponente an zwei verschiedenen Positionen in der Stückliste auftreten kann und zwei VSTLs jeweils eine der beiden Positionen enthalten existieren vermeintlich zwei mögliche Strukturen für die MSTL. Deshalb müssen Multikomponenten in den VSTLs erkannt und bei der Erstellung der MSTL berücksichtigt werden **(Anforderung A2d: Multikomponenten)**. Es muss außerdem berücksichtigt werden, dass Baugruppen in der Praxis beliebig viele und insbesondere mehr als zwei Subkomponenten aufweisen können **(Anforderungen A2e: Vielelementigkeit)**. Dadurch sind bestimmte in der Literatur beschriebene Ansätze prinzipiell ausgeschlossen.

3.2.2 Relevante Arbeiten

Im Folgenden werden entsprechend der Problemstellung Ansätze zur Erstellung von MSTLs auf Basis von VSTLs betrachtet. Für die vorliegende Arbeit ist die Kernidee der unten vorgestellten Arbeit von Moussa & ElMaraghy (2018) von herausragender Bedeutung. Die Arbeit nutzt einen Ansatz um phylogenetische Bäume, welche evolutionäre Beziehungen zwischen Lebewesen darstellen, zusammenzuführen. Dabei handelt es sich um eine Problemstellung aus dem Fachgebiet der Phylogenetik, die mit dem Zusammenführen von VSTLs in einer MSTL vergleichbar ist. Da diese Problemstellung in der Phylogenetik bereits umfassend erforscht ist, besteht großes Potenzial in einem Transfer bestehender Ansätze. Deshalb werden im Folgenden auch zwei ausgewählte Ansätze der Phylogenetik näher betrachtet, welche für die vorliegende Arbeit relevant sind. Anhang A2.1 geht auf die relevanten Begriffe der Phylogenetik ein und zeigt die Grenzen der Übertragbarkeit bestehender Ansätze der Phylogenetik.

Datenbasierte Erstellung von Maximalstücklisten und vergleichbarer Strukturen

Moussa & ElMaraghy (2019) und Kashkoush & ElMaraghy (2014) entwickeln aufeinander aufbauende Ansätze zur Ableitung sog. Master Assembly Networks aus sog. Assembly Sequence Trees. Assembly Sequence Trees und Master Assembly Networks sind Graphen, deren Knoten Zukaufkomponenten (ZKs) darstellen, die im Rahmen der Montage zu einer Baugruppe gefügt werden. Sie entsprechen in ihrer Funktion und ihrem Aufbau VSTL bzw. MSTL, weshalb sie im vorliegenden Kapitel betrachtet werden. Die Autoren codieren Master Assembly Networks als Binärmatrizen. Diese Codierung setzt voraus, dass das Master Assembly Network ein Binärbaum ist. Mittels Metaheuristiken werden Lösungen in Form von Binärmatrizen erstellt. Hierbei wird die Robinson-Foulds-Distanz zwischen dem zugehörigen Master Assembly Network und den eingehenden Assembly Sequence Trees als zu minimierende Fitnessfunktion verwendet. Bei dieser Distanz handelt es sich um eine in der Phylogenetik gebräuchliche Metrik für die Unähnlichkeit zweier Bäume. Das Verfahren stellt nicht sicher, dass alle eingehenden Assembly Sequence Trees aus dem Master Assembly Network konfiguriert werden können. Insbesondere können Fälle auftreten in denen die Fitnessfunktion optimal wird, wenn bestimmte Assembly Sequence Trees im Master Assembly Network nicht berücksichtigt werden. Dies ist vor allem dann der Fall, wenn deren Struktur von der Mehrheit der Assembly Sequence Trees abweicht. Multikomponenten werden nicht betrachtet.

Moussa & ElMaraghy (2018) entwickeln ebenfalls einen Ansatz zur Ableitung von Master Assembly Networks aus Assembly Sequence Trees. Die Autoren nutzen einen Algorithmus aus der Phylogenetik um sog. Galled Networks zu erstellen. Dabei handelt es sich um Graphen mit beschränkter Abweichung von einer Baumstruktur. Wie Anhang A2.1 ausführt, stellt der Ansatz nicht sicher, dass alle eingehenden Assembly Sequence Trees aus dem resultierenden Master Assembly Network konfiguriert werden können. Darüber hinaus kann der Ansatz keine Multikomponenten berücksichtigen, da hierfür keine Analogie in der Phylogenetik existiert.

Kashkoush & ElMaraghy (2015) und Kashkoush & ElMaraghy (2016) entwickeln einen Ansatz um Produktfamilien bzw. Master Assembly Sequences in Baumstruktur aus VSTLs bzw. Assembly Sequence Trees abzuleiten. Hierfür codieren sie sowohl die eingehenden Bäume als auch den resultierenden Baum jeweils als Matrix. Sie nutzen mathematische Optimierung, um eine möglichst hohe Ähnlichkeit der Matrix des resultierenden Baums mit den Matrizen der eingehenden Bäume zu erzielen. Das von ihnen aufgestellte Optimierungsproblem geht jedoch davon aus, dass alle Bäume Binärbäume sind und keine Multikomponenten vorliegen. STAs werden nicht berücksichtigt. Treten in den eingehenden Bäumen STAs auf, bildet der resultierende Baum genau eine davon ab, sodass Bäume mit anderen Strukturen nicht abgeleitet werden können. Multikomponenten werden nicht berücksichtigt.

Romanowski & Nagi (2004) entwickeln eine Methode, um aus VSTLs eine generische STL zu erstellen. Ein wiederkehrendes Prinzip der entwickelten Methode ist das Clustering auf Basis von Ähnlichkeitsmaßen. Zunächst werden ZKs durch Clustering generalisiert wobei ein aggregiertes Ähnlichkeitsmaß verwendet wird, das u. a. syntaktische und semantische Ähnlichkeit der Bezeichnungen berücksichtigt. Als nächstes werden die VSTLs auf Basis der Ähnlichkeit ihrer Baugruppen geclustert. Dabei ergibt sich die Ähnlichkeit der Baugruppen jeweils aus der Ähnlichkeit der untergeordneten Komponenten und somit letztlich aus der Ähnlichkeit der enthaltenen ZKs. Zuletzt werden je Cluster die Baugruppen aller enthaltenen VSTLs geclustert, so dass jeweils eine generische MSTL mit einer Baugruppenklasse je Baugruppencluster entsteht. Der Ansatz von Romanowski & Nagi (2004) berücksichtigt explizit STAs, allerdings bestehen bei deren Ermittlung folgende Defizite. Es ist a priori nicht bekannt, wie viele STAs in den VSTLs vorliegen und damit wie viele Cluster von VSTLs zu bilden sind. Heuristische Vorgehen wie insbesondere das eingesetzte Silhouette-Verfahren können nicht garantieren, dass die ermittelte Anzahl von STAs tatsächlich vorliegt. Darüber hinaus lässt

das verwendete Distanzmaß für VSTLs nicht zwingend auf STAs schließen. VSTLs, die nur in wenigen Baugruppen übereinstimmen, können sehr unterschiedliche ZKs enthalten und müssen sich nicht zwingend in ihrer Struktur widersprechen. Außerdem gehen innerhalb eines Clusters von VSTLs Informationen über STAs verloren, wenn Baugruppen auf Basis ihrer Ähnlichkeit zusammengefasst werden. Zuletzt kann nicht ausgeschlossen werden, dass durch die Zusammenfassung von ZKs im Rahmen der Generalisierung STAs entstehen, die tatsächlich in den VSTLs nicht vorliegen. Dieser Fall kann auftreten, wenn die ZKs eigentlich verschiedenen Positionen der MSTL entsprechen. Multikomponenten können im beschriebenen Ansatz nicht berücksichtigt werden, da diese zwangsläufig zusammengefasst würden und damit zu STAs führen würden, die in den VSTLs nicht vorliegen.

Algorithmen aus der Phylogenetik

Wie zuvor erläutert, lässt sich eine Analogie zwischen phylogenetischen Bäumen und STLs herstellen, die auch von Moussa & ElMaraghy (2018) genutzt wird. Damit ist es naheliegend, Ansätze zur Synthese phylogenetischer Bäume bzw. Netzwerke auf die Anwendbarkeit für die Erstellung von MSTLs zu untersuchen. Die Literatur zur Synthese phylogenetischer Bäume bzw. Netzwerke ist umfangreich, weshalb im Folgenden nur zwei ausgewählte, besonders relevante Arbeiten vorgestellt werden.

Deng & Fernández-Baca (2018) stellen einen Algorithmus vor, der die Synthese eines Baums ermöglicht, der eine Menge gegebener Bäume darstellen kann. Existiert kein solcher Baum gibt der Algorithmus eine entsprechende Ausgabe zurück. Damit kann zwar erkannt werden, dass STAs vorliegen, jedoch nicht welche. Außerdem lassen sich aus dem synthetisierten Baum die eingehenden Bäume nicht in dem für die vorliegende Arbeit relevanten Sinne ableiten (siehe Problematik der Darstellbarkeit in Anhang A2.1).

Nach Huson & Linz (2018) gab es zum Zeitpunkt der Veröffentlichung keine Algorithmen, die in der Lage waren, aus Bäumen mit beliebiger Topologie, Netzwerke zu synthetisieren, die alle Bäume darstellen können. Sie präsentieren einen solchen Algorithmus, der jedoch nur für genau zwei eingehende Bäume angewandt werden kann. Es existieren neuere Arbeiten, die jedoch erhebliche Einschränkungen bzgl. der Topologie

der eingehenden Bäume aufweisen, weshalb sie für die vorliegende Arbeit nicht relevant sind [31].

3.2.3 Lösungsdefizit

Tabelle 3.2: Stand der Forschung zur datenbasierten Erstellung von MSTLs

	A2a	A2b	A2c	A2d	A2e
● = Vollständig erfüllt ◐ = Teilweise erfüllt ○ = Nicht erfüllt	**Informationserhaltung**	**Skalierbarkeit**	**Strukturalternativen**	**Multikomponenten**	**Vielelementigkeit**
Datenbasierte Erstellung von Maximalstücklisten und vergleichbarer Strukturen					
Moussa & ElMaraghy 2019; Kashkoush & ElMaraghy 2014	○	●	●	○	○
Moussa & ElMaraghy 2018	○	●	●	○	●
Kashkoush & ElMaraghy 2016, 2015	○	●	○	○	○
Romanowski & Nagi 2004	○	●	◐	○	●
Algorithmen aus der Phylogenetik					
Deng & Fernández-Baca 2018	○	●	◐	○	●
Huson & Linz 2018	○	○	●	○	●

Tabelle 3.2 gibt einen Überblick über die Ansätze zur datenbasierten Erstellung von MSTLs nach Stand der Forschung sowie deren Anforderungserfüllung. Es lässt sich festhalten, dass die Anforderungen A2a und A2d trotz ihrer praktischen Relevanz bisher nicht berücksichtigt werden. Die zu entwickelnde Methode 2 soll diese Lücke schließen. Sie soll in der Lage sein, MSTLs auf Basis von VSTLs zu erstellen und dabei alle Anforderungen A2a bis A2e erfüllen.

3.3 Problem 3: Datenbasierte Erstellung von Maximalarbeitsplänen

3.3.1 Anforderungen

Analog zu den Anforderungen an Methoden zur datenbasierten Erstellung von MSTLs (Problem 2) gelten für die datenbasierte Erstellung von MAPLs (Problem 3) die folgenden Anforderungen: Alle eingegangenen VAPLs müssen sich aus dem MAPL konfigurieren lassen **(Anforderung A3a: Informationserhaltung)**, auch große Anzahlen von VAPLs müssen verarbeitet werden können **(Anforderung A3b: Skalierbarkeit)** und

[31] Beispielhaft sei hier auf den Ansatz von Schaller et al. (2021) verwiesen der voraussetzt, dass die Bäume alle auf derselben Menge an Taxa definiert sind.

STAs müssen erkannt und dargestellt werden können **(Anforderung A3c: Strukturalternativen)**. Analog zu Multikomponenten in VSTLs können in VAPLs Multivorgänge vorliegen, d. h. mehrfach auftretende identisch bezeichnete AVOs, was berücksichtigt werden muss **(Anforderung A3d: Multivorgänge).** Zuletzt sollen auch bezüglich der verarbeitbaren VAPLs Einschränkungen in der Struktur vermieden werden. Es sollen deshalb nicht nur lineare VAPLs, sondern auch solche mit Parallelitäten berücksichtigt werden können **(Anforderung A3e: Parallelitäten)**.

3.3.2 Relevante Arbeiten

Im Folgenden werden entsprechend der Problemstellung Ansätze zur Erstellung von MAPLs auf Basis von VAPLs betrachtet. Darüber hinaus werden Arbeiten betrachtet, die sich mit verwandten Problemstellungen befassen. Hierzu zählen die datenbasierte Erstellung von Vorranggraphen sowie das Process Discovery, d. h. die Ermittlung von Geschäftsprozessmodellen aus Ereignisdaten.

Datenbasierte Erstellung von Maximalarbeitsplänen

Es liegen ähnliche Ansätze von Zhang (2012), Zhang & Rodrigues (2009), Zhang et al. (2008) sowie Jiao et al. (2007) zur Erstellung von Strukturen vor, die als Generic Processes, Process Platforms oder Generic Routings bezeichnet werden. In Ihrer Funktion entsprechen diese Strukturen einem MAPL. Allerdings gehen die Autoren von APLs der Montage aus und nehmen an, dass diese aufgrund der konvergenten Struktur der Montage als Bäume dargestellt werden können. Jedem AVO sind Komponenten der STL und ggf. Eigenschaften wie Ressource oder Bearbeitungszeit zugeordnet. Jiao et al. (2007) und Zhang et al. (2008) clustern die eingehenden VAPLs auf Basis ihrer Ähnlichkeit. Dabei wird die strukturelle Ähnlichkeit mittels Tree Edit Distance und die Ähnlichkeit der enthaltenen AVOs u. a. durch syntaktische und semantische Ähnlichkeit ermittelt. Für jedes Cluster wird ein sog. Basisbaum erstellt, der das Cluster repräsentiert, wobei auf diesen Schritt in keiner der Arbeiten im Detail eingegangen wird. Die einzelnen Basisbäume werden zu einem MAPL zusammengefasst indem schrittweise solche Kanten aus der Vereinigungsmenge aller Kanten hinzugefügt werden, die die Baumstruktur des MAPL nicht verletzen. Im Gegensatz zu Jiao et al. (2007) und Zhang et al. (2008) nehmen Zhang & Rodrigues (2009) und Zhang (2012) eine explizite Generalisierung der AVOs in den eingehenden VAPLs vor. Daraus entstehen generalisierte Arbeitsvorgangsklassen (AVKs). Sie fassen VAPLs zu Basisbäumen zusammen, wenn diese die gleichen AVKs sowie die gleichen sequentiellen Beziehungen enthalten. Das

weitere Vorgehen ist i. W. analog zu dem von Jiao et al. (2007) und Zhang et al. (2008). Grundsätzlich können die erstellten Basisbäume als STAs aufgefasst werden. Letztlich sind diese im resultierenden MAPL jedoch nicht mehr erkennbar. Bei der Erstellung des MAPL aus den Basisbäumen und evtl. bereits bei der Erstellung der Basisbäume selbst ist nicht garantiert, dass Vorgängerbeziehungen zwischen AVOs vollständig erhalten bleiben. Deshalb lassen sich nicht zwingend alle eingehenden VAPLs aus dem MAPL konfigurieren. Die Ansätze sind darüber hinaus nur anwendbar, wenn sich die betrachteten VAPLs als Bäume darstellen lassen, was u. a. bedeutet, dass für eine Komponente keine parallelen AVOs ausgeführt werden können.

Mit der Arbeit von Navaei & ElMaraghy (2018) existiert ein auf mathematischer Optimierung basierender Ansatz zur Erstellung von MAPLs aus VAPLs. Die Autoren codieren die VAPLs als Adjazenzmatrizen und bestimmen mittels Optimierung eine Adjanzenzmatrix für einen MAPL, die die geringste euklidische Distanz zu allen Adjazenzmatrizen der VAPLs aufweist. STAs in den VAPLs werden im MAPL nicht abgebildet und Multivorgänge werden nicht berücksichtigt. Eine Informationserhaltung ist auch bei optimaler Ähnlichkeit nicht sichergestellt.

Datenbasierte Erstellung von Vorranggraphen

Vorranggraphen besitzen eine große Bedeutung für die Montageplanung. Sie schränken den Raum aller Montagereihenfolgen auf technisch mögliche Montagereihenfolgen einschließlich Parallelitäten ein und definieren damit den Lösungsraum für eine Optimierung von Montagereihenfolgen (Altemeier et al. 2009, S. 73). Da die manuelle Erstellung von Vorranggraphen aufwändig ist, existieren in der Literatur Ansätze zur datenbasierten Erstellung von Vorranggraphen. Da zu jedem APL genau ein Vorranggraph existiert, der nur diesen APL zulässt, können VAPLs und MAPLs als Vorranggraphen dargestellt werden. Die datenbasierte Erstellung von Vorranggraphen ist deshalb für die Problemstellung der vorliegenden Arbeit relevant.

Wird ein Vorranggraph auf Basis von APLs erstellt, wird zunächst für jedes Paar von AVOs ermittelt, in welcher Abfolge diese in den APLs auftreten. Ist diese Abfolge immer gleich, wird davon ausgegangen, dass eine entsprechende Vorrangbeziehung besteht. Ist diese Abfolge nicht für alle APLs gleich, gibt es zwei alternative Vorgehensweisen. Die erste Vorgehensweise sieht vor, dass in diesem Fall keine Vorrangbeziehung zwischen den beiden AVOs existiert, da beide Abfolgen in gültigen APLs auftreten. Dieser Ansatz wird von Altemeier et al. (2009) eingeführt. Klindworth et al. (2012) integrieren

diesen Ansatz in eine Methode zur Linienaustaktung. Otto & Otto (2014) ergänzen die Methode von Klindworth et al. (2012) in dem sie aufzeigen, wie Vorrangbeziehungen auf Basis von Experteninterviews und im Unternehmen vorhandenen Daten ermittelt werden können. Guiza et al. (2022) entwickeln die Methode von Altemeier et al. (2009) weiter, indem sie auch APLs anderer Produkte berücksichtigen. Dazu ermitteln sie mit Hilfe von Ähnlichkeitsanalysen, welche AVOs aus verschiedenen APLs einander entsprechen. Alle diese Ansätze berücksichtigen keine STAs. Wenn für verschiedene Varianten entgegengesetzte Vorrangbeziehungen existieren, führt dies dazu, dass im resultierenden Vorranggraph keine entsprechenden Vorrangbeziehungen enthalten sind. Damit gehen bestimmte Vorrangbeziehungen in den eingehenden APLs verloren. Multivorgänge werden in keiner der Arbeiten berücksichtigt.

Die zweite Vorgehensweise berücksichtigt demgegenüber STAs. Im Ansatz von Henrioud et al. (2002) wird eine Vorrangbeziehung zwischen zwei Vorgängen i und j nur dann verworfen, wenn eine Folge $\alpha ij\beta$ und eine Folge $\alpha ji\beta$ vorliegen, d. h. wenn die alternative Reihenfolge im selben Kontext auftritt. Treten alternative Reihenfolgen in unterschiedlichen Kontexten auf, sehen die Autoren im resultierenden Vorranggraphen bedingte Vorrangbeziehung vor. Auf Basis des Wissens über bedingte Vorrangbeziehungen im Vorranggraphen ist ersichtlich, welche Vorrangbeziehungen von STAs betroffen sind, jedoch nicht welche in sich konsistenten STAs in den VAPLs auftreten. Aufgrund der hohen Generalität des resultierenden Vorranggraphen lassen sich alle eingehenden VSTLs daraus konfigurieren. Der Ansatz von Henrioud et al. (2002) ist allerdings auf lineare VAPLs beschränkt und berücksichtigt keine Multivorgänge.

Mit den Arbeiten von Mînzu & Bratcu (1999) sowie Bratcu et al. (1999) existiert ein Ansatz, der widersprüchliche Reihenfolgen von AVOs auf alternative Vorranggraphen zurückführt. Die Autoren zeigen, wie die Menge der APLs so partitioniert werden kann, dass für jede Klasse ein widerspruchsfreier Vorranggraph erstellt werden kann. Dies entspricht der Ermittlung optionaler Strukturen in MAPLs. Allerdings berücksichtigt der von Mînzu & Bratcu (1999) entwickelte Algorithmus nur lineare APLs, die darüber hinaus jeweils alle existierenden AVOs enthalten müssen. Multivorgänge werden nicht betrachtet.

Process Discovery

Process Discovery ist ein Teilgebiet des Process Minings, das sich mit der Erstellung von Prozessmodellen auf Basis von Ereignisdaten befasst. Ein Prozessmodell stellt die

Aktivitäten eines Prozesses und deren Beziehungen zueinander dar. Ereignisdaten als Eingangsdaten des Process Discoverys geben an, welche Aktivitäten in einem betrachteten System für welche Prozessinstanz zu welchem Zeitpunkt ausgeführt wurden. Daraus lassen sich für jede Prozessinstanz lineare Abfolgen von Aktivitäten, sog. Traces, ableiten. (van der Aalst 2022)

Ein Prozessmodell kann als eine Darstellungsform eines MAPL interpretiert werden, weshalb die automatische Erstellung von Prozessmodellen prinzipiell für die vorliegende Arbeit relevant ist. Ein wesentlicher Unterschied zur datenbasierten Erstellung von MAPLs besteht jedoch in den Eingangsdaten: Traces sind lineare Abfolgen von Aktivitäten wohingegen APLs auch Parallelitäten aufweisen können.

Es existieren Methoden des Process Discoverys nach Stand der Forschung, die STAs teilweise berücksichtigen können: Sofern in einem Trace eine Aktivität A vor einer Aktivität B und in einem anderen Trace B vor A ausgeführt wird, kann dies durch Algorithmen nach Stand der Forschung erkannt und dargestellt werden (siehe beispielsweise Augusto et al. 2022). Es entsteht ein Zyklus im Prozessmodell, der zwar auf das Vorhandensein von STAs hinweist, aber nicht erkennen lässt, welche in sich konsistenten STAs vorliegen.

3.3.3 Lösungsdefizit

Tabelle 3.3: Stand der Forschung zur datenbasierten Erstellung von Maximalarbeitsplänen

	A3a	A3b	A3c	A3d	A3e
● = Vollständig erfüllt ◐ = Teilweise erfüllt ○ = Nicht erfüllt	**Informationserhaltung**	**Skalierbarkeit**	**Strukturalternativen**	**Multivorgänge**	**Parallelitäten**
Datenbasierte Erstellung von Maximalarbeitsplänen					
Zhang 2012; Zhang & Rodrigues 2009; Zhang et al. 2008; Jiao et al. 2007	○	●	◐	○	○
Navaei & ElMaraghy 2018	○	●	○	○	●
Datenbasierte Erstellung von Vorranggraphen					
Guiza et al. 2022; Otto & Otto 2014; Klindworth et al. 2012; Altemeier et al. 2009	○	●	○	○	●
Henrioud et al. 2002	○	●	◐	○	○
Bratcu et al. 1999; Mînzu & Bratcu 1999	●	●	●	○	○
Process Discovery					
Process Discovery nach van der Aalst (2022) und Augusto et al. (2022)	●	●	◐	●	○

Tabelle 3.3 gibt einen Überblick über den Stand der Forschung zur datenbasierten Erstellung von MAPLs. Die einschlägigen Ansätze beschränkten sich auf APLs mit Baumstruktur und zeigen keine STAs im MAPL. In der Literatur zum Lernen von Vorranggraphen werden STAs überwiegend nicht berücksichtigt. Lediglich die Arbeit von Mînzu & Bratcu (1999) adressiert diesen Aspekt vollumfänglich. Deren Einschränkung auf lineare VAPLs, die jeweils alle existierenden AVOs enthalten limitiert die praktische Anwendbarkeit jedoch stark. Die zu entwickelnde Methode 3 soll die Vorteile der existierenden Methoden kombinieren um die Anforderungen A3a bis A3e vollständig zu erfüllen.

3.4 Problem 4: Datenbasierte Erstellung von Regeln

3.4.1 Anforderungen

Die Regeln eines LLKM entsprechen einem prädiktiven Multi-Label-Modell des überwachten Lernens (SL), wobei jeder abhängige Parameter einem Label entspricht (siehe auch Kapitel 2.3.2). Ebenso, wie ein Multi-Label-Modell nach dem Binary-Relevance-Ansatz in mehrere Single-Label-Modelle zerlegt werden kann, lassen sich die Regeln eines LLKM unabhängig voneinander betrachten. Werden Regeln in Form von Auswahlbedingungen (siehe Kapitel 2.2.2.4) betrachtet, entspricht die datenbasierte Erstellung dieser Regeln mehreren binären Klassifikationsproblemen mit jeweils einem Label. Hierfür existieren verschiedene Verfahren des SL. Für die vorliegende Problemstellung können jedoch nur Verfahren des SL verwendet werden, die interpretierbare Klassifikationsmodelle erstellen **(Anforderung A4a: Interpretierbarkeit)**. Ansonsten wären die resultierenden Regeln und damit das LLKM nicht interpretierbar.

Um eine allgemeine Anwendbarkeit der Methode zu gewährleisten, müssen darüber hinaus alle prinzipiell möglichen logischen Regeln zwischen binären Variablen erkannt werden können **(Anforderung A4b: Logikagnostik)**. Analog zu den Anforderungen der Informationserhaltung in den Kapiteln 3.2.1 und 3.3.1 muss das Modell in der Lage sein, die verwendeten Daten zu reproduzieren. Wird also eine Variante ausgewählt, die in den Trainingsdaten vorhanden ist, müssen die resultierende VSTL und der resultierende VAPL dieselben sein wie in den Trainingsdaten. Das entspricht einer maximalen Vorhersagegenauigkeit auf den Trainingsdaten, die in der Literatur auch als perfekte Trainingsgenauigkeit bezeichnet wird **(Anforderung A4c: Perfekte Trainingsgenauigkeit)**.

Wie in Kapitel 2.3.2 beschrieben, muss bei der Erstellung eines Modells mittels SL i. d. R. ein Kompromiss zwischen hoher Trainingsgenauigkeit und geringer Komplexität gefunden werden. Aufgrund der vorliegenden Problemstellung wird eine perfekte Trainingsgenauigkeit vorausgesetzt. Im Sinne einer guten Generalisierbarkeit ist somit eine minimale Komplexität des Modells anzustreben **(Anforderung A4d: Minimale Komplexität)**. Diese Anforderung trägt auch zur Interpretierbarkeit der gelernten Regeln bei. Zuletzt soll die Methode für allgemeine Fälle und damit auch für Problemstellungen mit vielen Features und großen Datensätzen anwendbar sein. Deshalb sollen Algorithmen für das Training verwendet werden, die hinsichtlich ihrer Recheneffizienz zumindest nicht hinter anderen Arbeiten aus dem Stand der Forschung zurückstehen **(Anforderung A4e: Recheneffizienz)**.

3.4.2 Relevante Arbeiten

Relevant sind, wie oben erläutert, nur Arbeiten, die sich mit binären Singlelabel-Klassifikationsproblemen befassen. Darüber hinaus müssen die gelernten Modelle interpretierbar sein. In der Literatur finden sich drei Arten von interpretierbaren Modellen, die Regeln darstellen oder in Regeln überführt werden können: Entscheidungsbäume, Regellisten (engl. Decision Lists) und Regelmengen (engl. Decision Sets). Regellisten und Regelmengen enthalten Regeln, die jeweils aus einer Bedingung und einer Folgerung bestehen. Die Bedingungen entsprechen jeweils einem Monom, d. h. konjunktiv verknüpften Literalen[32]. Bei der binären Klassifikation entspricht die Folgerung der Zuordnung des Datenpunkts zu einer positiven oder negativen Klasse. Im Folgenden werden die Regeln entsprechend als positiv oder negativ bezeichnet. Bei einer Regelliste ist im Gegensatz zu einer Regelmenge die Reihenfolge der Regeln von Bedeutung. Die Inferenz erfolgt durch die Zuordnung eines Datenpunkts zu einer Klasse gemäß der ersten Regel, deren Bedingung sie erfüllt. Regelmengen im Sinne

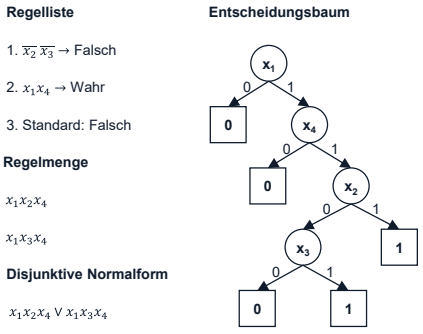


Abbildung 3.1: Interpretierbare Modelle

[32] Literale bezeichnen boolesche Variablen oder deren Negation

von Rudin et al. (2022) enthalten ausschließlich positive Regeln. Eine Zuordnung zur negativen Klasse erfolgt, falls keine der Regeln zutrifft. Eine Regelmenge entspricht somit einem booleschen Ausdruck in disjunktiver Normalform (DNF). Dabei handelt es sich um eine disjunktive Verknüpfung von Monomen. (Rudin et al. 2022, S. 11–16)

Abbildung 3.1 veranschaulicht die Modelle anhand derselben booleschen Funktion. Es sei angemerkt, dass Regellisten und Regelmengen nicht nur von binären Labels, sondern auch von binären Features ausgehen, die sich als boolesche Variablen darstellen lassen. Dies stellt allerdings keine Beschränkung des Anwendungsfalls dar, da sich numerische Features in diskrete Features und diskrete Features in binäre Features umwandeln lassen (Kotsiantis & Kanellopoulos 2006, S. 47–56, Potdar et al. 2017, S. 7–8).

Abweichend hiervon werden in der Logical Analysis of Data – einem Teilgebiet des Operations Research – Regelmengen mit gewichteten positiven und negativen Regeln verwendet (Ouyang & Chou 2020, S. 1). Bei der Inferenz wird das kumulierte Gewicht der erfüllten positiven und negativen Regeln verglichen und ein Datenpunkt derjenigen Klasse mit dem höherem kumulierten Gewicht zugeordnet (Ouyang & Chou 2020, S. 1). Da bei der Interpretation solcher Regelmengen konkurrierende Regeln und Gewichte zu berücksichtigen sind, ist davon auszugehen, dass sie schwer zu interpretieren sind. Sie sind deshalb für die Problemstellung der vorliegenden Arbeit weniger geeignet als Regelmengen im Sinne von Rudin et al. (2022) und werden nicht weiter betrachtet. Im Folgenden liegt deshalb das Verständnis von Regelmengen nach Rudin et al. (2022) zu Grunde, d. h. Regelmengen enthalten ausschließlich positive Regeln.

Sowohl Entscheidungsbäume als auch Regellisten und Regelmengen lassen sich in eine Abfolge von Wenn-Dann-Abfragen überführen und sind damit grundsätzlich in KSs nach Stand der Technik implementierbar. Regelmengen gelten jedoch als besonders einfach zu interpretieren und insbesondere einfacher als Entscheidungsbäume und Regellisten (Lakkaraju et al. 2016, S. 1675). Aufgrund der besseren Interpretierbarkeit befasst sich die vorliegende Arbeit mit Abhängigkeiten in Form von Regelmengen, d. h. booleschen Ausdrücken in DNF. Im Folgenden werden deshalb Ansätze des SL betrachtet, die in der Lage sind, solche Modelle zu lernen.

Heuristiken zur Optimierung der Komplexität von Regelmengen

Viele Ansätze zum Lernen von Regelmengen nutzen Heuristiken, um Regelmengen mit geringer Komplexität zu lernen. An dieser Stelle sei auf Yang et al. (2021) verwiesen,

die einen Überblick über den diesbezüglichen Stand der Forschung geben und eine Heuristik zur datenbasierten Erstellung von Regelmengen entwickeln. Iterativ werden mittels lokaler Suche Regeln aufgestellt, die das gewichtete Mittel aus der Anzahl falsch klassifizierter negativer Datenpunkte, falsch klassifizierter positiver Datenpunkte, der Überlappung positiver Regeln und der Anzahl von Regeln bestmöglich verbessert. Im Allgemeinen können heuristische Verfahren weder eine minimale Komplexität noch eine perfekte Trainingsgenauigkeit garantieren. Eine Ausnahme hiervon bildet die Heuristik „One Clause a Time“ von Triantaphyllou (2006). Sie erstellt boolesche Ausdrücke in Konjunktiver Normalform (KNF) mit möglichst wenigen Klauseln. Eine Erzeugung von Ausdrücken in DNF ist nach demselben Prinzip möglich. Dem Ausdruck wird schrittweise diejenige Klausel minimaler Länge hinzugefügt, die möglichst viele negative Datenpunkte ablehnt, d. h. auf falsch abbildet, und zugleich alle positiven Datenpunkte akzeptiert, d. h. auf wahr abbildet. Der Algorithmus terminiert, sobald alle negativen Datenpunkte durch den Ausdruck abgelehnt werden. Damit wird eine perfekte Trainingsgenauigkeit garantiert. Aufgrund der heuristischen Vorgehensweise kann jedoch ebenfalls keine minimale Komplexität des resultierenden Ausdrucks garantiert werden.

Logikminimierung

Logikminimierung bezeichnet die Minimierung der Komplexität boolescher Ausdrücke (Sasao 2023, S. 12). Um die entsprechenden Methoden für eine Klassifikation zu nutzen, wird der Trainingsdatensatz als Wahrheitstabelle interpretiert, wobei alle nicht vorhandenen Einträge als sog. Don’t Care-Einträge aufgefasst werden. Im Zuge der Logikminimierung werden die Don’t Care-Einträge so gewählt, dass sich möglichst einfache Ausdrücke ergeben. Klassische Verfahren der Logikminimierung, wie insbesondere das Verfahren nach Quine und McCluskey, sind jedoch für große Datenmengen und große Anzahlen von Don’t Care-Einträgen aus Gründen der Recheneffizienz nicht geeignet (Safaei & Beigy 2007, S. 405). Safaei & Beigy (2007) entwickeln eine heuristische Adaption des Quine-McCluskey-Verfahrens mit höherer Recheneffizienz. Sasao (2023) nutzt denselben Ansatz mit neueren Algorithmen der Logikminimierung, kann damit jedoch trotzdem nur kleine Probleme mit weniger als 20 Features lösen.

Exakte Optimierung mittels Ganzzahliger Linearer Optimierung

Dash et al. (2018) und Lawless et al. (2023) entwickeln zwei eng verwandte Ansätze, die mittels ganzzahliger linearer Optimierung (engl. Integer Linear Programming, ILP) eine Regelmenge ermitteln, die den Vorhersagefehler auf den Trainingsdaten

minimiert. Dabei wird die Komplexität – definiert als Anzahl der Literale über alle Regeln – auf eine vorgegebene Höhe beschränkt. Die bei der Optimierung berücksichtigten Regeln werden iterativ mittels **Spaltengenerierung** erstellt. Dadurch wird sichergestellt, dass jede hinzugefügte Regel zur Reduzierung des Zielfunktionswerts beiträgt und dass eine optimale Lösung vorliegt, sobald keine weitere solche Regel mehr gefunden werden kann. Da die Komplexität selbst nicht optimiert wird, weisen die resultierenden Regelmengen im Allgemeinen keine minimale Komplexität auf. Eine perfekte Trainingsgenauigkeit wird nur erreicht, wenn die Komplexität ausreichend hoch beschränkt wird.

Su et al. (2016) nutzen ILP um einen booleschen Ausdruck in KNF zu erstellen, der die gewichtete Summe aus der Gesamtzahl an Literalen und der Trainingsgenauigkeit optimiert. Dabei werden jedoch nur positive Literale berücksichtigt, was die darstellbaren Modelle stark einschränkt. Zudem muss die Anzahl der Klauseln a priori festgelegt werden.

Exakte Optimierung mittels Satisfiability-Solvern

Satisfiability-Solver (SAT-Solver) sind Programme zur Überprüfung der Erfüllbarkeit von booleschen Ausdrücken. Hiermit kann überprüft werden, ob ein boolescher Ausdruck existiert, der bestimmte Anforderungen, wie insbesondere eine bestimmte Komplexität oder eine bestimmte Genauigkeit auf den Trainingsdaten, erfüllt. Zum einen existieren Ansätze von Júnior et al. (2023), Ghosh et al. (2022), Cao et al. (2020), Ghosh & Meel (2019) und Malioutov & Meel (2018), die die beiden genannten Zielkriterien gewichtet optimieren. Prinzipiell ist es möglich, ein Gewicht für die Trainingsgenauigkeit zu ermitteln, das dazu führt, dass die optimale Lösung eine perfekte Trainingsgenauigkeit aufweist. Mit den Ansätzen von Ignatiev et al. (2021), Yu et al. (2020) und Ignatiev et al. (2018) existieren jedoch Verfahren, die speziell für eine Optimierung der Komplexität unter Gewährleistung einer perfekten Trainingsgenauigkeit entwickelt wurden. Es ist deshalb davon auszugehen, dass sie für diese Problemstellung recheneffizienter sind.

Für die vorliegende Arbeit ist der Ansatz von Ignatiev et al. (2021) am relevantesten, da er boolesche Ausdrücke in DNF mit einer minimalen Anzahl von Literalen und einer perfekten Trainingsgenauigkeit bestimmt. Dabei werden zunächst mittels SAT-Solvern alle Monome bestimmt, die alle negativen Datenpunkte ausschließen und nach ausgeschlossenen positiven Datenpunkten nicht von anderen Monomen dominiert werden.

Anschließend wird mittels ILP die komplexitätsminimale Menge von Monomen bestimmt, die alle positiven Datenpunkte akzeptiert. Die Autoren zeigen, dass ihr Ansatz recheneffizienter ist als die unten vorgestellten Ansätze von Yu et al. (2020) und Ignatiev et al. (2018). Der Ansatz ist dennoch nicht in der Lage, mit Rechnern nach Stand der Technik Probleme mit mehr als 800 Datenpunkten in vertretbarer Zeit zu lösen. Der Ansatz sieht vor, dass ein ILP-Problem aufgestellt und gelöst wird, dessen Matrix eine Spalte für jedes nach Problemstellung zulässige Monom enthält. Die optimale Auswahl von Monomen entspricht einem Set-Partitioning-Problem (siehe hierzu Rönnberg & Larsson 2014, S. 532). Nach Stand der Forschung existiert mit Spaltengenerierung (CG) ein effizientes Lösungsverfahren für Set-Partitioning-Probleme, das ohne die Erzeugung aller Spalten auskommt (Rönnberg & Larsson 2014, S. 529). Das Effizienzproblem des Ansatzes lässt sich also auch dadurch erklären, dass Optimierungsverfahren nach Stand der Forschung nicht genutzt werden. Hierfür wäre eine andere Modellierung notwendig.

Nach Ansatz von Yu et al. (2020) werden sowohl positive als auch negative Regeln ermittelt, deren Bedingungen disjunkt sind. Dabei wird die Anzahl der Literale über alle Regeln minimiert. Werden die negativen Regeln vernachlässigt, ergibt sich ein Ausdruck in DNF. Allerdings ist dessen Anzahl von Literalen damit nicht unmittelbar Gegenstand der Optimierung.

Nach dem Ansatz von Ignatiev et al. (2018) wird ein boolescher Ausdruck in DNF ermittelt, der aus einer vorgegebenen Anzahl von Monomen besteht. Diese Anzahl kann iterativ angepasst werden, um einen Ausdruck mit einer minimalen Anzahl von Monomen zu finden. Der resultierende Ausdruck ist damit optimal in Bezug auf die Anzahl der Monome, garantiert aber keine optimale Lösung in Bezug auf die Anzahl der Literale, d. h. die resultierenden Monome selbst können beliebig komplex sein.

Approximative Logiksynthese

Approximative Logiksynthese (engl. Approximate Logic Synthesis, ALS) ist ein Teilgebiet der Informatik, das sich mit dem Entwurf möglichst kostengünstiger logischer Schaltungen befasst, die eine boolesche Funktion möglichst genau abbilden. Wird der Trainingsdatensatz als unvollständige Wahrheitstabelle der zu approximierenden booleschen Funktion aufgefasst, können existierende Verfahren der ALS für die vorliegende Problemstellung verwendet werden. Die entsprechenden Ansätze verfolgen in der Regel nicht das Ziel, Trainingsdaten vollständig korrekt vorherzusagen (siehe z. B.

Ansatz von Boroumand et al. 2021). Costamagna & Micheli (2023) stellen jedoch mehrere ALS-Ansätze vor, die dies ermöglichen und auf schrittweiser Dekomposition basieren. Auch wenn die durch ALS erstellten Ausdrücke i. d. R. gute Ergebnisse hinsichtlich der Komplexität aufweisen, erfolgt keine Optimierung der Komplexität womit insbesondere keine minimale Komplexität garantiert wird.

3.4.3 Lösungsdefizit

Tabelle 3.4: Stand der Forschung zur datenbasierten Erstellung von Regeln

	A4a	**A4b**	**A4c**	**A4d**	**A4e**
● = Vollständig erfüllt ◐ = Teilweise erfüllt ○ = Nicht erfüllt	**Interpretierbarkeit**	**Logikagnostik**	**Perfekte Trainingsgenauigkeit**	**Minimale Komplexität**	**Recheneffizienz**
Heuristische Optimierung der Komplexität					
Yang et al. 2021	●	●	○	○	●
Triantaphyllou 2006	●	●	●	○	●
Logikminimierung					
Sasao 2023, Safaei & Beigy 2007	●	●	●	●	○
Exakte Optimierung der Komplexität					
Dash et al. 2018; Lawless et al. 2023	●	●	◐	○	●
Su et al. 2016	●	○	●	●	●
SAT-Solver					
Cao et al. 2020; Ghosh et al. 2022; Ghosh & Meel 2019; Júnior et al. 2023; Malioutov & Meel 2018	●	●	●	●	◐
Ignatiev et al. 2021	●	●	●	●	◐
Ignatiev et al. 2018; Yu et al. 2020	●	●	●	◐	◐
Approximative Logiksynthese					
Costamagna & Micheli 2023	●	●	●	○	●

Wie Tabelle 3.4 zeigt, kann der Ansatz von Ignatiev et al. (2021) prinzipiell für die datenbasierte Erstellung von Regeln in LLKMs genutzt werden. Er weist jedoch eine Schwäche hinsichtlich seiner Recheneffizienz auf und ist damit für große Probleme nur bedingt geeignet. Mit den Arbeiten von Lawless et al. (2023) und Dash et al. (2018), die ein ähnliches Optimierungsproblem mittels CG lösen, existiert ein Ansatz, der ein Optimierungsverfahren nach Stand der Forschung nutzt. Dieser ist somit effizient, garantiert jedoch keine perfekte Trainingsgenauigkeit. Die zu entwickelnde Methode 4 soll beide Ansätze kombinieren um die Anforderungen A4a bis A3e vollständig zu erfüllen.

3.5 Problem 5: Auswahl von repräsentativen Varianten zur Erweiterung der Datenbasis

Wie in Kapitel 3.1.3 dargestellt, berücksichtigen weder die bestehenden Ansätze zur datenbasierten Erstellung von Konfigurationsmodellen, noch die bestehenden Ansätze zur datenbasierten Erstellung prädiktiver Modelle für die Arbeitsablaufplanung den Fall, dass keine ausreichend große Datenbasis vorliegt. Wie in Kapitel 1.1 beschrieben, ist es grundsätzlich möglich, die Datenbasis vor der Erstellung des KM zu erweitern. Dafür müssen jedoch aus dem Konfigurationsraum systematisch Varianten ausgewählt[33] werden, die einen hohen Informationsgewinn für eine datenbasierte Erstellung der Regeln des Konfigurationsmodells erwarten lassen. Dieses Vorgehen ist für die datenbasierte Erstellung von Regeln in der Literatur noch nicht erforscht[34]. Es weist jedoch eine große Ähnlichkeit mit dem Aktiven Lernen (AL, siehe Kapitel 2.3.2) im Kontext des ML auf. Im Folgenden sollen deshalb ausgewählte Arbeiten des AL betrachtet werden und es soll untersucht werden, inwieweit sich diese für die Anwendung auf die vorliegende Problemstellung eignen.

3.5.1 Anforderungen

Wie in Kapitel 3.4.1 beschrieben, können die Regeln eines LLKM als Multi-Label-Modell im Sinne des SL interpretiert werden. Bei der Auswahl der Variante muss somit der Informationsgehalt dieser Variante für alle Regeln berücksichtigt werden. Dies entspricht einem Multi-Label-AL **(Anforderung A5a: Multi-Label-AL)**.

Eine Herausforderung der vorliegenden Problemstellung ist der typischerweise große Konfigurationsraum mit u. U. 10^{24} zulässigen Varianten (siehe Kapitel 1.1), der einem großen diskreten Featureraum im Sinne des SL entspricht. Da diese Featureräume aufgrund ihrer Größe nicht vollständig enumeriert werden können, müssen mögliche

[33] Die Auswahl einer Variante aus dem Konfigurationsraum entspricht einer zulässigen Kombination von Ausprägungen der Produktmerkmale (siehe Kapitel 2.2.2.1). Auch wenn der Konfigurationsraum nicht zwangsläufig eine diskrete Menge ist, wird in der vorliegenden Arbeit davon gesprochen, dass Varianten aus dem Konfigurationsraum ausgewählt werden.

[34] Der in Kapitel 3.3.2 vorgestellte Ansatz von Guiza et al. (2022) zur Erstellung von Montagevorranggraphen adressiert das Problem einer zu kleinen Datenbasis, indem vorliegende Daten zu verwandten Produkten genutzt werden. Inwieweit diese Idee auf die datenbasierte Erstellung von Regeln übertragen werden kann, bleibt zu untersuchen. In der vorliegenden Arbeit wird dieser Ansatz nicht verfolgt, da er voraussetzen würde, dass im Unternehmen mehrere konfigurierbare Produkte mit ähnlichen Regeln existieren, was den Anwendungsbereich stark einschränkt.

Methoden des AL von nichtenumerierbaren Featureräumen ausgehen **(Anforderung A5b: Nichtenumerierbarer Featureraum)**. Der Konfigurationsraum und damit der Featureraum ist darüber hinaus i. d. R. nicht konvex und kann durch das High-Level-Konfigurationsmodell (HLKM) beliebig beschränkt sein. Dementsprechend müssen mögliche Methoden des Multi-Label-AL mit nichtkonvexen Featureräumen umgehen können **(Anforderung A5c: Nichtkonvexer Featureraum)** und Beschränkungen des Featureraums berücksichtigen **(Anforderung A5d: Beschränkter Featureraum)**.

Wie in Kapitel 3.4.1 beschrieben, sind Modelle in Form boolescher Ausdrücke in DNF für die vorliegende Problemstellung besonders geeignet. Sie unterscheiden sich von vielen Klassifikationsmodellen darin, dass aus ihnen keine Unsicherheit für eine Klassifikationsentscheidung ermittelt werden kann. Geeignete Ansätze dürfen deshalb solche Funktionen nicht voraussetzen **(Anforderung A5e: Boolesche Ausdrücke)**.

3.5.2 Relevante Arbeiten

Die Methoden des Multi-Label-AL nach Stand der Forschung sind sämtlich poolbasiert (siehe Kapitel 2.3.2), d. h. gehen von enumerierbaren Featureräumen aus und sind damit für die vorliegende Problemstellung nicht geeignet. Sie werden deshalb nur der Vollständigkeit halber kurz beleuchtet. Darüber hinaus werden Ansätze zum Single-Label-AL betrachtet, die Membership Query Synthesis (MQS) einsetzen und damit keine vollständige Enumeration des Featureraums voraussetzen. Schließlich wird auf spezielle Methoden des AL, eingegangen, die einen Versionenraum (engl. Version Space, VS) alternativer Modelle nutzen. Diese sind zwar ebenfalls nicht unmittelbar auf die vorliegende Problemstellung anwendbar, das Konzept ist jedoch übertragbar (siehe Kapitel 4.5).

Poolbasiertes Multi-Label-AL

Wu et al. (2020) geben einen umfassende Überblick über Ansätze des Multi-Label-AL. Diese werden vor allem zur Klassifikation von Bildern eingesetzt. In aktuellen Ansätzen werden dabei Datenpunkte multikriteriell nach repräsentationsbasierten und informationsbasierten Kriterien ausgewählt. Bei informationsbasierten Kriterien erfolgt eine Aggregation über alle Labels indem entweder die Ausprägungen der Kriterien aggregiert – z. B. gemittelt – werden oder die Datenpunkte je Label in eine Rangfolge gebracht und die Ränge aggregiert werden (zu letzterem siehe Dwork et al. 2001). Es existieren keine Arbeiten zu Multi-Label-AL, die von einem beschränkten Featureraum ausgehen

oder boolesche Ausdrücke als Modelle verwenden. Da außerdem durchgehend poolbasiertes AL vorliegt, wird ein enumerierbarer Featureraum vorausgesetzt.

AL mit MQS

Ansätze des MQS können nach dem betrachteten Featureraum unterschieden werden: stetig oder diskret. Für stetige Featureräume existieren einerseits Ansätze, die von einem Klassifikator ausgehen, der eine Klassengrenze im Featureraum induziert, wie z. B. eine Support-Vector-Machine oder eine begrenzende Hyperebene eines Halbraums (siehe Englhardt & Böhm 2020, Chen & Fuge 2018, Chen et al. 2017 und Alabdulmohsin et al. 2015). In diesem Fall können Datenpunkte nahe der Klassengrenze gewählt werden. Darüber hinaus existieren Ansätze, die von einer impliziten Klassengrenze ausgehen und Datenpunkte zwischen annotierten Datenpunkten verschiedener Klassen auswählen (Wang et al. 2015 und Xuelei Hu et al. 2012). Beides ist nur für konvexe Featureräume möglich. Beschränkungen des Featureraums oder boolesche Ausdrücke als Modelle werden nicht betrachtet.

Zur Anwendung von MQS auf diskreten Featureräumen existiert nur die Arbeit von Ling & Du (2008). Die Autoren nutzen den Bergsteigeralgorithmus, eine einfache Metaheuristik, um den Featureraum nach dem Datenpunkt mit der höchsten Vorhersageunsicherheit zu durchsuchen. Der Featureraum ist hierbei, abgesehen von den Definitionsbereichen der Feature, nicht beschränkt. Das verwendete Prädiktionsmodell ist ein Entscheidungsbaum.

AL unter Berücksichtigung des Versionenraums

Grundsätzlich kann das Ziel des AL als Verkleinerung eines Versionenraums (VR) alternativer möglicher Modelle verstanden werden. Dieser Ansatz wird beim QBC implizit verfolgt, da die einzelnen Modelle des Komitees als VR aufgefasst werden können. Um das Komitee zu erzeugen, werden beim QBC i. d. R. identische Algorithmen auf verschiedene Teilmengen des Trainingsdatensatzes angewandt (Kumar & Gupta 2020, S. 918). Prinzipiell kann der VR aber auch auf andere Weise aufgestellt werden. Ein für die vorliegende Arbeit relevanter Ansatz stammt von Mitchell (1977), der den VR aus der Menge aller Regeln mit minimaler oder maximaler Spezifität erstellt. Die Spezifität einer Regel ergibt sich aus der Anzahl der annotierten und nichtannotierten Datenpunkte für die sie gilt. Eine Regel ist hierbei ein einziges Monom im Sinne der Aussagenlogik, was die Mächtigkeit der resultierenden Modelle einschränkt. Außerdem geht der Autor nur am Rande darauf ein, wie Anfragen auf Basis des VR generiert werden

können. Der betrachtete Featureraum ist nicht beschränkt und wird für das poolbasierte Sampling als enumerierbar angenommen.

3.5.3 Lösungsdefizit

Tabelle 3.5: Stand der Forschung zur Auswahl von repräsentativen Varianten zur Erweiterung der Datenbasis

	A5a	A5b	A5c	A5d	A5e
● = Vollständig erfüllt ◐ = Teilweise erfüllt ○ = Nicht erfüllt	Multi-Label-AL	Nichtenumerierbarer Featureraum	Nichtkonvexer Featureraum	Beschränkter Featureraum	Boolesche Ausdrücke
Poolbasiertes Multi-Label-AL					
Poolbasiertes Multi-Label-AL (nach Wu et al. 2020)	●	○	●	○	○
AL mit MQS					
Alabdulmohsin et al. 2015; Chen & Fuge 2018; Englhardt & Böhm 2020; Wang et al. 2015; Xuelei Hu et al. 2012	○	●	○	○	○
Ling & Du 2008	○	●	●	○	○
AL unter Berücksichtigung des Versionenraums					
Mitchell 1977	○	○	●	○	◐

Es existiert in der Literatur keine Methode zur Auswahl von repräsentativen Varianten um LLKMs datenbasiert zu erstellen. Wie Tabelle 3.5 zeigt existiert auch in der Literatur zu AL keine hierfür geeignete Methode und auch keine Methode, die durch naheliegende Anpassung geeignet werden würde. Daher soll mit Methode 5 eine neue, problemspezifische AL-Methode entwickelt werden, die die Anforderungen A5a bis A5e erfüllt.

3.6 Problem 6: Datenbasierte Überprüfung von Regeln

3.6.1 Anforderungen

Inkonsistenzen in Regeln wie z. B. die Möglichkeit, zwei sich gegenseitig ausschließende Elemente der MSTL zu wählen, stellen offensichtlich starke Hinweise auf Fehler dar. Sie sollen bei der Überprüfung eines LLKM berücksichtigt werden **(Anforderung**

A6a: Berücksichtigung logischer Inkonsistenzen)[35]. Wie in Kapitel 2.3.3 beschrieben, wird darüber hinaus in vielen Anwendungsfällen anderer Disziplinen Anomalieerkennung eingesetzt, um Fehler zu entdecken. Diese Fehler entsprechen nicht zwingend logischen Widersprüchen in einem Modell. Werden die Regeln eines LLKM als Datenpunkte interpretiert, können einzelne Regeln gegenüber der Gesamtheit an Regeln Anomalien aufweisen. Damit liegen Hinweise auf Fehler vor, die über Konsistenzfehler hinausgehen und deshalb ebenfalls bei der Überprüfung des Modells verwendet werden sollten **(Anforderung A6b: Berücksichtigung von Anomalien)**. Es kann darüber hinaus Fehler im LLKM geben, die sich weder in logischen Inkonsistenzen niederschlagen noch in Anomalien. In Fällen, in denen eine hohe Genauigkeit gefordert ist, muss es möglich sein, mit zusätzlichem Aufwand die Genauigkeit des Modells weiter zu erhöhen, d. h. die Überprüfung muss skalierbar sein **(Anforderung A6c: Skalierbarkeit)**.

3.6.2 Relevante Arbeiten

Entsprechend der in Kapitel 2.2.3.2 eingeführten Einteilung nach Meseguer & Preece (1995, S. 337–339) kann die Literatur zur Überprüfung von KSs und KMs in Inspektion, statische Verifikation, empirisches Testen und Evaluation unterteilt werden. Da die Evaluation die Nutzung des KS und nicht primär das KM betrifft, wird sie im Folgenden nicht betrachtet. Auf bestehende Ansätze zur Inspektion und statischen Verifikation sowie zum empirischem Testen wird im Folgenden eingegangen.

Inspektion

Es existieren Methoden und Darstellungsweisen, die Experten dabei unterstützen, sich in KMs zurecht zu finden, wodurch wiederum die Inspektion unterstützt wird. Felfernig et al. (2013) entwickeln z. B. ein Recommender System für Beschränkungen in beschränkungsbasierten KMs: Benutzern, die sich eine Beschränkung angesehen haben, werden ähnliche Beschränkungen vorgeschlagen. Daneben existieren Möglichkeiten der Visualisierung von Regeln für allgemeine Expertensysteme. Beispielhaft sei auf Baumeister & Freiberg (2011) verwiesen, die Möglichkeiten zur graphischen Darstellung von Abhängigkeiten vorstellen. Darüber hinaus existieren Ansätze, um durch das Lösen von Erfüllbarkeitsproblemen dem Experten Informationen über ein KM

[35] In der Literatur werden logische Inkonsistenzen auch als Anomalien bezeichnet. In der vorliegenden Arbeit ist mit Anomalie jedoch immer Anomalie im Sinne des unüberwachten Lernens gemeint.

bereitzustellen, die für eine Inspektion relevant sein können. Sinz (2004) nennt hierfür folgende Informationen: Gruppen von Produktmerkmalausprägungen, die für jede zulässige Variante gewählt werden müssen und Produktmerkmalausprägungen, die keinen Einfluss auf die Zulässigkeit einer Variante haben. Tidstam et al. (2016) nennen darüber hinaus Komponenten, die immer oder nie gemeinsam auftreten und Produktmerkmalausprägungen, die nach High-Level-Beschränkungen immer gemeinsam auftreten. Diese Informationen dienen dem Verständnis des KM und sind nicht notwendigerweise Hinweise auf Fehler. Grundsätzlich können alle Fehler in einem KM durch eine vollständige und akribische Inspektion gefunden werden, weshalb das Vorgehen skalierbar ist, auch wenn der Aufwand u. U. nicht wirtschaftlich ist.

Statische Verifikation

Sinz (2004) beschreibt folgende Hinweise auf Fehler, die durch das Lösen von Erfüllbarkeitsproblemen ermittelt werden können: Nichtwählbare Produktmerkmalausprägungen nach High-Level-Beschränkungen, Reihenfolgenabhängigkeit der Regelausführung, nichtwählbare Komponenten in der MSTL und das Vorkommen von sich gegenseitig ersetzenden Komponenten in derselben VSTL. Darüber hinaus sieht er das Vorhandensein von nichtwählbaren Produktmerkmalausprägungen als Information für die Inspektion, was jedoch von Tidstam et al. (2016) als Hinweis auf einen Fehler im Sinne der statischen Verifikation angesehen wird. Tidstam et al. (2016) ergänzen außerdem Produktmerkmale für die nur eine Ausprägung gewählt werden kann, verschiedene Produktmerkmalausprägungen, die nur gemeinsam auftreten und Regeln, die bereits von anderen Regeln impliziert werden. Es existieren Arbeiten weiterer Autoren, die dieselben Fehlerhinweise für eine statische Verifikation verwenden, wie z. B. Voronov (2013) und Braun (2021). Die statische Verifikation hat den Vorteil, dass sie tatsächliche Fehler im KM identifizieren kann. Sie ermöglicht jedoch keine vollständige Überprüfung eines KM, da sich nicht alle Fehler in logischen Inkonsistenzen niederschlagen.

Empirisches Testen

Die Herausforderung beim empirischen Testen liegt in der Auswahl der zu konfigurierenden und zu überprüfenden Varianten. Die Arbeit von Glos et al. (2023) behandelt das Testen eines konfigurierbaren Fahrzeugs durch den Bau von Vorserienfahrzeugen. Die Autoren gehen davon aus, dass bestimmte Testfälle definiert sind, wie z. B. bestimmte Komponenten, die neu eingeführt wurden und getestet werden sollen. Mittels

mathematischer Optimierung ermitteln sie die kleinste Anzahl zu produzierender Varianten, die zusammen alle Testfälle abdecken. Einen ähnlichen Fall betrachten Walter et al. (2016), die diejenigen Fahrzeugvarianten ermitteln, die zusammen alle Produktmerkmalausprägungen mindestens einmal realisieren. Die Literatur zum Testen von industriellen KMs ist damit sehr begrenzt. Für die Auswahl von Tests für konfigurierbare Software existieren jedoch zahlreiche Ansätze. Hier sind t-way Teststrategien vorherrschend, die eine minimale Anzahl von Konfigurationen auswählen, so dass jede Kombination der Ausprägungen von t Produktmerkmalen mindestens einmal vorkommt. Dadurch können Fehler ermittelt werden, die aus der Interaktion von Produktmerkmalen resultieren. Medeiros et al. (2016) zeigen eine Übersicht und einen empirischen Vergleich dieser und weiterer gängiger Teststrategien. Wie Medeiros et al. (2016) zeigen, steigt der Aufwand zum Finden von Fehlern beim empirischen Testen exponentiell an, d. h. einige Fehler werden mit wenigen Tests gefunden, das Finden der letzten verbleibenden Fehler erfordert jedoch einen hohen Aufwand. Liegen, wie bei industriellen KMs üblich, Beschränkungen hinsichtlich der wählbaren Konfigurationen vor, müssen diese bei der Zusammenstellung der Tests gemäß der Teststrategie berücksichtigt werden. Für eine Übersicht über entsprechende Verfahren sei auf Wu et al. (2019) verwiesen.

3.6.3 Lösungsdefizit

Tabelle 3.6: Stand der Forschung zur datenbasierten Überprüfung von Konfigurationsmodellen

	A6a	**A6b**	**A6c**
● = Vollständig erfüllt ◐ = Teilweise erfüllt ○ = Nicht erfüllt	**Berücksichtigung von logischen Inkonsistenzen**	**Berücksichtigung von Anomalien**	**Skalierbarkeit**
Inspektion			
Felfernig et al. 2013	○	○	●
Tidstam et al. 2016, Inspektion nach Sinz (2004)	○	○	●
Visualisierung nach Baumeister & Freiberg (2011) u. a.	○	○	●
Statische Verifikation			
Braun 2021, Tidstam et al. 2016, Voronov 2013, Statische Verifikation nach Sinz (2004)	●	○	○
Empirisches Testen			
Glos et al. 2023, Walter et al. 2016	○	○	○
Kombinatorisches Testen von konfigurierbarer Software nach Medeiros et al. (2016) und Wu et al. (2019)	○	○	●

Wie Tabelle 3.6 zeigt, stehen mit der Inspektion und dem empirischen Testen zwei Möglichkeiten zur Verfügung, um KMs prinzipiell vollständig auf Fehler überprüfen zu können. Die Inspektion kann durch Methoden aus der Literatur unterstützt werden. Eine vollständige Überprüfung durch Inspektion stößt jedoch für große KMs an ihre Grenzen. Das empirische Testen geht mit exponentiellem Aufwand für das Finden von Fehlern einher. Durch die Berücksichtigung von Hinweisen aus der statischen Verifikation können Fehler, die sich in logischen Inkonsistenzen niederschlagen, unmittelbar gefunden werden, wodurch die Überprüfung von KMs effizienter wird. Durch eine Anomalieerkennung könnten zusätzliche relevante Hinweise für die Überprüfung gewonnen werden. Ein solcher Baustein zur Überprüfung von KMs existiert jedoch nach Stand der Forschung nicht. Daher soll Methode 6 diese Möglichkeit bereitstellen und damit bestehende Methoden zur Überprüfung von KMs komplementieren.

4 Methoden

Im Folgenden werden die Methoden 1 bis 6 zur Lösung der Probleme 1 bis 6, welche in Kapitel 1.2 eingeführt und in Kapitel 3 konkretisiert wurden, vorgestellt (Abbildung 4.1). Dies entspricht dem dritten Schritt des Design Science Research Process (DSRP, siehe Kapitel 1.3). Ebenso wie Problem 1 den Problemen 2 bis 5 übergeordnet ist, greift die Methode 1 auf die ihr untergeordneten Methoden 2 bis 5 zurück (siehe Kapitel 1.3). Sie beschreibt, wie die Methoden 2 bis 5 in Abhängigkeit des Anwendungsfalls integriert werden können, um Konfigurationsmodelle datenbasiert zu erstellen. Demgegenüber beschreibt Methode 6 zur Lösung von Problem 6, wie Konfigurationsmodelle (KMs) datenbasiert überprüft werden können. Im Folgenden werden die Methoden 1 bis 6 vorgestellt[36].

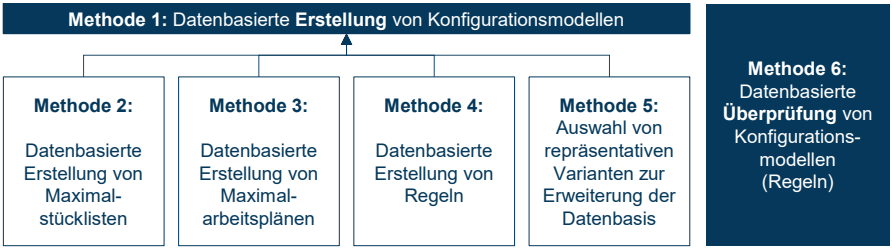


Abbildung 4.1: Übersicht über die Methoden 1 bis 6

4.1 Methode 1: Datenbasierte Erstellung von Konfigurationsmodellen

Im Folgenden wird zunächst das Schema der datenbasiert zu erstellenden Low-Level-Konfigurationsmodelle (LLKMs) eingeführt (Kapitel 4.1.1). Anschließend werden auf Basis des industriellen Kontextes der vorliegenden Arbeit (siehe Kapitel 1.1) Anwendungsszenarien für die datenbasierte Erstellung von LLKMs herausgearbeitet (Kapitel 4.1.2). Abschließend wird dargelegt, wie diese Anwendungsszenarien auf die dem Problem 1 untergeordneten Probleme 2 bis 5 zurückgeführt werden können und wie

[36] Vom Autor der vorliegenden Arbeit wurde bereits ein erster Ansatz veröffentlicht, um Low-Level-Konfigurationsmodelle datenbasiert zu erstellen und zu überprüfen (Frey et al. 2023). Die darin skizzierten Ideen für geeignete Methoden werden z. T. durch die Methoden 1 bis 6 aufgegriffen, ausgearbeitet und weiterentwickelt. Die Methoden 1 bis 6 selbst sind bisher jeweils nicht veröffentlicht und nicht zur Veröffentlichung eingereicht.

die in den folgenden Kapiteln vorgestellten Methoden zur Lösung dieser Probleme durch Methode 1 integriert werden, um LLKMs datenbasiert zu erstellen (Kapitel 4.1.3).

4.1.1 Schema der betrachteten Low-Level-Konfigurationsmodelle

Im Folgenden werden **industrieübliche integrierte KMs** betrachtet, wie sie in Kapitel 2.2.2.4 beschrieben sind. Diese lassen sich in ein High-Level-Konfigurationsmodell (HLKM) und ein LLKM gliedern. Das LLKM besteht aus einer Maximalstückliste (MSTL), einem oder mehreren Maximalarbeitsplänen (MAPLs) sowie Abhängigkeiten in Form von Regeln. Abbildung 4.2 (2) zeigt exemplarisch das im Rahmen der vorliegenden

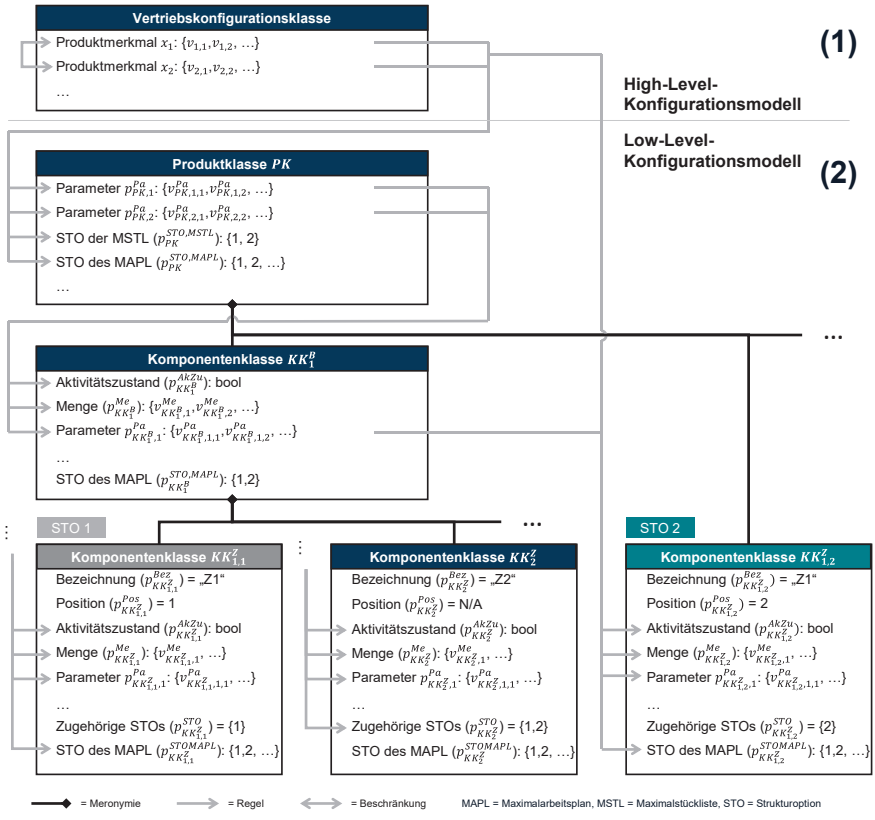


Abbildung 4.2: Schema der Maximalstückliste und deren Abhängigkeiten in der vorliegenden Arbeit

Arbeit entwickelte Schema einer MSTL in Form eines UML-Klassendiagramms sowie mögliche zugehörige Abhängigkeiten als Teil des LLKM. Die Klasse Vertriebskonfiguration in Abbildung 4.2 (1) stellt das HLKM dar und wird im Rahmen des Vertriebskonfigurationsprozesses als Beschreibung einer Variante aus Kundensicht instanziiert. Zwischen den Merkmalen des Produkts im HLKM können Abhängigkeiten in Form von Beschränkungen bestehen.

Die **MSTL** besteht aus einer Klasse PK, aus der das Produkt instanziiert wird sowie Komponentenklassen (KKs), aus denen Komponenten instanziiert werden. Ebenso wie Komponenten in Baugruppen und Zukaufkomponenten (ZKs) eingeteilt werden können, kann eine MSTL Baugruppenklassen (BGKs) und Zukaufkomponentenklassen (ZKKs) enthalten. Die Produktklasse PK, aus der das Endprodukt instanziiert wird, verfügt über Parameter p^{Pa}, die das Endprodukt aus technischer Sicht beschreiben sowie weitere Parameter, die später eingeführt werden. Im Gegensatz zur Produktklasse PK sind KKs nicht zwingend aktiv, d. h. werden nicht zwingend bei der Konfiguration einer variantenbezogenen Stückliste (VSTL) instanziiert. Sie verfügen deshalb über einen Parameter p^{AkZu}, der den Aktivitätszustand der KK angibt und im Rahmen des Konfigurationsprozesses gesetzt wird. Aktive KKs werden zum Abschluss des Konfigurationsprozesses instanziiert. Für aktive KKs gibt der Parameter p^{Me} an, in welcher Menge die zugehörige Komponente in der VSTL in ihre übergeordnete Komponente eingeht. ZKKs verfügen darüber hinaus über eine Bezeichnung (p^{Bez}) und, sofern ZKKs mit identischer Bezeichnung an verschiedenen Positionen der MSTL auftreten, über eine Positionsnummer (p^{Pos}). Ebenso wie die Produktklasse PK können BGKs und ZKKs über Parameter p^{Pa} verfügen, die die Komponenten aus technischer Sicht beschreiben, wie z. B. deren Dimensionen oder Werkstoffe. Parameter von BGKs und ZKKs können, über Wenn-Dann-Regeln, von den Parametern übergeordneter KKs abhängen. Ebenso können die Parameter der Produktklasse PK von den Produktmerkmalen abhängen. Die Produktkonfiguration erfolgt von oben nach unten in der Hierarchie der MSTL. Nachdem die Vertriebskonfiguration vorliegt und somit das Produkt nach dem HLKM konfiguriert ist, wird zunächst die Produktklasse PK entsprechend der zugehörigen Regeln instanziiert, wobei ihre Parameter ausgeprägt werden. Anschließend werden die Komponenten der darunterliegenden Ebene auf Basis ihrer Abhängigkeiten von der Vertriebskonfiguration und der Produktklasse PK instanziiert usw. Baugruppen ohne instanziierte Zukaufkomponenten sind leer und werden aus der resultierenden VSTL

entfernt. Ebenso werden Komponenten entfernt, die nur über genau eine Subkomponente verfügen (siehe Kapitel 2.1).

Das verwendete Schema sieht die Möglichkeit vor, Strukturalternativen der VSTLs in der MSTL abzubilden, was einen wesentlichen Unterschied zu ähnlichen Schemata nach Stand der Forschung darstellt. Aus der MSTL können VSTLs mit unterschiedlichen Strukturen, d. h. unterschiedlichen Fügereihenfolgen der ZKs, konfiguriert werden. Diese alternativen, in der MSTL vorgesehenen Strukturen werden im Folgenden als **Strukturoptionen** (STOs) bezeichnet. Im gezeigten Beispiel existieren zwei STOs, die sich in der Position der Komponente mit Bezeichnung (Parameter p^{Bez}) „Z1“ unterscheiden – in der Abbildung in grün und grau dargestellt. Der Parameter p^{STO} einer ZKK legt fest, für welche STOs diese ZKK instanziiert werden kann: ZKK $KK^{Z}_{1,1}$ kann nur für STO 1 instanziiert werden und $KK^{Z}_{1,1}$ nur für STO 2[37]. Für STO 1 wird also die Komponente Z1 zunächst mit der Komponente Z2 gefügt bevor die resultierende Baugruppe in das Endprodukt eingeht. Für STO 2 geht Z1 unmittelbar in das Endprodukt ein. Es besteht offensichtlich ein enger Zusammenhang zwischen STOs in der MSTL und Strukturalternativen (STAs) in den VSTLs die aus der MSTL konfiguriert werden. Werden zwei VSTLs aus derselben MSTL mit unterschiedlichen STOs konfiguriert, können die beiden VSTLs STAs enthalten. Abbildung 4.3 zeigt für den Beispielfall, dass das jedoch nicht zwingend der Fall sein muss. Würden im Beispielfall zwei VSTLs konfiguriert, die beide Zukaufkomponente Z1 nicht enthalten, würden diese keine STAs

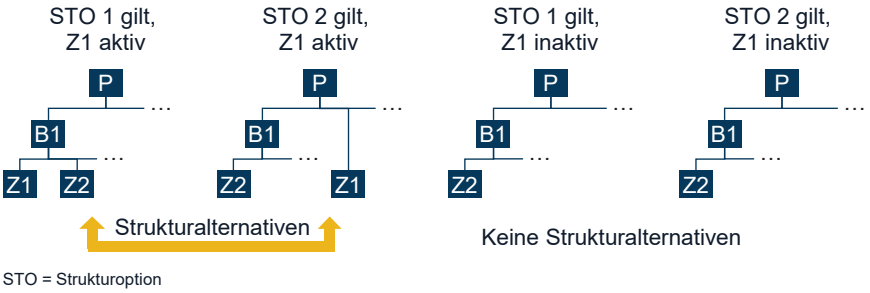


Abbildung 4.3: Konfiguration variantenbezogener Stücklisten mit und ohne Strukturalternativen für den Beispielfall in Abhängigkeit der gültigen Strukturoption

[37] Eine ZKK wird somit im Konfigurationsprozess genau dann instanziiert, wenn ihr Aktivitätszustand dies vorsieht ($p^{AkZu} = wahr$) und wenn die gültige STO ihre Instanziierung zulässt.

aufweisen, auch wenn jeweils von einer anderen STO ausgegangen würde. Verschiedene STOs schlagen sich somit nicht immer in STAs konfigurierter VSTLs nieder. Bei der datenbasierten Erstellung von MSTL entspricht jede STO der MSTL einer STA in den eingegangenen VSTLs. Sind die STAs in den VSTLs technisch begründet (siehe Kapitel 2.2.2.2) können die zugehörigen STOs in der MSTL für die Konfiguration genutzt werden. Die für einen Konfigurationsprozess gültige STO (Parameter $p^{STO,MSTL}$ der Produktklasse PK) kann entweder von der Vertriebskonfiguration abhängen oder von Parametern, die nicht Teil des KM sind, wie z. B. dem produzierenden Werk.

Zwei STOs können sich nicht nur, wie im Beispielfall, in der Position einer Komponente, sondern in den Positionen beliebig vieler Komponenten unterscheiden. Auch STOs, die die Positionen ganzer Baugruppen betreffen können abgebildet werden. Dazu werden BGKs mit identisch bezeichneten Zukaufkomponenten definiert und die ZKKs je BGK jeweils derselben STO zugeordnet. Identisch bezeichnete ZKKs an verschiedenen Positionen der MSTL, im Folgenden **Multipositionen** genannt, müssen im beschriebenen Schema nicht zwingend verschiedenen STOs zugeordnet sein. Sind sie identischen STOs zugeordnet, können sie zugleich instanziiert werden, wodurch Multikomponenten in den aus der MSTL konfigurierten VSTLs auftreten können. In dem im Rahmen der vorliegenden Arbeit verwendeten Schema können STOs nicht nur in MSTLs, sondern auch in MAPLs vorliegen, wie unten ausgeführt wird. Ebenso wie der Parameter $p^{STO,MSTL}$ der Produktklasse PK die gültige STO für die Konfiguration der VSTL angibt, geben die Parameter $p^{STO,MAPL}$ einer Produkt- oder Komponentenklasse die gültige STO für die Konfiguration des zugehörigen variantenbezogenen Arbeitsplans (VAPL) an.

Abbildung 4.4 zeigt exemplarisch das in der vorliegenden Arbeit verwendete Schema eines **MAPL** mit den möglichen Abhängigkeiten zur MSTL in Form von Regeln für die Baugruppe B1 der in Abbildung 4.2 dargestellten MSTL. Jede Arbeitsvorgangsklasse (AVK) verfügt über einen Parameter p^{AkZu}, der angibt, ob der Arbeitsvorgang (AVO) für eine bestimmte Variante aktiv ist, d. h. instanziiert wird. Ggf. verfügt sie über weitere Parameter p^{Pa}, die z. B. den Arbeitsplatz, an dem der AVO ausgeführt wird, das verwendete Werkzeug oder Einstellparameter einer verwendeten Maschine angeben. Die Parameter der AVK können – wie dargestellt – grundsätzlich von den Parametern der zugehörigen Komponentenklasse (KK), sowie deren über- und untergeordneten KKs, der Produktklasse und der Vertriebskonfiguration abhängen. STOs in MAPLs können

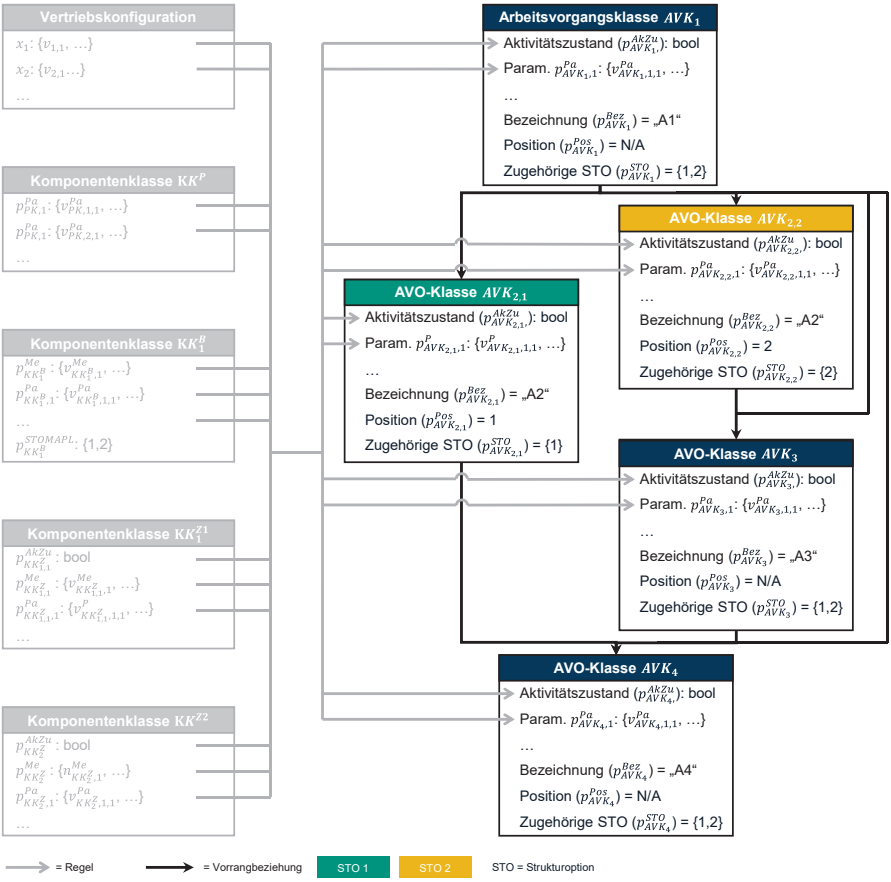


Abbildung 4.4: Schema des Maximalarbeitsplans und dessen Abhängigkeiten in der vorliegenden Arbeit

analog zur Darstellung von STOs in MSTLs durch Duplizierung von AVKs dargestellt werden, wie z. B. im Fall der AVK von AVO A2. In Abhängigkeit der gültigen STO[38] ist in diesem Fall eine parallele Ausführung von A2 und A3 möglich oder nicht. Der MAPL wird als Vorranggraph aufgefasst. Die Beziehungen zwischen seinen AVKs sind Vorrangbeziehungen, d. h. bestimmte AVOs müssen abgeschlossen sein, bevor ein

[38] STO 1 und 2 in Abbildung 4.4 sind STO eines MAPL und entsprechen damit nicht den STO der MSTL in Abbildung 4.2.

anderer bestimmter AVO begonnen werden kann. Vorrangbeziehungen sind transitiv, d. h. da A1 vor A2 auf Position 1 und A2 auf Position 1 vor A4 durchgeführt werden muss, muss auch A1 vor A4 durchgeführt werden. Auf die Darstellung von Vorrangbeziehungen, die durch andere Vorrangbeziehungen impliziert werden, kann jedoch nicht verzichtet werden, da diese im Rahmen der Konfiguration verloren gehen können. Würden z. B. die AVOs A2 und A3 im Konfigurationsprozess nicht instanziiert werden, würde im VAPL keine Vorrangbeziehung zwischen A1 und A4 bestehen und beide AVOs könnten parallel durchgeführt werden.

Da die AVKs von den Parametern der MSTL abhängig sind, erfolgt die **Prozesskonfiguration nach der Produktkonfiguration**. Da keine Abhängigkeiten zwischen den AVKs des MAPL bestehen, kann die Instanziierung der AVKs im Rahmen der Prozesskonfiguration in beliebiger Reihenfolge erfolgen. Dabei werden auch die Parameter der AVOs ausgeprägt. Entspricht ein VAPL nach der Konfiguration einem Graphen ohne Quelle oder ohne Senke, wird ein fiktiver Start- bzw. Endarbeitsvorgang eingefügt, der allen Quellen vorausgeht bzw. allen Senken nachfolgt. VAPLs sind damit immer gerichtete, zyklenfreie Graphen mit einer Quelle und einer Senke.

Für die datenbasierte Erstellung von LLKMs wird in der vorliegenden Arbeit von einem LLKM ausgegangen, das durch eine MSTL, MAPLs und Regeln, wie zuvor beschrieben, definiert ist.

4.1.2 Konkretisierung der Anwendungsszenarien

Im Folgenden werden, wie Abbildung 4.5 zeigt, drei Phasen der Erstellung von LLKMs unterschieden. Die Phasen können jeweils datenbasiert oder manuell durchgeführt oder übersprungen werden. Die Erstellung des LLKM vor Inbetriebnahme des zugehörigen Low-Level-Konfigurationssystems (LLKS) wird in die Phasen Initialisierung und Finalisierung eingeteilt. Bei der **Initialisierung** wird auf Basis bestehender Daten oder manuell ein LLKM erstellt, das grundsätzlich die Konfiguration von VSTLs und VAPLs ermöglicht. Bei der **Finalisierung** wird dieses Modell durch die Generierung zusätzlicher Daten oder manuell verfeinert, sodass die geforderte Genauigkeit der erstellten VSTLs und VAPLs erreicht wird. I. d. R. wird ein LLKM nach Implementierung in einem LLKS noch für einige Zeit begleitet, indem VSTLs und VAPLs, die vom LLKS generiert werden, von Experten überprüft werden. Bei Bedarf werden Änderungen am LLKM vorgenommen. Die **Anpassung** kann dabei, wie heute üblich, manuell erfolgen oder ebenfalls datenbasiert. Bei der datenbasierten Anpassung nehmen Experten zunächst

Korrekturen an der VSTL oder den VAPLs vor. Diese neuen Daten werden anschließend zusammen mit den bestehenden Daten für eine erneute automatische Erstellung des LLKM genutzt. Einzelne Phasen können ausgelassen werden. Z. B. kann auf die Finalisierung vor dem Betrieb des LLKS verzichtet und das LLKM im Betrieb abschließend verfeinert werden. Aus den in Abbildung 4.5 gezeigten Möglichkeiten ergeben sich kombinatorisch 27 verschiedene Anwendungsszenarien für die Erstellung von LLKMs, wobei jedoch nicht alle sinnvoll sind. Im Folgenden wird auf die sinnvollen und für die vorliegende Arbeit relevanten Anwendungsszenarien eingegangen. Die Szenarien werden mit den Abkürzungen aus Abbildung 4.5 bezeichnet.

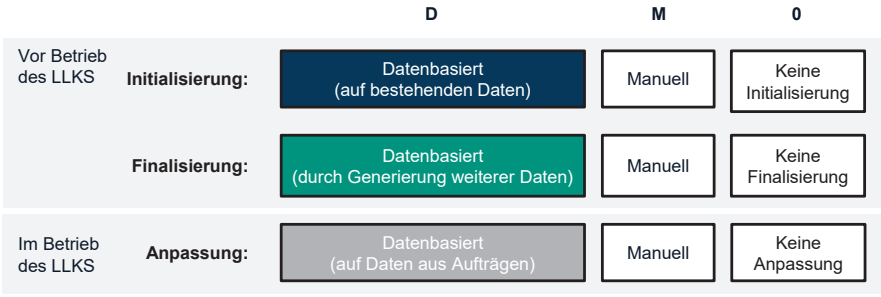


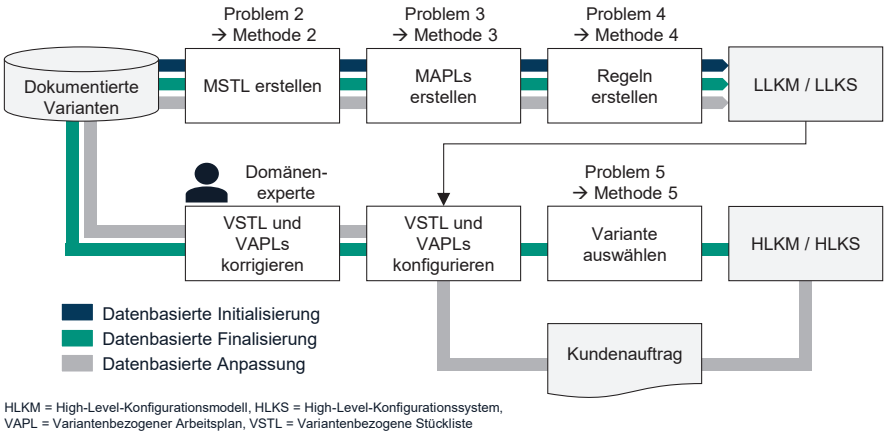
Abbildung 4.5: Möglichkeiten der datenbasierten Erstellung von Low-Level-Konfigurationsmodellen je Phase der Modellerstellung

Im **Szenario DMM** (datenbasierte Initialisierung, manuelle Finalisierung und manuelle Anpassung) werden bestehende Daten genutzt, um ein initiales LLKM zu erstellen. Dadurch kann der Aufwand gegenüber einer rein manuellen Erstellung reduziert werden. Aspekte des LLKM, die von der datenbasierten Methode korrekt erfasst werden, müssen von den Experten nicht mehr eingegeben und können insbesondere nicht übersehen werden. Dieses Szenario ist damit auch für Unternehmen relevant, die über Expertise in der Erstellung von KMs verfügen. Im **Szenario DDD** wird das LLKM vollständig datenbasiert erstellt, sodass prinzipiell keine Expertise hinsichtlich LLKMs benötigt wird. Dieses Szenario ist für Unternehmen relevant, die über wenig Expertise hinsichtlich KMs verfügen oder Schnittstellenprobleme zwischen Domänenexperten und Experten für die Wissensrepräsentation (siehe Kapitel 1.1) ausschließen möchten. In **Szenario D0M** wird das LLKM initial auf Basis bestehender Daten erstellt und im Betrieb manuell angepasst. Auf eine Finalisierung vor Inbetriebnahme des LLKS wird

verzichtet. Dieses Szenario ist u. a. relevant, wenn initial viele Daten zur Verfügung stehen, sodass von einer ausreichend hohen Genauigkeit des automatisch erstellten LLKM auszugehen ist und nur wenige manuelle Anpassungen im Betrieb zu erwarten sind. **Szenario D0D** stellt eine effiziente Alternative zur vollständig manuellen Erstellung von VSTLs und VAPLs auf Basis von Vertriebskonfigurationen dar. Es ist damit für Unternehmen relevant, die bisher ausschließlich einen Vertriebskonfigurator nutzen. Das LLKM wird auf Basis bestehender Daten automatisch erstellt und schlägt VSTLs und VAPLs vor, die durch automatische Anpassung des Modells auf Basis zusätzlicher Daten immer genauer werden. Damit kann ohne Aufwand für die Erstellung eines LLKM der Aufwand für die manuelle Erstellung von VSTLs und VAPLs reduziert werden. Im folgenden Kapitel wird erläutert, wie die Methoden 2 bis 5 eingesetzt werden können, um die datenbasierte Erstellung von LLKMs in den verschiedenen Phasen zu ermöglichen.

4.1.3 Integration der entwickelten Methoden

Abbildung 4.6 stellt das Vorgehen für die drei Phasen der datenbasierten Erstellung eines LLKM bei bestehendem HLKM dar. Bei der **Initialisierung** eines LLKM auf Basis von bestehenden Daten – in Blau dargestellt – wird die Erstellung der MSTL, der



HLKM = High-Level-Konfigurationsmodell, HLKS = High-Level-Konfigurationssystem,
VAPL = Variantenbezogener Arbeitsplan, VSTL = Variantenbezogene Stückliste

Abbildung 4.6: Vorgehen je Phase der datenbasierten Erstellung von Low-Level-Konfigurationsmodellen

MAPLs und der zugehörigen Regeln jeweils einmal durchlaufen. Diesen Schritten liegen die in Kapitel 1.3 beschriebenen Probleme 2, 3 und 4 zugrunde.

Im Rahmen einer datenbasierten **Finalisierung** – in Abbildung 4.6 in Grün dargestellt – wird der zur Verfügung stehende Datensatz erweitert. Dafür werden VSTLs und VAPLs für Varianten erstellt, die systematisch mittels Methode 5 ausgewählt werden. Dieser Prozess verläuft iterativ. Zunächst wird eine Variante ausgewählt, die im Sinne des aktiven Lernens (AL) einen hohen Informationsgewinn verspricht (siehe Kapitel 2.3.2). Dies entspricht dem in der vorliegenden Arbeit betrachteten Problem 5. Das zuvor auf Basis der anfänglich vorhandenen Daten erstellte LLKM ermöglicht bereits eine Konfiguration einer zugehörigen VSTL sowie von VAPLs, wenn auch nicht mit ausreichend hoher Genauigkeit. Die durch das LLKM generierte VSTL und die generierten VAPLs werden von einem Domänenexperten überprüft und bei Bedarf korrigiert. Die überprüften VSTLs und VAPLs gehen zusammen mit der durch ihre Vertriebskonfiguration beschriebenen Variante in den Pool der verfügbaren Daten ein. Mit diesen zusätzlichen Daten kann das LLKM neu erstellt werden, wobei eine höhere Genauigkeit zu erwarten ist. Es wird eine weitere Variante ausgewählt und der Prozess fortgesetzt, bis ein bestimmtes vom Unternehmen festzulegendes Abbruchkriterium erreicht ist. Je höher die Anzahl der Iterationen, desto höher die Genauigkeit des LLKM, desto höher jedoch auch der Aufwand. Die in Kapitel 5.4.2 vorgestellten Ergebnisse der im Rahmen der vorliegenden Arbeit durchgeführten Experimente können für eine Einschätzung der benötigten Datenmenge genutzt werden. Außerdem können je nach Anwendungsfall vergleichbare Experimente an verwandten LLKMs durchgeführt werden, um eine geeignete Datenmenge zu ermitteln. In diesem Prozess müssen VSTLs und VAPLs durch die Domänenexperten nicht neu erstellt, sondern nur überprüft und evtl. korrigiert werden. Damit ist der Aufwand für die Bereitstellung einer bestimmten Menge von Dokumenten nicht mit dem Aufwand für die Erstellung dieser Menge von Dokumenten gleichzusetzen. Darüber hinaus nimmt der Aufwand für die Korrektur der VSTLs und VAPLs mit jedem Durchlauf ab, da die durch das LLKM erstellten Dokumente immer genauer werden.

Die datenbasierte **Anpassung** des LLKM im Betrieb – in Abbildung 4.6 in Grau dargestellt – verläuft analog zur datenbasierten Finalisierung. Dabei werden jedoch Varianten nicht systematisch ausgewählt, sondern ergeben sich aus Kundenaufträgen.

Bei allen drei Schritten werden die Methoden 2, 3 und 4 nacheinander durchgeführt. Abbildung 4.7 zeigt, wie diese aufeinander aufbauen[39]. Zunächst liegt je Variante eine Vertriebskonfiguration, d. h. eine Ausprägung der Produktmerkmale, sowie eine VSTL und je Eigenfertigungskomponente dieser Variante ein VAPL vor (1). Die Gesamtheit der VSTLs dient als Basis für die datenbasierte Erstellung der MSTL mittels Methode 2 (2). Im Zuge der Erstellung der MSTL werden Komponenten der VSTLs, die dieselbe Funktion erfüllen, zusammengefasst, wie z. B. die Baugruppen $B1$, $B3$ und $B5$. Die VAPLs zusammengefasster Komponenten bilden einen Datenpool für die Erstellung von MAPLs für diese Komponenten mittels Methode 3. Die Erstellung von MAPLs erfolgt

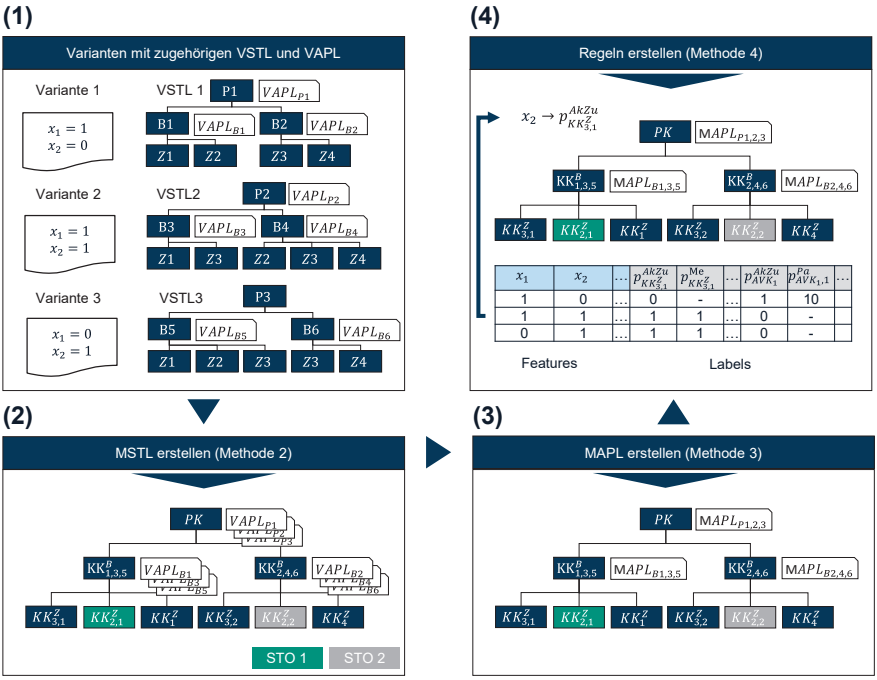


Abbildung 4.7: Integration der Methoden 2, 3 und 4

[39] Abbildung 4.7 nutzt gegenüber der Darstellung in Kapitel 4.1.1 eine vereinfachte Darstellung, die Komponenten lediglich mit ihrer Position und Bezeichnung und KK lediglich mit ihrem Namen darstellt. Dies bedeutet nicht, dass die Objekte bzw. Klassen keine weiteren Parameter aufweisen können.

somit für das Produkt und jede Eigenfertigungskomponente, wobei die Reihenfolge der Betrachtung beliebig ist. Die resultierenden MAPLs werden den entsprechenden Komponenten der MSTL zugeordnet (3). Liegen die MSTL und die MAPLs vollständig vor, sind alle Elemente des LLKM sowie deren Parameter bekannt und es können Abhängigkeiten, in Form von Regeln zwischen diesen, datenbasiert erstellt werden (4). Dafür werden die Parameter des KM, d. h. des HLKM, der MSTL und der MAPLs, in einer Tabelle zusammengefasst, wie in Abbildung 4.7 (4) beispielhaft zu sehen. Für jeden Parameter wird eine Regel[40] erstellt, die angibt, wie der Parameter von den Produktmerkmalen oder den anderen Parametern des KM abhängt. Im Sinne des überwachten Lernens (SL) (siehe Kapitel 2.3.2) stellt der zu prädizierende Parameter ein Label dar. Die Merkmale oder Parameter, von denen er abhängt, entsprechen Features, d. h. beim Lernen von Regeln liegt ein Single-Label-Problem vor.

In den folgenden Kapiteln werden die hier eingeordneten Methoden 2 bis 5 zur Umsetzung von Methode 1 erläutert.

4.2 Methode 2: Datenbasierte Erstellung von Maximalstücklisten

Im Folgenden wird die im Rahmen der vorliegenden Arbeit entwickelte Methode 2 zur Lösung des Problems 2 – der datenbasierten Erstellung von MSTLs – vorgestellt. Methode 2 behebt das in Kapitel 3.2.3 beschriebene Lösungsdefizit nach Stand der Forschung. Ausgehend von einer Menge S^{VSTL} von VSTLs (siehe Kapitel 4.1.3, Abbildung 4.7) wird eine MSTL erstellt. Es wird davon ausgegangen, dass jede ZK einer VSTL eine **Bezeichnung** aufweist, wie z. B. eine Materialnummer in einem Enterprise-Resource-Planning-System (ERP-System), die ihren Typ wie z. B. „Halterung 123“ eindeutig festlegt. Abbildung 4.8 gibt einen Überblick über die 5 Schritte der Methode 2. In Schritt 1 wird eine MSTL erstellt, indem Komponenten zu KKs zusammengefasst werden – zunächst ohne Berücksichtigung der Parameter der KKs. Die MSTL wird so erstellt, dass sie hinsichtlich der Anzahl ihrer ZKKs minimal ist. In Schritt 2 wird ermittelt, welche STOs in der MSTL vorzusehen sind, um die STAs der VSTLs aus S^{VSTL} abzubilden. In Schritt 3 wird durch einen Experten überprüft, ob die ermittelten STOs Hinweise auf Inkonsistenzen im Datensatz darstellen oder tatsächlich im Konfigurationsprozess zu berücksichtigen sind. In Schritt 4 werden die Parameter der KKs und deren

[40] Eine solche Regel muss nicht genau einem Monom entsprechen, wie die Regeln einer Regelmenge im Sinne von Rudin et al. (2022, siehe Kapitel 3.4.2). Sie kann z. B. auch eine Disjunktion mehrerer Monome sein und damit einer Regelmenge entsprechen.

mögliche Ausprägungen aus den Parametern ihrer originären Komponenten und deren Ausprägungen abgeleitet. In Schritt 5 werden ähnliche ZKKs durch eine Superklasse generalisiert. Die Schritte werden nacheinander durchlaufen, wobei die Schritte 3 und 5 optional sind. Abschließend liegt eine MSTL, wie in Kapitel 4.1.1 beschrieben, vor, aus der alle VSTLs aus S^{VSTL} konfiguriert werden können. Die fünf Schritte werden im Folgenden erläutert.

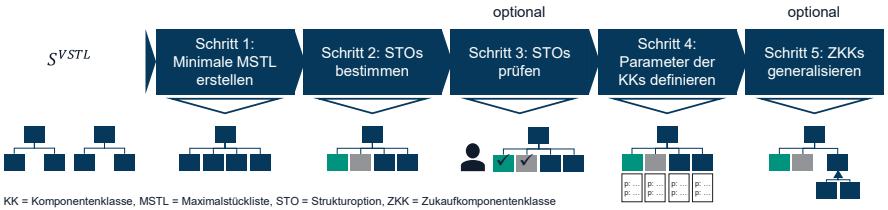


Abbildung 4.8: Überblick über die Schritte der Methode 2

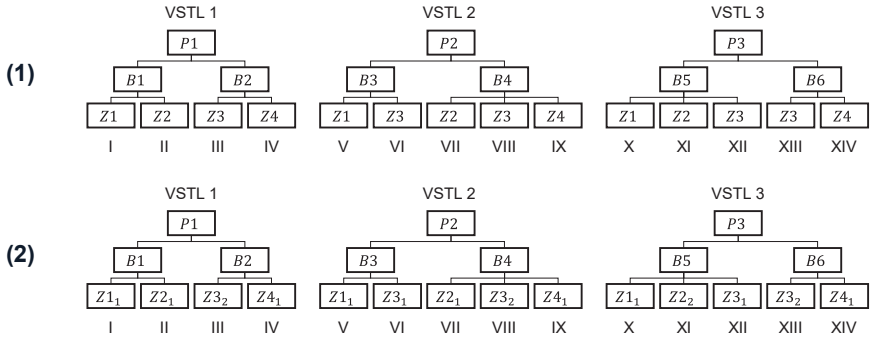
4.2.1 Schritt 1: Minimale Maximalstückliste erstellen

Schritt 1 greift auf den im Rahmen der vorliegenden Arbeit entwickelten und in Anhang 3.1 beschriebenen Algorithmus **Alg^{MSTL}** zurück. Dieser ist in der Lage, aus einer Menge von VSTLs ohne Multikomponenten und ohne STAs eine MSTL zu erstellen, aus der alle eingehenden VSTLs konfiguriert werden können. Für VSTLs mit Multikomponenten ist Alg^{MSTL} nicht anwendbar. Für VSTLs mit STAs gibt Alg^{MSTL} keine Lösung zurück, wodurch Alg^{MSTL} auch genutzt werden kann, um zu überprüfen, ob STAs in einer Menge von VSTLs vorliegen. Aufgrund seiner Einschränkungen kann er jedoch nicht unmittelbar genutzt werden, um aus einer Menge von VSTLs mit Multikomponenten oder STAs eine MSTL zu erstellen.

Die **Idee** hinter dem ersten Schritt der Methode ist deshalb, die gegebenen VSTLs aus S^{VSTL} so zu adaptieren, dass sie keine Multikomponenten und keine STAs mehr aufweisen und anschließend daraus mittels Alg^{MSTL} eine MSTL zu erstellen. Abbildung 4.9 veranschaulicht diese Idee. Die ZKs der initial vorliegenden VSTLs 1, 2 und 3 sind zur Referenz römisch nummeriert. Die VSTLs in Abbildung 4.9 (1), weisen Multikomponenten auf, wie z. B. ZK $Z3$ in VSTL 2. Außerdem tritt die Komponente $Z2$ in VSTL 2 und VSTL 3 an verschiedenen Positionen auf. Somit entsprechen VSTL 2 und VSTL 3 entweder verschiedenen STAs oder es können bis zu zwei Positionen $Z2$ im Endprodukt

existieren, wobei je eine andere bei VSTL 2 und VSTL 3 aktiv ist. Alg^{MSTL} ist für den Beispielfall nicht unmittelbar anwendbar.

Ex ante ist nicht bekannt, welche ZKs der VSTLs Instanzen derselben Klasse der zu erstellenden MSTL sind. Dies kann jedoch – wie im Folgenden erläutert – prognostiziert werden. Auf Basis dieser Prognose können ZKs über ihre Bezeichnung hinaus mit einer Nummer annotiert werden, die der Position ihrer Klasse in der MSTL entspricht. Diese Nummer wird im Folgenden **Klassennummer** (KN) genannt. Durch die Klassennummern werden identisch bezeichnete ZKs unterscheidbar, wodurch Multikomponenten und STAs aufgelöst werden können. Die Subskripte der Zukaufkomponentenbezeichnungen in Abbildung 4.9 (2) entsprechen den KNs der ZKs. Die Verbindung aus Bezeichnung und KN einer ZK wird im Folgenden als **Label** dieser ZK bezeichnet. Verfügt eine ZK noch über keine KN entspricht ihr Label ihrer Bezeichnung. Nach vollständiger Annotation wären alle ZKs einer VSTL durch ihre Labels unterscheidbar, sodass keine Multikomponenten mehr vorlägen. Da außerdem ZKs derselben Klasse immer an derselben Position über alle VSTLs hinweg aufträten, lägen auch keine STAs mehr vor. Um also mittels Alg^{MSTL} eine MSTL erstellen zu können, ist eine Annotation der ZKs mit KNs zu bestimmen, sodass keine Multikomponenten und keine STAs vorliegen. Eine solche Annotation wird im Folgenden als zulässige Lösung bezeichnet. Jedes in den annotierten VSTLs auftretende Label entspricht abschließend einer ZKK der zu erstellenden MSTL.



VSTL = Variantenbezogene Stückliste

Abbildung 4.9: Umwandlung von variantenbezogenen Stücklisten mit Multikomponenten und Strukturalternativen (1) in variantenbezogene Stücklisten ohne Multikomponenten und ohne Strukturalternativen (2)

Es existiert immer eine triviale zulässige Lösung, in der alle ZKs über alle VSTLs hinweg mit unterschiedlichen KNs annotiert sind. Aus dieser resultiert jedoch eine große Anzahl verschiedener Labels und damit eine große Anzahl von ZKKs in der MSTL. Darüber hinaus könnten aus dieser MSTL ausschließlich die gegebenen VSTLs sowie Teilgraphen davon abgeleitet werden. Im Sinne des maschinellen Lernens (ML) weist diese Lösung eine Überanpassung an die gegebenen Daten auf. Nach dem in Kapitel 2.3.2 eingeführten Prinzip wird deshalb das **komplexitätsminimale Modell**, das die Daten abbildet, d .h. diejenige MSTL mit der geringsten Anzahl von ZKKs, aus der alle gegebenen VSTLs konfiguriert werden können, gesucht. Die in Abbildung 4.9 (2) dargestellte Lösung ist zulässig und geht von sechs ZKKs aus – je eine ZKK $Z1$ und $Z4$ und je zwei ZKK $Z2$ und $Z3$. Eine zulässige Lösung mit weniger als sechs ZKKs existiert in diesem Fall nicht, weshalb die dargestellte Lösung optimal ist. Es ist prinzipiell möglich, eine optimale Lösung durch vollständige Enumeration zu ermitteln. Dabei würde für alle möglichen Lösungen überprüft, ob diese zulässig sind und abschließend diejenige Lösung mit der geringsten Anzahl unterschiedlicher Labels ausgewählt. Die Anzahl möglicher KNs je Bezeichnung ist nur durch die Anzahl von ZKs[41] mit dieser Bezeichnung über alle gegebenen VSTLs hinweg begrenzt ist. Dadurch wächst die Menge zu überprüfender Lösungen exponentiell mit der Anzahl der ZKs in den gegebenen VSTLs. Eine solche triviale Methode scheitert deshalb für relevante Problemstellungen an der Recheneffizienz. Schritt 1 der Methode nutzt deshalb den effizienten Algorithmus **Alg$^{\text{MinMSTL}}$** zur optimalen Annotation von ZKs. Nach der Anwendung von Alg$^{\text{MinMSTL}}$ wird mittels Alg$^{\text{MSTL}}$ eine MSTL erstellt, aus der alle VSTLs aus S^{VSTL} konfiguriert werden können. Alg$^{\text{MinMSTL}}$ findet sich in Anhang A3.3 als Pseudocode. Er besteht aus einer Initialisierung sowie Iterationen, was in den folgenden Kapiteln erläutert wird.

4.2.1.1 Initialisierung von Algorithmus Alg$^{\text{MinMSTL}}$

Alg$^{\text{MinMSTL}}$ beginnt mit einer Initialisierung, in der die Reihenfolge festgelegt wird, in der die ZKs aus den VSTLs annotiert werden.

[41] Wenn hier und im Folgenden von der Anzahl von Komponenten gesprochen wird, ist damit immer die Anzahl der entsprechenden Objekte in der STL gemeint. Die Menge, in der die Komponenten je Position in ihre übergeordneten Komponenten eingehen, ist – wie in Kapitel 4.1.1 beschrieben – ein Attribut des Objekts und wird dabei nicht berücksichtigt.

4.2.1.1.1 Problematik der Betrachtungsreihenfolge

Die Entscheidung über die Annotation von ZKs kann sequenziell getroffen werden. Damit ergibt sich ein Suchbaum, in dessen Knoten – im Folgenden Entscheidungsknoten genannt – jeweils über die Annotation einer ZK entschieden wird. Da eine ZK durch die Annotation ein neues Label erhält, kann jede Entscheidung auch als Entscheidung über das neue Label der ZK verstanden werden. In der Wurzel des **Suchbaum**s wurden noch keine, in den anderen inneren Knoten einige und in den Blättern alle ZKs in VSTLs aus S^{VSTL} annotiert. Jedes Blatt des Suchbaums entspricht somit einer Lösung. Den Suchbaum vollständig zu traversieren entspricht der oben beschriebenen vollständigen Enumeration und ist nicht effizient. Die vorliegende Methode nutzt deshalb zum einen eine heuristische **Tiefensuche**, um schnell gute Lösungen zu finden. Zum anderen nutzt sie **Pruning**, um Teilbäume des Suchbaums von der Betrachtung auszuschließen, die keine besseren Lösungen als die beste bereits bekannte Lösung enthalten können.

Welche KN der betrachteten ZK in einem Entscheidungsknoten zugeordnet wird entscheidet darüber mit welchem untergeordneten Entscheidungsknoten die Suche fortgesetzt wird. Jeder untergeordnete Entscheidungsknoten entspricht der Wurzel eines Teilbaums des gesamten Suchbaums. Nach dem Prinzip der Tiefensuche, das in der Methode angewandt wird, wird zunächst dieser Teilbaum vollständig durchsucht. Falls auf einer hohen Ebene des Suchbaums eine ungünstige Annotation vorgenommen wird, führt das dazu, dass viel Rechenkapazität dafür aufgewandt wird, Teilbäume des Suchbaums zu durchsuchen, die keine guten Lösungen enthalten. Deshalb ist die **Betrachtungsreihenfolge** der ZKs, die im Rahmen der Initialisierung festgelegt wird, entscheidend für die Effizienz des Algorithmus. Um Fehlentscheidungen auf hohen Ebenen zu vermeiden, ist es sinnvoll, zunächst über diejenigen Annotationen zu entscheiden, für die die Wahrscheinlichkeit falscher Entscheidungen gering ist. Insbesondere lassen sich ZKs mit Bezeichnungen, für die nur eine oder wenige Klassen in der zu erstellenden MSTL existieren, mit höherer Sicherheit korrekt annotieren als solche, für die viele Klassen existieren. Die Anzahl der Klassen je Bezeichnung in der MSTL ist jedoch ex ante nicht bekannt, da die zu erstellende MSTL nicht bekannt ist. Im zuvor eingeführten Beispiel ist z. B. bekannt, dass es mindestens zwei Klassen mit Bezeichnung $Z3$ in der MSTL geben muss, da z. B. die beiden Komponenten $Z3$ in VSTL 2 nicht aus derselben Klasse instanziiert worden sein können. Da die Bezeichnung $Z3$ insgesamt fünfmal auftritt, können jedoch auch bis zu fünf Klassen $Z3$ in der MSTL vorliegen.

Analog gilt z. B., dass die Klassenanzahl für $Z2$ in der MSTL zwischen 1 und 3 liegen muss. Um im Rahmen der Initialisierung von Alg$^{\text{MinMSTL}}$ eine effiziente Betrachtungsreihenfolge festlegen zu können, wird deshalb zunächst je Bezeichnung l die Anzahl n_l^{PrKl} an ZKKs mit dieser Bezeichnung in der MSTL prognostiziert. Dafür wird ein **Distanzmaß** zwischen ZKs derselben Bezeichnung verwendet, um einschätzen zu können, welche ZKs wahrscheinlich aus derselben ZKK der letztlichen MSTL instanziiert werden. Dieses wird im Folgenden vorgestellt.

4.2.1.1.2 Distanzen von Zukaufkomponenten

Das entwickelte Distanzmaß basiert auf der **Kontextähnlichkeit** von ZKs. Abbildung 4.10 zeigt die Distanzen für den Beispielfall und eine beispielhafte Berechnung. Da VSTLs eine Baumstruktur aufweisen, verfügt jedes Blatt, d. h. jede ZK, über genau einen Pfad zur Wurzel, d. h. zum Produkt. Ausgehend von der ZK selbst, wird die ZK in jedem Knoten ihres Pfads mit anderen ZKs gefügt. Z. B. wird ZK I mit Bezeichnung $Z1$ zunächst im Knoten der der Baugruppe $B1$ entspricht mit der ZK II ($Z2$) gefügt und anschließend im Knoten der dem Produkt entspricht mit den ZKs III ($Z3$) und IV ($Z4$).

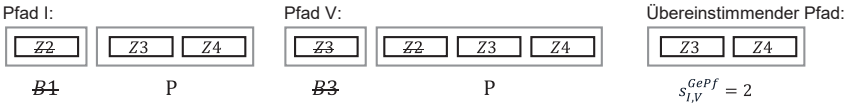
$Z1$	I	V	X
I	0	0,5*	0
V	0,5	0	0,33
X	0	0,33	0

$Z2$	II	VII	XI
II	0	0,5	0
VII	0,5	0	0,67
XI	0	0,67	0

$Z3$	III	VI	VIII	XII	XIII
III	0	0,5	1	0	0,5
VI	0,5	0	0,67	0,33	0,67
VIII	1	0,67	0	0,67	0,33
XII	0	0,33	0,67	0	0,67
XIII	0,5	0,67	0,33	0,67	0

$Z4$	IV	IX	XIV
IV	0	0,5	0
IX	0,5	0	0,33
XIV	0	0,33	0

*Beispiel: Berechnung der Distanz $d_{I,V}^{ZK}$ von I und V:



Potentielle Übereinstimmung

	Anz. $Z2$	Anz. $Z3$	Anz. $Z4$
I	1	1	1
V	1	2	1
Min	1	1	1

$\hat{s}_{I,V}^{GePf} = 3$

Relative Ähnlichkeit:

$$s_{I,V}^{ZK} = \frac{max\left(0; s_{I,V}^{GePf} - 1\right)}{\hat{s}_{I,V}^{GePf} - 1} = \frac{\max(0; 2-1)}{3-1} = 0{,}5$$

Distanz:

$$d_{I,V}^{ZK} = 1 - s_{I,V}^{ZK} = 0{,}5$$

Abbildung 4.10: Berechnung der Distanzen zwischen Zukaufkomponenten auf Basis ihrer Kontextähnlichkeiten im Rahmen von Alg$^{\text{MinMSTL}}$ für den Beispielfall

Um die Distanz zweier ZKs i und j zu ermitteln, wird der Grad $s_{i,j}^{GePf}$ der Übereinstimmung zwischen ihren Pfaden ermittelt, d. h. die Übereinstimmung ihrer Fügereihenfolge. Es werden dafür aus den Knoten beider Pfade so wenig Labels wie möglich entfernt, um beide Pfade identisch werden zu lassen. $s_{i,j}^{GePf}$ entspricht der Anzahl der Labels im resultierenden gemeinsamen Pfad. Die Pfade I und V stimmen beispielsweise überein, wenn aus dem Pfad der ZK I (Pfad I) der Knoten $B1$ und aus dem Pfad der ZK V (Pfad V) der Knoten $B3$ sowie aus dem Knoten P das Label $Z2$ entfernt wird. Beide Pfade enthalten nun einen Knoten, der wiederum zwei Labels – $Z3$ und $Z4$ – enthält. Damit ergibt sich $s_{I,V}^{GePf} = 2$.

Die Berechnung des gemeinsamen Pfads mit der größten Übereinstimmung weist Ähnlichkeit zu einem **Longest Common Subsequence-Problem**[42] auf. Es kann ebenso wie das Longest Common Subsequence-Problem mittels **dynamische**r **Optimierung** effizient gelöst werden. Eine Erläuterung sowie der Pseudocode des entsprechenden, im Rahmen der vorliegenden Arbeit entwickelten Algorithmus, **Alg**$^{\textbf{GemPfad}}$, findet sich in Anhang A3.2. Der Grad der Übereinstimmung wird normalisiert, indem er durch den Grad der maximal möglichen Übereinstimmung $\hat{s}^{GePf}$ dividiert wird. Der gemeinsame Pfad kann nicht mehr ZKs eines Labels enthalten als die geringere Anzahl von ZKs mit diesem Label in den beiden Pfaden. Es gilt also für die Pfade zweier ZKs i und j im Allgemeinen

$$\hat{s}_{i,j}^{GePf} = \sum_{k \in S^{LaZK}} \min(n_{i,k}^{PfLa}, n_{j,k}^{PfLa}) \quad , \tag{4.1}$$

wobei S^{LaZK} der Menge aller Labels von ZKs in den VSTLs in S^{VSTL} entspricht und $n_{i,k}^{PfLa}$ einen Parameter darstellt, der angibt, wie oft ein Label k in einem Pfad i auftritt. Enthält der gemeinsame Pfad genau ein Label, liegt keine Übereinstimmung im Sinne einer identischen Abfolge von Labels vor. Um das zu berücksichtigen, werden $s_{i,j}^{GePf}$ und sein Normierungsdivisor $\hat{s}_{i,j}^{GePf}$ um 1 korrigiert. Für $\hat{s}_{i,j}^{GePf} > 1$ ergibt sich damit folgendes normiertes Maß $s_{i,j}^{ZK}$ für die Ähnlichkeit zweier ZKs i und j:

$$s_{i,j}^{ZK} = \frac{\max(0, s_{i,j}^{GePf} - 1)}{\hat{s}_{i,j}^{GePf} - 1}. \tag{4.2}$$

[42] Für Longest Common Subsequence-Probleme sei auf Bergroth et al. (2000, S. 39–40) verwiesen.

Für $\hat{s}_{i,j}^{GePf} \leq 1$, d. h. falls beide Pfade zu wenige gemeinsame Labels enthalten um eine Aussage über ihre Ähnlichkeit treffen zu können, wird $s_{i,j}^{ZK} = 1$ angenommen. Da $s_{i,j}^{ZK}$ normiert ist, kann durch

$$d_{i,j}^{ZK} = 1 - s_{i,j}^{ZK} \quad 4.3$$

eine normierte Distanz zweier ZKs i und j bestimmt werden. Das so berechnete Distanzmaß lässt eine Aussage über die **Unähnlichkeit des Kontextes**, in denen Komponenten in ihren VSTLs auftreten, zu. Es wird eher davon ausgegangen, dass zwei ZKs nicht aus derselben ZKK der MSTL instanziiert werden, wenn sie in unterschiedlichen Reihenfolgen mit anderen ZKs gefügt werden. Eine Herausforderung bei der Ermittlung der Ähnlichkeiten von Pfaden in VSTLs ist, dass zu Beginn alle Labels der ZKs den Bezeichnungen der ZKs entsprechen. Es ist somit nicht klar, ob identische Labels in den Pfaden auch dieselben KKs in der MSTL referenzieren, d. h. die Referenzen des Maßes sind verzerrt. Je größer der Datensatz, desto geringer ist der Einfluss von unsystematischer Nichtübereinstimmung und desto genauer ist das Maß. Außerdem steigt die Güte des Maßes, je mehr Annotationen vorliegen.

Sind für eine bestimmte Bezeichnung die Distanzen der zugehörigen ZK bekannt, kann die Anzahl von ZKKs je Bezeichnung prognostiziert werden.

4.2.1.1.3 Prognose der Anzahl von Zukaufkomponentenklassen je Bezeichnung

Elemente auf Basis ihrer Distanzen zu gruppieren ist als **Clustering** eine zentrale Problemstellung des unüberwachten Lernens (UL, siehe 2.3.3). Die Prognose der Anzahl von Klassen je Bezeichnung kann damit auf die Ermittlung einer optimalen Clusteranzahl im Rahmen des UL zurückgeführt werden. Die maximale Anzahl von Clustern, n^{ClMax}, ist trivial durch die Anzahl zu clusternder ZKs gegeben. Entsprechend des in der Literatur gebräuchlichen Vorgehens (siehe z. B. Xu et al. 2016, S. 1493) wird für jede Clusteranzahl zwischen 1 und n^{ClMax} ein Clustering vorgenommen und die Güte der resultierenden Clusterings verglichen. Das Clustering erfolgt in Methode 2 agglomerativ hierarchisch nach dem Average-Linkage-Kriterium[43]. Als Maß für die Güte wird d^{MICD}, die **mittlere Intraclusterdistanz**, d. h. die mittleren Distanzen der ZKs eines Clusters gemittelt über alle Cluster, verwendet. Diese nimmt mit zunehmender Clusteranzahl tendenziell ab (siehe Abbildung 4.11). Bei einer bestimmten Clusteranzahl bildet

[43] Für eine Erläuterung dieses Verfahrens sei auf Miyamoto (2022, S. 19–24) verwiesen.

sich jedoch ein sog. Elbow, auch Knee genannt, aus, d. h. ein Punkt, ab dem die weitere Erhöhung der Clusteranzahl die Intraclusterdistanz nur noch geringfügig reduziert. Dieser kann für diskrete Funktion nach Satopaa et al. (2011) als Punkt mit größtem orthogonalem Abstand zu einer Diagonalen bestimmt werden (sog. **Kneedle-Verfahren**). In der vorliegenden Arbeit wird die Diagonale als Verbindung der Punkte (0, 1) sowie (n^{ClMax}, 0) definiert. Für das vorliegende Beispiel ergibt sich für Bezeichnung $Z2$ eine optimale Clusteranzahl von 2, indem die ZKs II und XI zu einem Cluster und VII zu einem anderen Cluster zusammengefasst werden. Die mittlere Intraclusterdistanz ist 0, d. h. eine Erhöhung der Clusteranzahl würde das Clustering nicht weiter verbessern.

	$Z1$	$Z2$	$Z3$	$Z4$
n^{PrKl}	1	2*	2	1
d^{MICD}	0,28	0	0,31	0,28

*Beispiel: Berechnung der erwarteten Clusteranzahl für $Z2$:

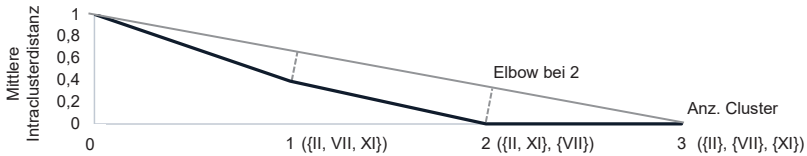


Abbildung 4.11: Ermittlung der prognostizierten Klassenanzahl mittels Kneedle-Verfahren im Rahmen von Alg^{MinMSTL} für den Beispielfall

Es ist somit im Rahmen der Unsicherheit des Distanzmaßes, wie oben beschrieben, davon auszugehen, dass in der MSTL tatsächlich zwei Klassen mit Bezeichnung $Z2$ vorliegen. Auf Basis der prognostizierten Klassenanzahl sowie der Bewertung d^{MICD} des optimalen Clusterings kann die Betrachtungsreihenfolge festgelegt werden.

4.2.1.1.4 Festlegung der Betrachtungsreihenfolge

Wie in Kapitel 4.2.1.1.1 erläutert, ergibt sich die Betrachtungsreihenfolge primär aus der **prognostizierten Klassenanzahl**. Sekundär wird die **Intraclusterdistanz** d^{MICD} herangezogen. Falls also zwei ZKs in der Anzahl prognostizierter Klassen ihrer Bezeichnung übereinstimmen, werden zunächst ZKs mit geringem d^{MICD} betrachtet, da deren Neigung zu einer höheren Klassenanzahl geringer ist. Liegt auch hier eine Übereinstimmung vor, werden ZKs die aus größeren VSTLs stammen vor solchen aus kleineren VSTLs betrachtet. Dafür wird jeder ZK ein Wert n^{STLGr} zugeordnet, der die **Anzahl von ZKs in derselben VSTL** angibt. Werden ZKs aus großen VSTLs falsch

zugeordnet, hat dies tendenziell einen großen Einfluss auf die Zulässigkeit der Zuordnung. Ein solcher Fehler kann deshalb schnell automatisch entdeckt werden (siehe Kapitel 4.2.1.2.3). Liegt auch für n^{STLGr} eine Übereinstimmung vor, wird willkürlich nach kleinerem Index entschieden. Für den Beispielfall ergibt sich somit die in Abbildung 4.12 gezeigte Betrachtungsreihenfolge. Auf Basis der Betrachtungsreihenfolge kann der Suchbaum in aufeinanderfolgenden Iterationen traversiert werden.

	V	IX	X	XIV	I	IV	VII	XI	II	VI	VIII	XII	XIII	III
Priorität 1: n^{PrKl}	1	1	1	1	1	1	2	2	2	2	2	2	2	2
Priorität 2: d^{MICD}	0,28	0,28	0,28	0,28	0,28	0,28	0	0	0	0,31	0,31	0,31	0,31	0,31
Priorität 3: n^{STLGr}	5	5	5	5	4	4	5	5	4	5	5	5	5	4

Abbildung 4.12: Festlegung der Betrachtungsreihenfolge der Zukaufkomponenten im Rahmen von Alg^{MinMSTL} für den Beispielfall

4.2.1.2 Iteration von Algorithmus Alg^{MinMSTL}

Eine Iteration entspricht einer Entscheidung im Suchbaum. Im Folgenden wird der Ablauf einer Iteration erläutert.

4.2.1.2.1 Ablauf einer Iteration

Die Baumsuche folgt der in der Initialisierung festgelegten Betrachtungsreihenfolge. Im Beispielfall wird also zuerst über die Annotation der ZK V entschieden. Im Folgenden wird eine Iteration von Alg^{MinMSTL} am Beispiel der Iteration 10 des Beispielfalls erläutert,

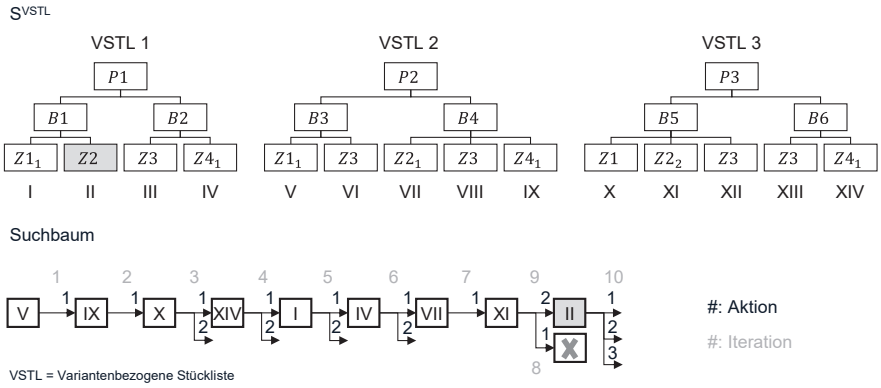


Abbildung 4.13: Vorgenommene Annotationen und Zustand des Entscheidungsbaums im Rahmen von Alg^{MinMSTL} für den Beispielfall zu Beginn von Iteration 10

da sich diese gut eignet, um das Vorgehen zu veranschaulichen. In Iteration 8 wurde die ZK XI mit 1 annotiert. Diese Entscheidung wurde anschließend verworfen und ZK XI in Iteration 9 mit 2 annotiert. Es ergibt sich der in Abbildung 4.13 gezeigte Zustand der VSTLs und der gezeigte Suchbaum als Ausgangslage für Iteration 10. In Iteration 10 wird entsprechend der Betrachtungsreihenfolge über die Annotation der ZK II mit Bezeichnung $\mathbb{Z}2$ entschieden.

4.2.1.2.2 Ermittlung der zulässigen Aktionen und Auswahl einer Aktion

Zunächst werden die **zulässige**n **Aktionen** im gegebenen Entscheidungsknoten bestimmt und bewertet. Eine Aktion entspricht dem Annotieren der betrachteten ZK mit einer bestimmten KN, d. h. dem Zuordnen eines bestimmten Labels. Wird eine KN gewählt, mit der bisher noch keine ZK derselben Bezeichnung annotiert wurde, entsteht ein neues Label und damit eine neue ZKK in der letztendlichen MSTL. Um zuerst Teilbäume zu durchsuchen, die eine geringe Komplexität der MSTL erwarten lassen, wird nur dann ein neues Label eingeführt, wenn der ZK kein bestehendes Label zugeordnet werden kann. Dabei sind **verbotene Aktionen** des Entscheidungsknotens zu berücksichtigen. Hierbei handelt es sich um Aktionen, die in diesem Entscheidungsknoten bereits zuvor ausgeführt und verworfen wurden. Außerdem ist es grundsätzlich unzulässig, eine ZK mit einer KN zu annotieren, mit der bereits eine ZK mit derselben Bezeichnung in derselben VSTL annotiert wurde. Durch diese Regel wird berücksichtigt, dass zwei ZK derselben VSTL nicht aus derselben ZKK der MSTL instanziiert werden können.

Existieren mehrere Labels, die der ZK zugeordnet werden können, ist eine **optimale Aktion** zu bestimmen. Für jedes der Labels, existiert eine Gruppe von ZKs in den VSTLs aus S^{VSTL}, denen dieses Label bereits zugeordnet wurde. Für alle ZKs einer solchen Gruppe wird angenommen, dass sie aus derselben ZKK der MSTL instanziiert werden. Um also eine optimale Aktion zu ermitteln, ist zu untersuchen, zu welcher Gruppe von ZKs die betrachtete ZK die **geringste Distanz** aufweist. Hierfür wird das in Kapitel 4.2.1.1.2 erläuterte Distanzmaß verwendet und je möglichem Label, d. h. je Gruppe, die mittlere Distanz zu den zugehörigen ZKs berechnet. Im Beispielfall (siehe Abbildung 4.14) liegen für die Bezeichnung der betrachteten Komponente, Z2, bereits zwei verschiedene KNs vor (1 und 2). Außerdem existieren im Entscheidungsknoten keine verbotenen Aktionen. Damit darf kein neues Label eingeführt werden, sondern es muss entweder das Label Z2;1 oder das Label Z2;2 zugeordnet werden. Die

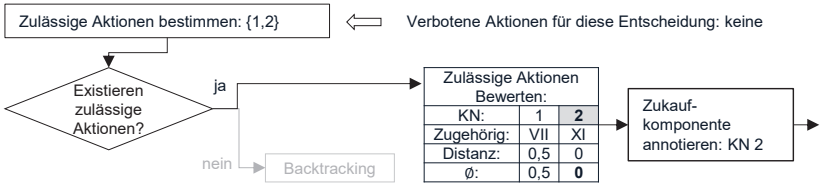


Abbildung 4.14: Vorgehen zur Ermittlung einer Aktion in einem Knoten des Suchbaums im Rahmen von Alg^{MinMSTL} für den Beispielfall

betrachtete ZK weist zu der ZK mit Label Z2;2 die geringste Distanz auf, woraus sich eine Annotation mit KN 2 als optimale Aktion ergibt.

Nachdem die optimale Aktion bestimmt ist, wird die ZK entsprechend annotiert. Existieren für einen Entscheidungsknoten keine zulässigen Aktionen, d. h. sowohl die Zuordnung von bestehenden Labels als auch die Einführung eines neuen Labels wurden bereits durchgeführt und verworfen, findet ein **Backtracking** statt. In diesem Fall wird die zuletzt gewählte Aktion rückgängig gemacht, der vorherige Entscheidungsknoten ausgewählt und die zuletzt gewählte Aktion als verbotene Aktion des Entscheidungsknotens gespeichert. Die bisher beschriebene Vorgehensweise ermöglicht prinzipiell das Ermitteln einer optimalen Lösung durch vollständige Traversierung des Suchbaums. Die Effizienz der Suche kann jedoch, wie im Folgenden erläutert wird, durch Pruning weiter erhöht werden.

4.2.1.2.3 Pruning

Es existieren in Alg^{MinMSTL} zwei Kriterien, auf Basis derer ein Entscheidungsknoten von der weiteren Betrachtung ausgeschlossen werden kann. Das erste basiert auf der **unteren Schranke** b^{Un} für die Komplexität der Lösungen, die aus dem Zustand im Entscheidungsknoten hervorgehen können. Die Anzahl der ZKKs je Bezeichnung l in der MSTL kann nicht geringer sein als die maximale Anzahl n_l^{MaxB} an Positionen von ZKs mit dieser Bezeichnung über alle VSTLs aus S^{VSTL} hinweg, da nicht mehrere ZKs einer VSTL aus derselben Klasse instanziiert werden können. Sie kann für die finale Lösung außerdem nicht geringer sein als die Anzahl n_l^{KNBez} der KNs zu dieser Bezeichnung l, die im aktuellen Zustand bereits vorliegen. Damit gilt für die untere Schranke in einem Entscheidungsknoten

$$b^{Un} = \sum_{l \in S^{Bez}} \max(n_l^{MaxB}, n_l^{KNBez}) \,, \qquad 4.4$$

wobei S^{Bez} der Menge aller Bezeichnungen von ZKs in VSTLs in S^{VSTL} entspricht. Falls b^{Un} größer als die Komplexität b^{Min} der besten bisher gefundenen Lösung ist, kann ausgehend vom aktuellen Knoten des Suchbaums keine Lösung mit einer geringeren Komplexität als die beste bisher gefundene Lösung erzielt werden. Der Knoten muss nicht weiter betrachtet werden und es wird ein Backtracking durchgeführt. Wird somit schnell eine gute Lösung gefunden, muss ein großer Teil des Suchbaums nicht betrachtet werden, was zur Effizienz des Algorithmus beiträgt. Abbildung 4.15 zeigt das Vorgehen für den Beispielfall wobei sich aus den vorherigen Iterationen bereits eine Lösung mit Komplexität 11 – d. h. eine MSTL mit 11 ZKKs – ergeben hat.

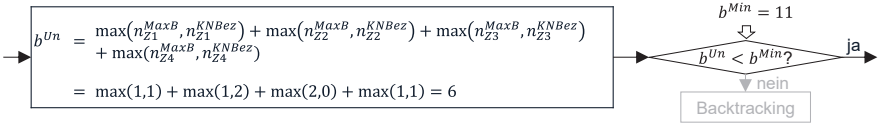


Abbildung 4.15: Pruning auf Basis einer unteren Schranke im Rahmen von Alg$^{\text{MinMSTL}}$ für den Beispielfall

Das zweite für das Pruning genutzte Kriterium ist die **Zulässigkeit** des Zustands im Entscheidungsknoten. Damit eine Lösung relevant ist, muss sie zulässig sein. Im Suchbaum können Knoten auftreten, aus denen sich durch weitere Annotationen von ZKs keine zulässigen Lösungen mehr ergeben können. In diesen Fällen müssen die untergeordneten Knoten im Suchbaum nicht betrachtet werden. Um diese Fälle zu ermitteln, wird Alg$^{\text{MSTL}}$ auf die VSTLs in S^{VSTL} angewandt, wobei die VSTLs auf die bereits annotierten ZKs reduziert werden. Eine solche Reduktion erfolgt, indem alle noch nicht

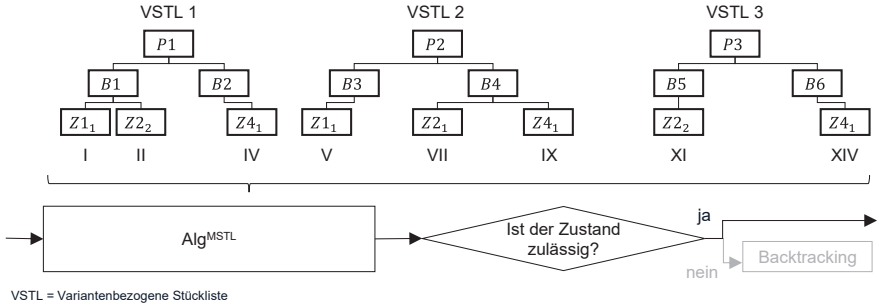


Abbildung 4.16: Zulässigkeitsprüfung im Rahmen von Alg$^{\text{MinMSTL}}$ für den Beispielfall

annotierten ZKs aus den VSTLs temporär entfernt werden und anschließend alle Baugruppen ohne untergeordnete ZKs ebenfalls temporär entfernt werden (siehe Abbildung 4.16). Alle annotierten Komponenten werden durch ihre Labels, d. h. ihre Bezeichnungen und KNs, identifiziert. Wie in Kapitel 4.2.1.2.2 beschrieben, können im Rahmen von $Alg^{MinMSTL}$ keine zwei identisch bezeichneten Komponenten einer VSTL mit derselben KN annotiert werden. Damit liegen für die reduzierten VSTLs keine Multikomponenten vor und Alg^{MSTL} ist anwendbar. Durch Anwendung von Alg^{MSTL} kann ermittelt werden, ob die reduzierten VSTLs STAs enthalten. Ggf. kann unterhalb des betrachteten Knotens keine zulässige Lösung existieren, weil STAs durch das Hinzufügen weiterer Annotationen nicht aufgelöst werden können. In diesem Fall erfolgt ein Backtracking.

4.2.1.2.4 Abschluss einer Iteration

Zum Abschluss einer Iteration wird überprüft, ob die in der Iteration erfolgte Annotation eine neue beste Lösung impliziert. Die höchste Komplexität der Lösung, die ausgehend von einem Knoten des Suchbaums noch erreicht werden kann, die obere Schranke b^{Ob}, wird realisiert, wenn alle noch nicht annotierten ZKs jeweils mit unterschiedlichen KNs annotiert werden. Entsprechend kann aus jedem Zustand trivial eine Lösung generiert werden. Sei n^{NiAn} die Anzahl noch nicht annotierter ZKs, dann gilt

$$b^{Ob} = n^{NiAn} + \sum_{l \in S^{Bez}} n_l^{KNBez} \quad . \tag{4.5}$$

Ist b^{Ob} kleiner als die Komplexität b^{Min} der besten bisher gefundenen Lösung wird der Wert von b^{Min} durch den Wert von b^{Ob} ersetzt und die aus dem Knoten abgeleitete Lösung als **neue beste Lösung** gespeichert. Wurden im betrachteten Knoten bereits alle ZKs zugeordnet, d. h. ist der Knoten ein Blatt des Suchbaums, kann die Suche von diesem Knoten aus nicht fortgesetzt werden und es erfolgt ein Backtracking. Abbildung 4.17 zeigt das Vorgehen für den Beispielfall.

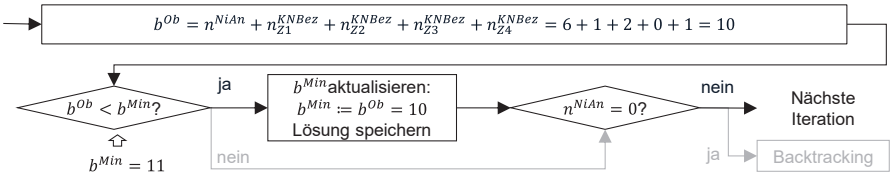


Abbildung 4.17: Abschluss einer Iteration von $Alg^{MinMSTL}$ für den Beispielfall

4.2.1.2.5 Terminierung von Algorithmus Alg^{MinMSTL}

Der Algorithmus Alg^{MinMSTL} terminiert, sobald der Suchbaum, exklusive der ausgeschlossenen Teilbäume, vollständig durchsucht wurde. Dies ist genau dann der Fall, wenn ein Backtracking zur Wurzel des Suchbaums erfolgt ist und hier keine zulässigen Aktionen mehr existieren. Da außerdem Alg^{MinMSTL} in der Lage ist zu jedem Zeitpunkt die beste bisher gefundene Lösung zurückzugeben, können weitere übliche **Abbruchkriterien** für Baumsuchen verwendet werden, wie z. B. eine bestimmte Laufzeit, Anzahl Iterationen, Zeit ohne Verbesserung von b^{Min} oder Anzahl Iterationen ohne Verbesserung von b^{Min}. Nach Abschluss von Alg^{MinMSTL} liegt eine optimale Annotation der ZKs vor. Mittels Alg^{MSTL} kann somit eine MSTL mit geringstmöglicher Anzahl von ZKKs erstellt werden. Für den Beispielfall terminiert Alg^{MinMSTL} nach 37 Iterationen und es ergibt sich die in Abbildung 4.18 gezeigte MSTL wobei die Subskripte der Klassen Multipositionen abbilden. Diese MSTL gibt noch keinen Aufschluss darüber, ob die zugrundeliegenden VSTLs STAs enthalten und ggf. welche.

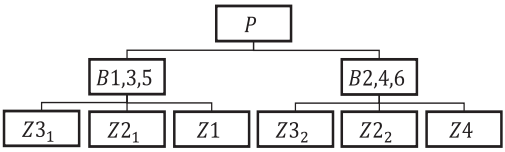


Abbildung 4.18: Resultierende Maximalstückliste nach Schritt 1 der Methode 2 für den Beispielfall

4.2.2 Schritt 2: Strukturoptionen bestimmen

In Schritt 2 werden die STAs der VSTLs aus S^{VSTL} ermittelt und in der MSTL als STOs abgebildet.

4.2.2.1 Problematik der Ermittlung von Strukturoptionen in einer Maximalstückliste

Bei den Klassen $Z2_1$ und $Z2_2$ in der beispielhaften MSTL (siehe Abbildung 4.18) kann es sich sowohl um STAs als auch um Multipositionen in der MSTL handeln. Entweder kann also ZK $Z2$ an verschiedenen Positionen in einer VSTL auftreten oder eine VSTL kann bis zu zwei ZKs $Z2$ enthalten. Welcher Fall vorliegt, kann aus den Daten nicht mit Sicherheit geschlossen werden. Deshalb wird folgende Annahme getroffen, die im Folgenden als **Maximalitätsannahme** bezeichnet wird: Eine ZK mit einer bestimmten Bezeichnung kann in einer VSTL, die aus der MSTL konfiguriert werden kann, nicht öfter

auftreten, als in einer der gegebenen VSTLs[44]. Je größer der vorhandene Datensatz ist, desto größer ist die Wahrscheinlichkeit, dass darin eine VSTL existiert, die die maximal mögliche Anzahl von Positionen für eine Bezeichnung enthält. Damit ermöglicht die Maximalitätsannahme, dass das vorhergesagte Modell bei zunehmender Datenmenge gegen das tatsächliche Modell konvergiert. Aufgrund der Maximalitätsannahme ist es bei einer kleinen Datenmenge möglich, dass dem Nutzer zu prüfende STOs angezeigt werden, die sich mit zusätzlichen Daten auflösen würden. Diese STOs sind ggf. vom Nutzer zu verwerfen.

Unter der Maximalitätsannahme lässt sich eindeutig entscheiden, ob STAs vorliegen, jedoch nicht, wie viele STAs vorliegen und welche VSTLs eine gemeinsame Struktur aufweisen. Alg^{MSTL} führt z. B. für die VSTLs α, β und γ in Abbildung 4.19 zu keiner Lösung, d. h. die VSTLs enthalten STAs. Wird VSTL α entfernt, ergibt sich hingegen eine Lösung. Daraus könnte geschlossen werden, dass VSTL β und VSTL γ einer STA entsprechen und VSTL α einer anderen. Derselbe Effekt tritt jedoch auf, wenn VSTL β oder VSTL γ entfernt werden. Welche VSTLs eine gemeinsame Struktur aufweisen und somit einer STA entsprechen, ist somit nicht eindeutig. Auch der Schluss, dass jede der VSTLs einer eigenen STA entspricht und damit also drei verschiedene Montagereihenfolgen für das Produkt in den Daten existieren, ist zunächst zulässig. Um systematisch über vorliegende STAs entscheiden zu können, wird die folgende Annahme getroffen, die im Folgenden als **Minimalitätsannahme** bezeichnet wird: Es liegen in den gegebenen VSTLs nicht mehr STAs vor, als notwendig sind, um den Datensatz vollständig zu erklären. Diese Annahme folgt dem Prinzip eines Modells geringer Komplexität. Für den Fall in Abbildung 4.19 ist somit jeder genannte Schluss zulässig, der von zwei STAs ausgeht; z. B. könnte die Struktur von VSTL α und VSTL β als eine STA angesehen werden und die von VSTL γ als eine andere. Je STA in den betrachteten VSTLs liegt

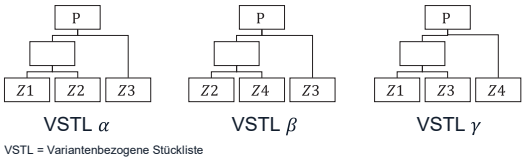


Abbildung 4.19: Beispielhafte variantenbezogene Stücklisten mit Strukturalternativen

[44] Eine alternative plausible Annahme wäre, dass grundsätzlich in den VSTL keine STAs vorliegen und Multipositionen in der MSTL immer mit mehrfach auftretenden Komponenten im Produkt einhergehen, auch wenn dieser Fall in den VSTL nicht vorliegt. Das würde allerdings dazu führen, dass relevante Hinweise auf Fehler unberücksichtigt bleiben. Aufgrund der in Kapitel 1.1 beschriebenen Auswirkungen von Fehlern in KM ist die gewählte Annahme somit geeigneter.

eine STO in der datenbasiert erstellten MSTL vor. Im Sinne der Minimalitätsannahme ist also eine MSTL mit einer minimalen Anzahl von STOs zu ermitteln, aus der sich alle VSTLs aus S^{VSTL} konfigurieren lassen.

4.2.2.2 Optimierungsproblem zur Ermittlung von Strukturoptionen in einer Maximalstückliste

Die STOs, die in der zu erstellenden MSTL abschließend vorliegen, sind ex ante nicht bekannt. Deshalb wird für das im Folgenden hergeleitete Optimierungsproblem eine ausreichend große Anzahl n^{Pl} an **STO-Platzhalter**n eingeführt. STO-Platzhalter können aktiv oder inaktiv sein. Jeder STO-Platzhalter, der im Zuge der Optimierung aktiviert wird, ergibt abschließend eine STO in der MSTL. Nach Minimalitätsannahme sind also so wenige STO-Platzhalter zu aktivieren wie nötig, d. h. **primäres Optimierungskriterium** ist die Anzahl aktiver STO-Platzhalter.

Jede ZKK der MSTL aus Schritt 1 muss abschließend denjenigen STOs zugeordnet sein, für die sie instanziiert werden kann (siehe Kapitel 4.1.1). Es muss sichergestellt werden, dass es für jede VSTL aus S^{VSTL} mindestens eine STO in der MSTL gibt, die alle ZKKs enthält, die notwendig sind um alle ZKs dieser VSTL zu instanziieren. Das im Folgenden hergeleitete Optimierungsproblem hat deshalb **zwei Aspekte**. Einerseits werden die **ZKKs** der MSTL den STO-Platzhaltern zugeordnet, andererseits werden die **VSTLs** aus S^{VSTL} den STO-Platzhaltern zugeordnet. Ein STO-Platzhalter ist aktiv, wenn ihm mindestens eine ZKK oder eine VSTL zugeordnet ist. Die beiden oben genannten Aspekte lassen sich nicht unabhängig voneinander betrachten: Falls eine bestimmte VSTL einer bestimmten STO zugeordnet ist, müssen auch alle ZKKs, die notwendig sind, um diese VSTL zu konfigurieren, dieser STO zugeordnet sein.

Die Struktur aller VSTLs, die im Zuge der Optimierung demselben STO-Platzhalter zugeordnet werden, kann als eine STA betrachtet werden. Es kann vorkommen, dass keine der VSTLs, die gemeinsam eine STA bilden ZKs mit einer bestimmten Bezeichnung enthalten. Damit könnte unter den oben genannten Bedingungen nicht ausgeschlossen werden, dass nach Abschluss der Optimierung STO-Platzhalter existieren, denen keine ZKKs mit entsprechender Bezeichnung zugeordnet sind. Damit wäre es möglich, dass abschließend STOs in der MSTL existieren, die nicht zulassen, dass Komponenten mit dieser Bezeichnung überhaupt instanziiert werden können. Dadurch würden STOs einen Zusammenhang zwischen der Struktur konfigurierbarer VSTLs und den darin vorkommenden Typen von Komponenten herstellen. Dies würde die Menge

an konfigurierbaren VSTLs unbegründet einschränken und würde auch nicht dem eigentlichen Zweck von STOs entsprechen. Um dies zu vermeiden, wird im Folgenden das **sekundäre Optimierungskriterium** verwendet, dass so viele ZKKs wie möglich einem STO-Platzhalter zugeordnet werden sollen. Damit wird außerdem sichergestellt, dass abschließend einer STO der MSTL mindestens eine ZKK je Bezeichnung zugeordnet ist.

Das Optimierungsproblem lässt sich wie folgt mit den Variablen aus Tabelle 4.1 beschreiben. Sein prinzipieller Aufbau entspricht einem Graph-Coloring-Problem[45], bei dem Knoten eines Graphen unter gewissen Restriktionen Farben zugeordnet werden.

Tabelle 4.1: Variablen und Parameter des Optimierungsproblems zur Ermittlung von Strukturoptionen in einer Maximalstückliste

I^{Bez}	Menge der Indizes von Bezeichnungen in S^{Bez}
I_l^{KBez}	Menge der Indizes von Klassen der MSTL, die die Bezeichnung l tragen; aus Schritt 1 bekannt
I^{Pl}	Menge der Indizes von STO-Platzhaltern; es gilt $I^{Pl} = \{1, \dots, n^{Pl}\}$
I^{VSTL}	Menge der Indizes von VSTLs aus S^{VSTL}
I^{ZKK}	Menge der Indizes von ZKKs in der MSTL
$I_k^{ZKK,VSTL}$	Menge der Indizes von ZKKs, die für die Konfiguration der VSTL k aus der MSTL benötigt werden; aus Schritt 1 bekannt
n_l^{MaxB}	Parameter, der angibt, wie oft eine Bezeichnung mit Index l maximal in einer VSTL aus S^{VSTL} auftritt
$u_j^{PlAk} \in \{0,1\}$	Variable, die angibt, ob STO-Platzhalter j aktiv ist (1) oder nicht (0)
$u_{k,j}^{VSTL,Pl} \in \{0,1\}$	Variable, die angibt, ob VSTL k dem STO-Platzhalter j zugeordnet ist (1) oder nicht (0)
$u_{i,j}^{ZKK,Pl} \in \{0,1\}$	Variable, die angibt, ob die ZKK i der MSTL dem STO-Platzhalter j zugeordnet ist (1) oder nicht (0)

Optimierungsproblem zur Ermittlung von Strukturoptionen 4.6

$$min \qquad (|I^{ZKK}| * |I^{Pl}| + 1) \sum_{j \in I^{Pl}} u_j^{PlAk} - \sum_{i \in I^{ZKK}} \sum_{j \in I^{Pl}} u_{i,j}^{ZKK,Pl}$$

$$s.t.: \qquad |I^{ZKK}| * u_j^{PlAk} \geq \sum_{i \in I^{ZKK}} u_{i,j}^{ZKK,Pl} \qquad \forall j \in I^{Pl} \qquad (1)$$

[45] Für Graph-Coloring-Probleme sei auf Méndez-Díaz & Zabala (2006, S. 826–827) verwiesen.

$$u_{j-1}^{PlAk} \geq u_j^{PlAk} \qquad \forall j \in I^{Pl}/\{1\} \qquad (2)$$

$$n_l^{MaxB} \geq \sum_{i \in I_l^{KBez}} u_{i,j}^{ZKK,Pl} \qquad \forall j \in I^{Pl}, \forall\, l \in I^{Bez} \qquad (3)$$

$$\sum_{j \in I^{Pl}} u_{k,j}^{VSTL,Pl} \geq 1 \qquad \forall k \in I^{VSTL} \qquad (4)$$

$$\sum_{i \in I_k^{ZKK,VSTL}} u_{i,j}^{ZKK,Pl} \geq |I_k^{ZKK,VSTL}| * u_{k,j}^{VSTL,Pl} \qquad \forall k \in I^{VSTL}, \forall j \in I^{Pl} \qquad (5)$$

Da stets

$$\sum_{i \in I^{ZKK}} \sum_{j \in I^{Pl}} u_{i,j}^{ZKK,Pl} < |I^{ZKK}| * |I^{Pl}| + 1 \qquad 4.7$$

gilt wird die Minimierung der Anzahl aktiver STO-Platzhalter als primäres Ziel und die Zuordnung von ZKKs zu STOs als sekundäres Ziel behandelt. **Nebenbedingung 1** stellt sicher, dass ein STO-Platzhalter als aktiv gilt und im Zielfunktionswert berücksichtigt wird, sobald ihm eine ZKK zugeordnet wurde[46]. **Nebenbedingung 2** besagt, dass ein STO-Platzhalter nur aktiv sein kann, wenn der STO-Platzhalter mit um 1 geringerem Index aktiv ist. Damit wird die Reihenfolge festgelegt, in der die Platzhalter aktiviert werden, wodurch der Lösungsraum eingeschränkt wird. Derartige Nebenbedingungen zum Brechen der Symmetrie des Lösungsraums werden auch für Graph-Coloring-Probleme eingesetzt (Méndez-Díaz & Zabala 2006, S. 828). **Nebenbedingung 3** setzt die Maximalitätsannahme um. **Nebenbedingung 4** stellt sicher, dass jede VSTL mindestens einem STO-Platzhalter und damit mindestens einer STO der letztlichen MSTL zugeordnet ist. Andernfalls könnten MSTLs entstehen, aus denen bestimmte VSTLs aus S^{VSTL} für keine der möglichen STOs abgeleitet werden können. **Nebenbedingung 5** stellt sicher, dass die Variable $u_{k,j}^{VSTL,Pl}$ nur dann auf 1 gesetzt werden darf, wenn tatsächlich alle für k benötigten ZKKs dem STO-Platzhalter j zugeordnet sind. Nebenbedingung 5 komplementiert somit Nebenbedingung 4. Aus der optimalen Lösung des aufgestellten Optimierungsproblems ergibt sich zum einen, wie viele STOs in der MSTL vorliegen und zum anderen, welche VSTLs und welche ZKKs jeweils welcher STO zugeordnet sind.

n^{Pl} kann mit $n^{Pl} = |S^{VSTL}|$ trivial gewählt werden. Anhang A3.4 erläutert, wie heuristisch ein kleineres n^{Pl} ermittelt werden kann, das dennoch die Lösbarkeit des

[46] Damit ist ein STO-Platzhalter insbesondere aktiv, wenn ihm eine VSTL und damit alle ZKKs dieser VSTL zugeordnet sind, so dass auf eine entsprechende Nebenbedingung für die Zuordnung von VSTLs verzichtet werden kann.

Optimierungsproblems 4.6 sicherstellt. Dadurch kann das Optimierungsproblem effizienter gelöst werden.

4.2.2.3 Abschluss von Schritt 2

Existiert nach der Lösung des Optimierungsproblems mehr als ein aktiver STO-Platzhalter liegen in der MSTL STOs vor. Welche ZKKs für welche STOs instanziiert werden können, ist durch $u_{k,j}^{VSTL,Pl}$ in der **optimale**n **Lösung** gegeben. Klassen, die jeder Gruppe zugeordnet sind, d. h. für jede STO instanziiert werden können, sind unabhängig von der geltenden STO. Für die entsprechenden ZKs können keine alternativen Positionen in der Fügereihenfolge existieren. Für den Beispielfall ergibt sich nach Lösung des Optimierungsproblems die in Abbildung 4.20 gezeigte MSTL mit zwei STOs. Die STOs, denen die ZKKs jeweils zugeordnet sind, sind mit Superskript notiert. Bei ZKKs, die unabhängig von der STO sind wird auf das Superskript verzichtet. In STO 1 wird $Z2$ zunächst mit $Z1$ und $Z3$ gefügt und die daraus resultierende Baugruppe mit einer Baugruppe, die aus $Z3$ und $Z4$ besteht. In STO 2 wird $Z2$ zunächst mit $Z3$ und $Z4$ gefügt und anschließend mit einer Baugruppe, die aus $Z1$ und $Z3$ besteht. Nach Abschluss von Schritt 2 liegt eine MSTL vor, deren ZKKs STOs zugeordnet sind. Weitere Parameter der KKs sind noch offen. Auf Basis der Ergebnisse aus Schritt 2 kann geprüft werden, ob die ermittelten STOs Hinweise auf Inkonsistenzen in den Daten darstellen.

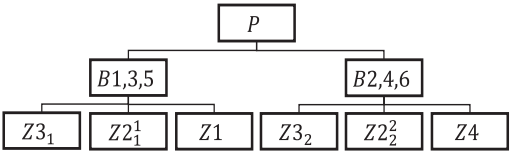


Abbildung 4.20: Resultierende Maximalstückliste nach Schritt 2 von Methode 2 für den Beispielfall

4.2.3 Schritt 3: Strukturoptionen prüfen

Das Überprüfen der STOs ist optional und der einzige manuelle Schritt der Methode. In der MSTL sind nach Schritt 2 STOs erkennbar, welche den STAs in den VSTLs aus S^{VSTL} entsprechen. Ein Domänenexperte kann nun die STOs und zugehörigen STAs daraufhin überprüfen, ob sie Inkonsistenzen in den Daten oder begründete **alternative Montagereihenfolgen** darstellen. Ist ersteres der Fall, können die Ursache für die **Inkonsistenz** ermittelt und unzutreffende STOs aus der MSTL gelöscht werden. Ist letzteres der Fall, kann der Experte entscheiden, ob die jeweils gültige STO von der gewählten Variante abhängt oder von sonstigen Einflussgrößen. Hängt sie von der

gewählten Variante, d. h. den Ausprägungen der Produktmerkmale ab, kann die Abhängigkeit der gültigen STO (siehe Kapitel 4.1.1, Parameter $p^{STO,MSTL}$) von den Produktmerkmalen mittels Methode 4 (siehe Kapitel 4.4) datenbasiert ermittelt werden. Hängt die gültige STO von anderen Einflussgrößen, wie z. B. dem ausführenden Werk ab, ist dies im KM zu hinterlegen. Wird keine Prüfung durchgeführt, wird davon ausgegangen, dass alle STOs gültig und von der gewählten Variante abhängig sind.

4.2.4 Schritt 4: Parameter der Komponentenklassen definieren

Wie oben beschrieben, sind für die Produktklasse und die Komponentenklassen der MSTL, abgesehen von ihrer STO-Zuordnung, noch keine Parameter definiert. Dies geschieht in Schritt 4. Aus Schritt 1 ist bekannt, aus welchen Komponenten eine Klasse der MSTL jeweils hervorgegangen ist. Die **Parameter** der Klassen ergeben sich aus den Parametern ihrer originären Komponenten. Die Bezeichnungen von ZKKs ergeben sich aus den Bezeichnungen ihrer originären Komponenten, ihre Positionen aus ihren KNs und ihre zugehörigen STOs aus Schritt 2. Der Aktivitätszustand aller KKs ist vom Typ Boolean. Die möglichen STOs des zugehörigen MAPL bleiben bis zur Durchführung von Methode 3 offen. Die **Definitionsbereiche** aller weiteren Parameter ergeben sich durch Vereinigung der Definitionsbereiche der ursprünglichen Komponenten, wie

Komponente I
Bezeichnung (p_I^{Bez}) = „B1“
Position (p_I^{Pos}) = N/A
Aktiv (p_I^{AkZu}) = True
Menge (p_I^{Me}) = v_I^{Me}
Zugehörige STOs (p_I^{STO}) = {1,2}
STO des MAPL ($p_I^{STOMAPL}$) = tbd
Parameter $p_{I,1}^{Pa}$ = $v_{I,1}^{Pa}$
…

Komponente V
Bezeichnung (p_V^{Bez}) = „B1“
Position (p_V^{Pos}) = N/A
Aktiv (p_V^{AkZu}) = True
Menge (p_V^{Me}) = v_V^{Me}
Zugehörige STOs (p_V^{STO}) = {1,2}
STO des MAPL ($p_V^{STOMAPL}$) = tbd
Parameter $p_{V,1}^{Pa}$ = $v_{V,1}^{Pa}$
…

Komponente X
Bezeichnung (p_X^{Bez}) = „B1“
Position (p_X^{Pos}) = N/A
Aktiv (p_X^{AkZu}) = True
Menge (p_X^{Me}) = v_X^{Me}
Zugehörige STOs (p_X^{STO}) = {1,2}
STO des MAPL ($p_X^{STOMAPL}$) = tbd
Parameter $p_{X,1}^{Pa}$ = $v_{X,1}^{Pa}$
…

Komponentenklasse KK_1^Z	
Bezeichnung ($p_{KK_1^Z}^{Bez}$) = „B1“	Zugehörige STOs ($p_{KK_1^Z}^{STO}$) = {1,2}
Position ($p_{KK_1^Z}^{Pos}$) = N/A	STO des MAPL ($p_{KK_1^Z}^{STOMAPL}$): tbd
Aktiv ($p_{KK_1^Z}^{AkZu}$): bool	Parameter$p_{KK_1^Z,1}^{Pa}$: $\{v_{I,1}^{Pa}, v_{V,1}^{Pa}, v_{X,1}^{Pa}\}$
Menge ($p_{KK_1^Z}^{Me}$): $\{v_I^{Me}, v_V^{Me}, v_X^{Me}\}$	…

MAPL = Maximalarbeitsplan, STO = Strukturoption

Abbildung 4.21: Bildung einer Komponentenklasse für den Beispielfall

für den Beispielfall in Abbildung 4.21 zu sehen. Die nach Schritt 4 vorliegende MSTL ist vollständig und kann für die Produktkonfiguration, wie in Kapitel 4.1.1 beschrieben, eingesetzt werden. Damit liegt der erste Baustein des zu erstellenden LLKM vor. Um jedoch die Verständlichkeit der MSTL zu erhöhen, können ZKKs mit ähnlicher Funktion in Superklassen aggregiert, d. h. generalisiert, werden.

4.2.5 Schritt 5: Zukaufkomponentenklassen generalisieren

Zwei ZKKs sind **Kandidaten für eine Generalisierung**, wenn sie an derselben Position in der MSTL – d. h. als meronymisch untergeordnete Elemente derselben Klasse – auftreten, wenn sie denselben STOs zugeordnet sind und wenn keine VSTL in S^{VSTL} existiert, in der beide ZKKs zugleich auftreten. Mit den von Romanowski & Nagi (2004) entwickelten **Ähnlichkeitsmaße**n (siehe Kapitel 3.2.2) kann ermittelt werden, ob es sinnvoll ist, ZKKs, die Kandidaten für eine Generalisierung sind, tatsächlich zu generalisieren. Ggf. wird eine Superklasse in die MSTL eingefügt und die entsprechenden ZKKs dieser untergeordnet. Im Gegensatz zum Ansatz von Romanowski & Nagi (2004) findet die Generalisierung damit statt nachdem die STAs in den VSTLs aus S^{VSTL} in der MSTL als STOs erfasst wurden, sodass keine STAs in den VSTLs durch die Generalisierung entstehen können.

4.3 Methode 3: Datenbasierte Erstellung von Maximalarbeitsplänen

Im Folgenden wird die im Rahmen der vorliegenden Arbeit entwickelte Methode 3 zur Lösung des Problems 3 – der datenbasierten Erstellung von MAPLs – vorgestellt. Methode 3 behebt das in Kapitel 3.3.3 beschriebene Lösungsdefizit nach Stand der Forschung. Ausgehend von einer Menge S^{VAPL} von VAPLs (siehe Kapitel 4.1.3, Abbildung 4.7) wird ein MAPL erstellt. Es wird davon ausgegangen, dass jeder AVO in VAPLs aus S^{VAPL} mit einer Bezeichnung versehen ist, die seinen Typ eindeutig bestimmt, wie z. B. durch eine **genormte Bezeichnung** gemäß DIN-Norm 8580 (DIN 8580:2022-12). Ist dies nicht der Fall, kann vorab eine Generalisierung vorgenommen werden, indem AVOs mit ähnlichen Bezeichnungen und Parametern zusammengefasst werden. Die Ähnlichkeit kann dabei in Anlehnung an das von Romanowski & Nagi (2004, S. 322) entwickelte Ähnlichkeitsmaß für ZKs berechnet werden. STAs, die aus der Generalisierung herrühren, müssen bei der manuellen Prüfung korrigiert werden. AVOs desselben Typs können an mehreren Positionen in einem VAPL zugleich auftreten, d. h. die

VAPLs aus S^{VAPL} können Multivorgänge enthalten. VAPLs aus S^{VAPL} können darüber hinaus STAs aufweisen. Entsprechend kann der resultierende MAPL Multipositionen und ausgewiesene STOs, wie in Kapitel 4.1.1 beschrieben, enthalten.

In der vorliegenden Arbeit wird von VAPLs in Form von **Vorranggraphen** ausgegangen. Liegen VAPLs in Form von Ablaufdiagrammen vor, können diese unter der Annahme, dass alle indirekten Vorrangbeziehungen gelten, in Vorranggraphen überführt werden. Umgekehrt lassen sich Vorranggraphen durch transitive Reduktion[47] in Ablaufdiagramme überführen. Abbildung 4.22 zeigt die im Folgenden als Beispiel verwendeten VAPLs aus S^{VAPL} jeweils in Form von Ablaufdiagrammen (1) und in Form von Vorranggraphen (2). Die AVOs sind mit „A“ und einer fortlaufenden Nummer bezeichnet, wobei identisch bezeichnete AVOs auftreten können. Zur Referenz sind die AVOs römisch nummeriert[48]. Analog zu dem in Methode 2 eingesetzten Algorithmus Alg^{MSTL} (siehe Anhang A3.1) wurde im Rahmen der vorliegenden Arbeit ein Algorithmus **Alg^{MAPL}** entwickelt. Dieser ist in der Lage für eine Menge S^{VAPL} von VAPLs in Form von Vorranggraphen ohne Multivorgänge und ohne STAs einen MAPL zu erstellen, aus dem alle VAPLs aus S^{VAPL}, wie in Kapitel 4.1.1 beschrieben, konfiguriert werden können. Enthalten die VAPLs aus S^{VAPL} STAs gibt Alg^{MAPL} zurück, dass keine Lösung ermittelt werden kann. Alg^{MAPL} wird in Anhang A4.1 erläutert und als Pseudocode dargestellt.

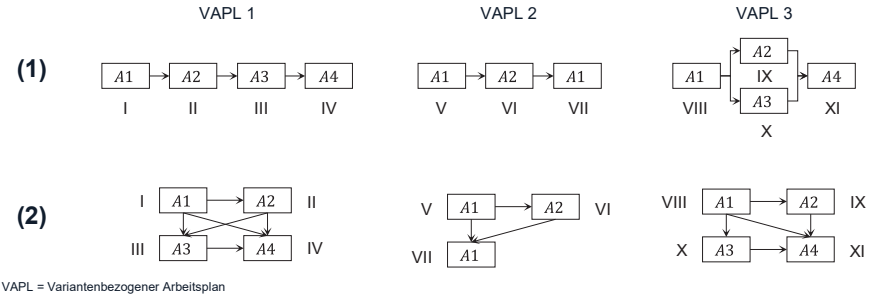


Abbildung 4.22: Beispielhafte variantenbezogene Arbeitspläne in Ablaufdiagramm- und Vorranggraphdarstellung

[47] Für transitive Reduktion sei auf Skiena (2020, S. 559–562) verwiesen
[48] Diese Nummerierung hat keinen Bezug zu der in Kapitel 4.2.1 verwendeten Nummerierung von Zukaufkomponenten.

Alg^{MAPL} kann eingesetzt werden, um einen MAPL aus VAPLs mit Multivorgängen und STAs zu erstellen. Dies erfolgt analog zu der in Kapitel 4.2.1 vorgestellten Erstellung von MSTLs aus VSTLs mit Multikomponenten und STAs durch Alg^{MSTL}. Die in Kapitel 4.2 dargestellten fünf **Schritte von Methode 2** zur Erstellung einer MSTL aus einer Menge S^{VSTL} von VSTLs können somit auf die Erstellung eines MAPL aus einer Menge S^{VAPL} von VAPLs übertragen werden. In den folgenden Kapiteln wird deshalb ausschließlich auf die notwendigen Anpassungen eingegangen. Abbildung 4.23 zeigt die fünf Schritte der Methode 3 im Überblick.

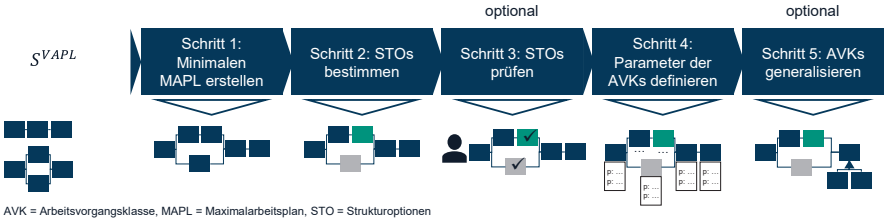


Abbildung 4.23: Überblick über die Schritte der Methode 3

4.3.1 Schritt 1: Minimalen Maximalarbeitsplan erstellen

Schritt 1 der Methode 3 nutzt einen Algorithmus **$Alg^{MinMAPL}$**, der weitgehend dem in Kapitel 4.2.1 vorgestellten Algorithmus $Alg^{MinMSTL}$ entspricht. Auf eine separate Darstellung als Pseudocode wird deshalb verzichtet. Die **Grundidee** ist ebenfalls, die AVOs der VAPLs aus S^{VAPL} mit Klassennummern (KNs) zu versehen um somit VAPLs ohne STAs und Multivorgänge zu erhalten. Die Verbindung einer **Bezeichnung** und einer **KN** wird auch für AVOs als **Label** bezeichnet (siehe Kapitel 4.2.1).

Zunächst findet eine Initialisierung statt, in der die **Betrachtungsreihenfolge** der AVOs in den VAPLs aus S^{VAPL} festgelegt wird. Dies erfolgt analog nach **prognostizierte**r **Klassenanzahl** je Bezeichnung, Intraclusterdistanz des Clusterings sowie der Anzahl von AVOs im zugehörigen VAPL. Für das Clustering wird analog zum **Distanzmaß** für ZKs in VSTLs ein Distanzmaß für AVOs in VAPLs benötigt. Hierfür wird das Maß $d_{i,j}^{AVO}$ für die Distanz zweier AVOs i und j eingeführt, das ebenfalls auf dem Konzept der Kontextähnlichkeit basiert. Für jeden der beiden AVOs werden jeweils die folgenden drei Mengen an Labels bestimmt: $\boldsymbol{S_i^{vor}}$ enthält die Labels der vorausgehenden AVOs im zugehörigen VAPL, $\boldsymbol{S_i^{Nach}}$ enthält die Labels der nachfolgenden AVOs im

zugehörigen VAPL und $\boldsymbol{S_i^{OB}}$ enthält die Labels von AVOs, die in keiner Beziehung zu dem betrachteten AVO stehen. Für AVO III in VAPL 1 im Beispielfall aus Abbildung 4.22 gilt z. B. $S_{III}^{Vor} = \{A1, A2\}$, $S_{III}^{Nach} = \{A4\}$ und $S_{III}^{OB} = \emptyset$. Für AVO X gilt $S_X^{Vor} = \{A1\}$, $S_X^{Nach} = \{A4\}$ und $S_X^{OB} = \{A2\}$. Für jede der drei Beziehungsarten wird die Kardinalität der Schnittmengen bestimmt. Für den Beispielfall gilt $|S_{III}^{Vor} \cap S_X^{Vor}| = |\{A1\}| = 1$, $|S_{III}^{Nach} \cap S_X^{Nach}| = |\{A4\}| = 1$ und $|S_{III}^{OB} \cap S_X^{OB}| = |\emptyset| = 0$. Die Übereinstimmung entspricht der Summe der Kardinalitäten, d. h. 2 im Beispielfall. Um diese zu normieren wird die maximal mögliche Übereinstimmung als Kardinalität der Schnittmenge $S_i^{V,L} \cap S_j^{V,L}$ bestimmt wobei $S_i^{V,L}$ die Menge aller Labels in dem zu AVO i gehörigen VAPL bezeichnet. Im Beispielfall gilt $|S_{III}^{V,L} \cap S_X^{V,L}| = |\{A1, A2, A3, A4\}| = 4$. Damit ergibt sich eine normierte Übereinstimmung $s_{III,X}^{AVO}$ von $\frac{2}{4} = 0{,}5$ und damit eine Distanz $d_{III,X}^{AVO}$ von $1 - 0{,}5 = 0{,}5$ für die beiden AVOs III und X. Allgemein gilt

$$s_{i,j}^{AVO} = \frac{|S_i^{Vor} \cap S_j^{Vor}| + |S_i^{Nach} \cap S_j^{Nach}| + |S_i^{OB} \cap S_j^{OB}|}{|S_i^{V,L} \cap S_j^{V,L}|} \quad , \tag{4.8}$$

falls $|S_i^{V,L} \cap S_j^{V,L}| > 0$ und ansonsten $s_{i,j}^{AVO} = 1$. Damit ergibt sich $d_{i,j}^{AVO}$ als

$$d_{i,j}^{AVO} = 1 - s_{i,j}^{AVO}. \tag{4.9}$$

Die **Iterationen** von Alg$^{\text{MinMAPL}}$ erfolgen wie in Kapitel 4.2.1.2 beschrieben. Dabei wird für die Überprüfung der Zulässigkeit anstelle von Alg$^{\text{MSTL}}$ der zuvor eingeführte Algorithmus Alg$^{\text{MAPL}}$ verwendet. Analog zur Überprüfung von teilweise annotierten VSTLs, wie in Kapitel 4.2.1.2.3 beschrieben, werden bei der Überprüfung der VAPLs in einem bestimmten Knoten des Suchbaums jeweils nur die bereits annotierten AVOs berücksichtigt. Analog zu der in Kapitel 4.2.1.2.3 beschriebenen Reduktion von VSTLs auf annotierte ZKs können VAPLs auf ihre annotierten AVOs reduziert werden. Hierbei werden alle nicht annotierten AVOs und alle zu diesen AVOs inzidenten Kanten entfernt. Alg$^{\text{MinMAPL}}$ nutzt dieselben Abbruchkriterien wie Alg$^{\text{MinMSTL}}$. Nach dem Abbruch lässt sich auf Basis der gegebenen VAPLs mit annotierten AVOs mittels Alg$^{\text{MAPL}}$ ein MAPL erstellen, aus dem alle VAPLs konfiguriert werden können. In diesem sind noch keine STOs hinterlegt, was im folgenden Schritt erfolgt. Für den Beispielfall ergibt sich

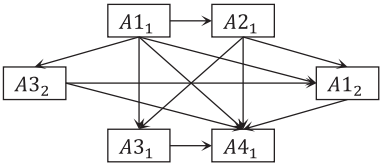


Abbildung 4.24: Maximalarbeitsplan nach Schritt 1 für den Beispielfall

der in Abbildung 4.24 dargestellte MAPL, wobei Indizes die Positionen der AVKs angeben.

4.3.2 Schritt 2: Strukturoptionen bestimmen

Die Ermittlung der STOs in dem zuvor erstellten MAPL erfolgt **analog zu Schritt 2 der Methode 2** (siehe Kapitel 4.2.2) durch mathematische Optimierung. Es wird das gleiche Optimierungsproblem aufgestellt, wobei AVOs und VAPLs STO-Platzhaltern zugeordnet werden, sodass ebenfalls nicht mehr AVOs mit derselben Bezeichnung einem Platzhalter zugeordnet werden, als diese Bezeichnung maximal in einem der VAPLs aus S^{VAPL} auftritt. Im Beispielfall ergeben sich die in Abbildung 4.25 (1) mit Superskripten dargestellten STO 1 und STO 2. Diese betreffen jeweils die mit $A3$ bezeichneten AVOs. Der MAPL in Vorranggraphdarstellung (1) kann mittels transitiver Reduktion in einen MAPL in Ablaufdiagrammdarstellung (2) umgewandelt werden. Die ermittelten STOs sind daraufhin zu überprüfen, ob sie Fehlerhinweise darstellen.

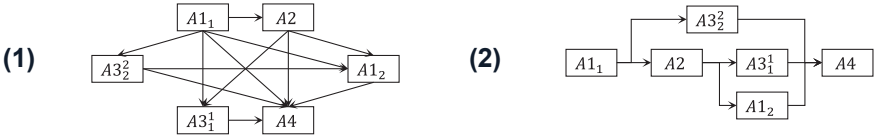


Abbildung 4.25: Resultierender Maximalarbeitsplan mit Strukturoptionen für den Beispielfall

4.3.3 Schritt 3: Strukturoptionen prüfen

Die Prüfung der STOs erfolgt **analog zu Schritt 3 der Methode 2** (siehe Kapitel 4.2.3). Es ist dabei zu berücksichtigen, dass STOs wie zuvor erläutert auch die Konsequenz einer unzulässigen Generalisierung sein können. Die Überführung des MAPL in eine Ablaufdiagrammdarstellung mittels transitiver Reduktion erleichtert die Überprüfung durch Domänenexperten.

4.3.4 Schritt 4: Parameter der Arbeitsvorgangsklassen definieren

Die Parameter einer AVK des MAPL ergeben sich **analog zu Schritt 4 der Methode 2** (siehe Kapitel 4.2.4) aus deren originären AVOs. Wie in Kapitel 4.2.4 beschrieben, ergibt sich die Gesamtheit der Parameter einer AVK als Komposition der Parameter ihrer AVOs. Deren Definitionsbereiche ergeben sich als Vereinigung der entsprechenden Ausprägungen in den AVOs. Der resultierende MAPL ist vollständig und kann für

einen Konfigurationsprozess verwendet werden. Um die Verständlichkeit des MAPL zu erhöhen, können auch für AVKs Generalisierungen vorgenommen werden.

4.3.5 Schritt 5: Generalisierung von Arbeitsvorgangsklassen

Zwei oder mehreren AVKs des MAPL kann prinzipiell eine Superklasse zugeordnet werden, wenn sie in keinem eingehenden VAPL **gemeinsam instanziiert** werden und wenn sie **an derselben Position** des MAPL auftreten, d. h. dieselben Mengen S_i^{Vor}, S_i^{Nach} und S_i^{OB} aufweisen. Bei dieser Betrachtung sind die Labels der beiden Vorgänge selbst zu vernachlässigen. Um zu entscheiden, welche AVKs letztlich zusammengefasst werden sollen, kann ebenfalls ein **Ähnlichkeitsmaß für AVKs** auf Basis ihrer Bezeichnung und Parameter verwendet werden. Dies ist jedoch nicht Gegenstand der vorliegenden Arbeit.

Nach Anwendung von Methode 3 liegt ein MAPL vor, aus dem alle VAPLs aus S^{VAPL} konfiguriert werden können. Im Rahmen von Methode 1 (siehe Kapitel 4.1.3) wird Methode 3 einmal für jede Klasse der MSTL angewandt, die dem Produkt oder einer eigengefertigten Komponente entspricht. Dabei werden jeweils alle VAPLs betrachtet, die zu Objekten gehören, die aus dieser Klasse instanziiert werden. Für jede dieser Klassen in der MSTL liegt somit abschließend ein MAPL vor. Um ein vollständiges LLKM zu erhalten, sind zuletzt die Regeln der Parameter der MSTL und der MAPLs zu ermitteln.

4.4 Methode 4: Datenbasierte Erstellung von Regeln

Im Folgenden wird die im Rahmen der vorliegenden Arbeit entwickelte Methode 4 zur Lösung des Problems 4 – der datenbasierten Erstellung von Regeln – vorgestellt. Methode 4 behebt das in Kapitel 3.4.3 beschriebene Lösungsdefizit nach Stand der Forschung.

Je nach abhängigem Parameter (siehe Kapitel 4.1.3) liegt bei der datenbasierten Erstellung von Regeln entweder eine Regression oder eine Klassifikation im Sinne des ML vor. Handelt es sich bei der zu erstellenden Regel um eine Auswahlbedingung, ist der abhängige Parameter binär. Er gibt an, ob die entsprechende Klasse der MSTL oder des MAPL für die gewählte Variante instanziiert wird oder nicht. Es liegt somit ein **binäres Klassifikationsproblem** vor. Hierauf liegt der Schwerpunkt der vorliegenden Arbeit (siehe Kapitel 2.2.2.4). Für andere Fälle von Regeln sei auf interpretierbare Modelle des ML in der Literatur verwiesen, wie z. B. von Rudin et al. 2022

zusammenfassend dargestellt. Im Folgenden wird von binären Features ausgegangen. Wie in Kapitel 3.4.2 beschrieben, stellt dies jedoch keine Einschränkung des Anwendungsfalls dar. Ggf. sind nichtbinäre Produktmerkmale oder Parameter des KM zunächst in ein oder mehrere **binäre Features** umzuwandeln. Ein Feature entspricht damit im Folgenden nicht zwingend genau einem Produktmerkmal oder einem Parameter des KM. Der Trainingsdatensatz ergibt sich wie in Kapitel 4.1.3 beschrieben. Dabei entspricht ein Datenpunkt einer Variante und es wird jeweils nur das Label betrachtet, das dem vorherzusagenden Parameter entspricht. Somit liegt ein Trainingsdatensatz $T^{Training}$ mit binären Features und genau einem **binären Label** vor. Auf Basis dessen ist ein **boolescher Ausdruck** in disjunktiver Normalform (DNF) mit **minimaler Komplexität**, d. h. minimaler Anzahl Literale, zu bestimmen, der eine **perfekte Trainingsgenauigkeit** aufweist. Dieser Ausdruck entspricht der Regel[49], durch die der abhängige Parameter, wie z. B. das Vorhandensein einer Komponente in der Maximalstückliste, im LLKM festgelegt wird.

Ein boolescher Ausdruck in DNF besteht aus konjunktiv verknüpften **Monome**n, wie z. B. der Ausdruck $(x_1 \wedge x_4) \vee (\overline{x_1} \wedge x_3 \wedge x_4)$ eine Konjunktion der Monome $(x_1 \wedge x_4)$ und $(\overline{x_1} \wedge x_3 \wedge x_4)$ darstellt. Damit der Ausdruck zulässig ist, muss er alle negativen Datenpunkte des Trainingsdatensatzes ablehnen, d. h. auf falsch abbilden, und alle positiven Datenpunkte akzeptieren, d. h. auf wahr abbilden. Diese Anforderung lässt sich auf die Monome des Ausdrucks herunterbrechen. Jedes der Monome muss insofern zulässig sein, als es alle negativen Datenpunkte ablehnt. Außerdem muss für jeden positiven Datenpunkt mindestens ein Monom existieren, das diesen akzeptiert. Für einen Datensatz können prinzipiell alle zulässigen Monome ermittelt werden. Somit kann prinzipiell das folgende **Master-Problem** (MP) mit den in Tabelle 4.2 eingeführten Variablen und Parametern aufgestellt werden.

Tabelle 4.2: Variablen und Parameter des Master-Problems

$a_{m,i} \in \{0,1\}$	Parameter, der angibt, ob Monom m den positiven Datenpunkt i im Trainingsdatensatz akzeptiert (1) oder nicht (0)
$c_m \in \mathbb{N}$	Anzahl der Literale in Monom m. Entspricht dem entsprechenden Zielfunktionskoeffizienten des MP
$n^{Dp} \in \mathbb{N}$	Anzahl der positiven Datenpunkte im Trainingsdatensatz

[49] Eine solche Regel ist eine Disjunktion von Monomen und entspricht damit einer Regelmenge im Sinne von Rudin et al. (2022).

$n^M \in \mathbb{N}$	Anzahl der berücksichtigten Monome in einem RMP
$u_m \in \{0,1\}$	Entscheidungsvariable des RMP, die festlegt, ob Monom m Teil des zu lernenden booleschen Ausdrucks ist (1) oder nicht (0)

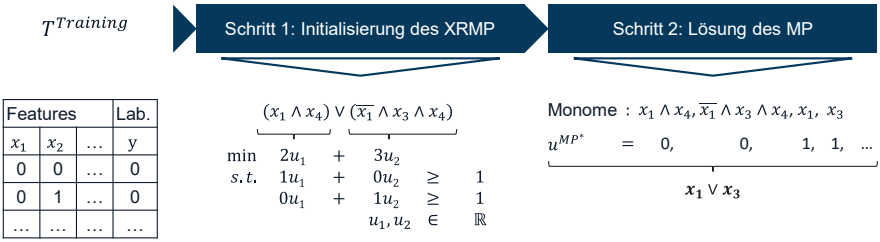
Master-Problem 4.10

$$
\begin{aligned}
min \quad & c_1 u_1 + \ \dots \ + c_{n^M} u_{n^M} & & & \\
s.t.: \quad & a_{1,1} u_1 + \ \dots \ + a_{1,n^M} u_{n^M} & \geq & 1 & \\
& \vdots \qquad \ddots \qquad \vdots & & \vdots & (1) \\
& a_{n^{Dp},1} u_1 + \ \dots \ + a_{n^{Dp},n^M} u_{n^M} & \geq & 1 & \\
& u_1, \dots, u_{n^M} & \geq & 0 & (2) \\
& u_1, \dots, u_{n^M} & \in & \mathbb{Z} & (3) \quad .
\end{aligned}
$$

Zu jedem MP existiert eine Liste L^{Monome}, die jeder Spalte das zugehörige Monom zuordnet. Die Zielfunktion entspricht der kumulierten Anzahl der Literale in allen ausgewählten Monomen. Die **Nebenbedingungen 1** stellen sicher, dass für jeden positiven Datenpunkt mindestens ein Monom gewählt wird, das diesen Datenpunkt akzeptiert. Somit wird jeder positive Datenpunkt des Trainingsdatensatzes durch den resultierenden booleschen Ausdruck akzeptiert. Da außerdem jedes der wählbaren Monome alle negativen Datenpunkte ablehnt, ist für jede zulässige Lösung des MP sichergestellt, dass der resultierende boolesche Ausdruck jeden negativen Datenpunkt ablehnt. Aus der Definition von u_m ergibt sich neben **Nebenbedingung 2** auch $u_1, \dots, u_{n^M} \leq 1$. Auf entsprechende Nebenbedingungen im MP kann jedoch verzichtet werden, da aufgrund der positiven Zielfunktionskoeffizienten keine optimale Lösung mit $u_m > 1$ für ein $t \in \{1, \dots, n^M\}$ existiert.

Die Anzahl der Variablen des MP entspricht der Anzahl zulässiger Monome für den Trainingsdatensatz und kann damit sehr groß werden. Außerdem besteht ein erheblicher **Rechenaufwand** darin, diese Monome vollständig zu bestimmen. Deshalb wird in der vorliegenden Methode 4 das MP weder explizit aufgestellt noch direkt gelöst, sondern das Prinzip der **Spaltengenerierung** (CG) angewandt (siehe Anhang A1.1). Abbildung 4.26 gibt einen Überblick über die Schritte der Methode. In Schritt 1 wird zunächst eine heuristische Lösung des MP ermittelt. Auf Basis dessen werden ein **reduziertes Master-Problem** (RMP) und das zugehörige **relaxierte reduzierte Master-Problem** (XRMP) aufgestellt. Dem initialen XRMP werden in Schritt 2 solange Spalten

hinzugefügt, bis sich eine Lösung des relaxierten Master-Problems (XMP) ergibt. Ist diese nichtganzzahlig und somit keine zulässige Lösung des MP, findet eine ggf. mehrfache, Verzweigung statt, sodass abschließend eine optimale Lösung des MP vorliegt. Die optimale Lösung des MP gibt an, welche Monome Teil des komplexitätsminimalen booleschen Ausdrucks sind.



MP = Master-Problem, XRMP = Relaxiertes reduziertes Master-Problem

Abbildung 4.26: Überblick über die Schritte der Methode 4

4.4.1 Schritt 1: Initialisierung des relaxierten reduzierten Master-Problems

Um ein lösbares RMP aufstellen zu können, müssen zulässige Monome bekannt sein, die zusammen genommen alle positiven Datenpunkte des Trainingsdatensatzes akzeptieren. Dies entspricht einem booleschen Ausdruck mit perfekter Trainingsgenauigkeit, welcher jedoch nicht zwingend komplexitätsminimal sein muss. Zur Ermittlung eines solchen booleschen Ausdrucks kann grundsätzlich jede **Heuristik** zur datenbasierten Erstellung von booleschen Ausdrücken eingesetzt werden, die eine perfekte Trainingsgenauigkeit gewährleistet. Wie in Kapitel 3.4.2 beschrieben, trifft dies auf mehrere der von Costamagna & Micheli (2023) vorgestellten Verfahren zu. Darüber hinaus gilt dies auch für Heuristiken zur Erstellung von Entscheidungsbäumen, die kein Pruning einsetzen. An dieser Stelle ist eine geringe Rechenzeit wichtiger als ein resultierender Ausdruck mit möglichst geringer Komplexität, da die Lösung lediglich als Ausgangsbasis für eine Optimierung dient. Außerdem kann selbst für gute Lösungen nicht zwingend davon ausgegangen werden, dass ihre Monome für die optimale Lösung relevant sind. Deshalb wird auf Heuristiken zur Erstellung von **Entscheidungsbäume**n zurückgegriffen, welche sehr recheneffizient sind (Osisanwo et al. 2017, S. 133–134). Gängige Verfahren wie z. B. C5.0 oder CART gehen von einem Wurzelknoten aus und fügen schrittweise untergeordnete Knoten hinzu, wobei jeder Knoten eine Partition des Datensatzes anhand eines Features darstellt (Patil et al. 2012, S. 2). Für die Auswahl des Features

existieren verschiedene sog. Splitting-Kriterien. Tangirala (2020) vergleicht die beiden gängigen Splitting-Kriterien Gini-Index und Informationsgewinn basierend auf Entropie miteinander. Der Vergleich ergibt, dass sich die jeweils resultierenden Entscheidungsbäume hinsichtlich ihrer Generalisierungsfähigkeit nicht signifikant unterscheiden. Für die vorliegende Arbeit wird der Gini-Index als Splitting-Kriterium verwendet[50]. Auf Pruning wird verzichtet, um eine initiale Lösung mit perfekter Trainingsgenauigkeit zu erhalten. Jeder Pfad des Baumes, der von der Wurzel zu einem Blatt führt, das der positiven Klasse zugeordnet ist, entspricht einem Monom des booleschen Ausdrucks, mit dem das RMP initialisiert wird. Aus den Datenpunkten die in seinen Blättern verbleiben ergibt sich, welche der positiven Datenpunkte das Monom akzeptiert. Damit kann ein zulässiger boolescher Ausdruck ermittelt und somit können die initialen Monome des RMP bestimmt werden.

Abbildung 4.27 (2) veranschaulicht die heuristische Initialisierung eines RMP anhand des beispielhaften Datensatzes, der in Abbildung 4.27 (1) dargestellt ist. Für die Erstellung des Entscheidungsbaums sind je Knoten die verbleibenden Datenpunkte sowie der Gini-Index der möglichen Splits dargestellt. Bei uneindeutigem minimalem Gini-Index wird der Split willkürlich in dem zuerst gelisteten Feature durchgeführt. Features, die keine Information über eine Klassenaufteilung enthalten, weil sie nur eine Ausprägung aufweisen, werden nicht berücksichtigt. Es ergeben sich zwei Monome, die als Basis für die Formulierung des initialen RMP dienen, was in Abbildung 4.27 (3) dargestellt ist. Je Monom enthält das RMP eine Spalte. Die Zielfunktionskoeffizienten des RMP entsprechen der Anzahl der Literale von Monom 1 bzw. Monom 2. Dessen Nebenbedingungskoeffizienten geben wieder, dass Monom 1 den positiven Datenpunkt 1 akzeptiert und Monom 2 den positiven Datenpunkt 2.

Das ermittelte RMP entspricht dem MP aus Formel 4.10, berücksichtig jedoch nur die Monome aus L^{Monome} und die entsprechenden Spalten der Koeffizientenmatrix. Auf eine explizite allgemeine Darstellung des RMP wird deshalb an dieser Stelle verzichtet. Aus dem initialen RMP wird ein initiales XRMP abgeleitet, im Folgenden als XRMP 1 bezeichnet, indem Nebenbedingung 3 in Optimierungsproblem 4.10 durch Nebenbedingung 3.1 ersetzt wird:

---

[50] Für die binäre Klassifikation mit binären Features ergibt sich der Gini-Index eines Features als $r_0 * (1 - r_{00}^2 - r_{01}^2) + r_1 * (1 - r_{10}^2 - r_{11}^2)$ wobei r_i den Anteil der Datenpunkte darstellt, für die das Feature den Wert i annimmt und r_{ij} den Anteil der Datenpunkte, für die das Feature den Wert i annimmt und deren Label j ist (hergeleitet aus Aggarwal 2021, S. 195).

$$u_1, \dots, u_{n^M} \in \mathbb{R} \ (3.1). \qquad 4.11$$

Das XRMP 1 dient als **Ausgangspunkt für die Lösung des MP**. Für den Beispielfall ergibt sich als XRMP 1 das in Abbildung 4.27 gezeigte Optimierungsproblem, jedoch mit $u_1, u_2 \in \mathbb{R}$.

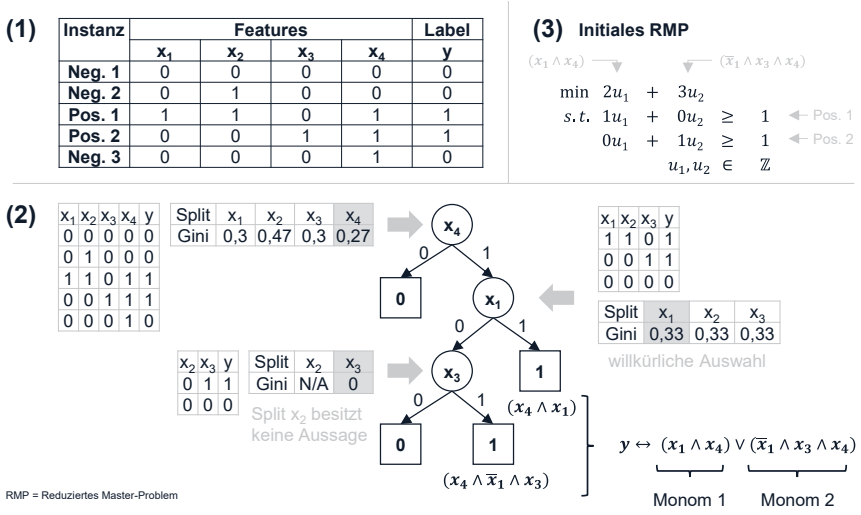


Abbildung 4.27: Initialisierung des reduzierten Master-Problems für den Beispielfall

4.4.2 Schritt 2: Lösung des Master-Problems

Das XMP wird mittels **Alg^{CG}** gelöst, indem XRMP 1 solange Spalten hinzugefügt werden, bis sich durch das Hinzufügen weiterer Spalten keine Verbesserung des optimalen Zielfunktionswerts z^{XMP^*} mehr erzielen lässt (siehe Kapitel 4.4.2.1). Für jede Spalte, die dem XRMP hinzugefügt wird, wird das zugehörige Monom in L^{Monome} gespeichert. Nachdem dem XRMP die letzte Spalte hinzugefügt wurde, liegt ein abschließendes XRMP vor, das im Folgenden als XRMP L bezeichnet wird. Die optimale Lösung des XRMP L entspricht der **optimale**n **Lösung des XMP**. Aufgrund der Relaxierung ist die optimale Lösung des XMP potenziell **nicht ganzzahlig** und damit nicht zwangsläufig eine zulässige Lösung des MP. Um eine ganzzahlige optimale Lösung zu erhalten, können nichtganzzahlige optimale Lösungen sukzessive ausgeschlossen und neue XMPs aufgestellt werden, welche wiederum mittels Alg^{CG} gelöst werden können. Auf diese

Weise ist Alg^{CG} in einen übergeordneten Algorithmus $Alg^{B\&P}$ eingebettet. **$Alg^{B\&P}$** verbindet CG basierend auf Pricing-Modellen mit einem Branching-Ansatz und ist deshalb der Klasse der sog. Branch-and-Price-Verfahren[51] zuzuordnen.

Aus didaktischen Gründen wird im Folgenden zunächst der Algorithmus Alg^{CG} und im darauffolgenden Kapitel 4.4.2.2 dessen Einbettung in den übergeordneten Algorithmus $Alg^{B\&P}$ erläutert.

4.4.2.1 Optimierung eines reduzierten Master-Problems mittels Spaltengenerierung (Alg^{CG})

Alg^{CG} löst ein XMP, ausgehend von einem initialen XRMP. Anhang A5.2 zeigt den Algorithmus in Pseudocode. Zunächst handelt es sich bei diesem XRMP um das in Schritt 1 ermittelte XRMP 1. Als erstes wird das XRMP in ein Dualproblem (DP) überführt um die Opportunitätskosten seiner Nebenbedingungen zu berechnen.

4.4.2.1.1 Aufstellen und Lösen des Dualproblems

Das DP des XRMP lässt sich mit den in Tabelle 4.3 beschriebenen Variablen sowie den in Kapitel 4.4 eingeführten Variablen und Parametern des MP – die auch im XRMP vorliegen – beschreiben.

Tabelle 4.3: Variablen des Dualproblems

$v_i \in \mathbb{R}$	Entscheidungsvariable des DP, die der Nebenbedingung i des XRMP zugeordnet ist

Dualproblem 4.12

$$max \quad \sum_{i \in \{1,\dots n^{Dp}\}} v_i$$

$$s.t.: \quad \sum_{i \in \{1,\dots n^{Dp}\}} a_{t,i} v_i \leq c_m \quad \forall m \in \{1,\dots,n^M\} \quad (1)$$

$$v_i \geq 0 \quad \forall i \in \{1,\dots,n^{Dp}\} \quad (2)$$

$$v_i \in \mathbb{R} \quad \forall i \in \{1,\dots,n^{Dp}\} \quad (3)$$

[51] Für Branch-and-Price-Verfahren sei auf Wilhelm (2001, S. 177) verwiesen

Die Zielfunktionskoeffizienten des DP sind entsprechend der rechten Seite des XRMP durchgehend 1. **Nebenbedingung 1** ergibt sich aus der Dualisierung. Da $u_1, \dots, u_{n^M} \geq 0$ gilt, muss nach der Dualisierung $v_1, \dots, v_{n^{DP}} \geq 0$ gelten (**Nebenbedingung 2**). Das DP kann mittels Solvern für lineare Optimierung (engl. Linear Programming, LP) gelöst werden. Die optimale Lösung $\boldsymbol{v}^* = (v_1^*, \dots, v_{n^{DP}}^*)$ des DP entspricht den **Opportunitätskosten** der Nebenbedingungen des XRMP. v_i^* gibt an, um wie viel sich der optimale Zielfunktionswert des XRMP verringern, d. h. verbessern, würde, wenn die rechte Seite der entsprechenden Nebenbedingung des XRMP um 1 verringert werden würde. Wird ein Eintrag der rechten Seite des XRMP um 1 verringert wird die entsprechende Nebenbedingung i inaktiv. Damit kann v_i^* im vorliegenden Fall wie folgt interpretiert werden: v_i^* entspricht der Verbesserung des Zielfunktionswertes des XRMP, falls ein Monom in das XRMP aufgenommen und ausgewählt würde, das den positiven Datenpunkt i akzeptiert. Daraus ergibt sich eine **Gewichtung der positiven Datenpunkte**. Die Verbesserung des optimalen Zielfunktionswerts des XRMP entsteht konkret dadurch, dass durch das neue Monom, das den Datenpunkt i akzeptiert, andere zuvor ausgewählte Monome, die den Datenpunkt i akzeptieren, nicht mehr benötigt werden. Dieser Verbesserung steht jedoch eine Verschlechterung des Zielfunktionswerts gegenüber, die sich daraus ergibt, dass das neue Monom in den booleschen Ausdruck aufgenommen wird. Auf Basis dessen lässt sich ein Pricing-Problem (auch Subproblem, SP) aufstellen. Dieses dient dazu ein zulässiges Monom zu ermitteln, das positive Datenpunkte mit einem möglichst hohen kumulierten Gewicht akzeptiert und dabei möglichst wenige Literale enthält.

Abbildung 4.28 (1) zeigt das initiale XRMP 1, das sich für den in Kapitel 4.4.1 eingeführten Beispielfall aus dem initialen RMP ergibt. Abbildung 4.28 (2) zeigt darüber hinaus das zugehörige DP und dessen optimale Lösung. Gemäß $\boldsymbol{v}^*$ hat ein zusätzliches Monom, das den positiven Datenpunkt 2 akzeptiert ein höheres Gewicht, als eines, das

(1) XRMP 1

$$\begin{array}{llllll} \min & 2u_1 & + & 3u_2 & & \\ s.t. & 1u_1 & + & 0u_2 & \geq & 1 \\ & 0u_1 & + & 1u_2 & \geq & 1 \\ & & & u_1, u_2 & \in & \mathbb{R} \end{array}$$

(2) Dualproblem 1

$$\begin{array}{llllll} max & v_1 & + & v_2 & & \\ s.t. & 1v_1 & + & 0v_2 & \leq & 2 \\ & 0v_1 & + & 1v_2 & \leq & 3 \\ & & & v_1, v_2 & \in & \mathbb{R} \end{array} \Rightarrow (v_1^*, v_2^*) = (2, 3)$$

XRMP = Relaxiertes reduziertes Master-Problem

Abbildung 4.28: Relaxiertes reduziertes Master-Problem und zugehöriges Dualproblem für den Beispielfall

den Positiven Datenpunkt 1 akzeptiert. Das ist dadurch begründet, dass im ersten Fall das Monom $(x_1 \wedge x_4)$ mit 2 Literalen und in letzten Fall das Monom $(\overline{x_1} \wedge x_3 \wedge x_4)$ mit 3 Literalen im initialen booleschen Ausdruck ersetzt werden kann.

4.4.2.1.2 Aufstellen des Subproblems

Mit der optimalen Lösung $\boldsymbol{v}^* = (v_1^*, \dots, v_{n^{Dp}}^*)$ des DP lässt sich das SP wie folgt mit den zusätzlichen in Tabelle 4.4 eingeführten Größen aufstellen.

Tabelle 4.4: Variablen und Parameter des Subproblems

$a_{i,n^M+1} \in \{0,1\}$	Entscheidungsvariable, die angibt, ob das neu hinzuzufügende Monom $n^M + 1$ den positiven Datenpunkt i akzeptiert (1) oder nicht (0)
$n^F \in \mathbb{N}$	Anzahl der Features im Trainingsdatensatz
$n^{Dn} \in \mathbb{N}$	Anzahl der negativen Datenpunkte im Trainingsdatensatz
$v_i^* \in \mathbb{N}$	Wert einer Entscheidungsvariablen v_i in der optimalen Lösung des DP
$w_f^n \in \{0,1\}$	Entscheidungsvariable, die angibt, ob das neu hinzuzufügende Monom die dem Feature f entsprechende Variable als negatives Literal enthält (1) oder nicht (0)
$w_f^p \in \{0,1\}$	Entscheidungsvariable, die angibt, ob das neu hinzuzufügende Monom die dem Feature f entsprechende Variable als positives Literal enthält (1) oder nicht (0)
$x_{if}^n \in \{0,1\}$	Wahrheitswert des i-ten negativen Datenpunkts hinsichtlich Feature f
$x_{if}^p \in \{0,1\}$	Wahrheitswert des i-ten positiven Datenpunkts hinsichtlich Feature f

Subproblem 4.13

$$min \quad \sum_{f \in \{1,\dots,n^F\}} w_f^p + w_f^n - \sum_{i \in \{1,\dots,n^{Dp}\}} v_i^* a_{i,n^M+1}$$

$$s.t.: \quad \sum_{f \in \{1,\dots,n^F\}} \left(w_f^p \left(1 - x_{if}^p\right) + w_f^n x_{if}^p\right) + n^F a_{i,n^M+1} \leq n^F \quad \forall i \in \{1,\dots,n^{Dp}\} \quad (1)$$

$$\sum_{f \in \{1,\dots,n^F\}} w_f^p (1 - x_{if}^n) + w_f^n x_{if}^n \geq 1 \quad \forall i \in \{1,\dots,n^{Dn}\} \quad (2)$$

$$w_f^p \in \{0,1\} \quad \forall i \in \{1,\dots,n^F\} \quad (3)$$

$$w_f^n \in \{0,1\} \quad \forall i \in \{1,\dots,n^F\} \quad (4)$$

$$a_{i,n^M+1} \in \{0,1\} \quad \forall i \in \{1,\dots,n^{Dp}\} \quad (5)$$

Durch die Variablen w_f^p und w_f^n ist das neu hinzuzufügende Monom definiert. Diese Darstellung eines Monoms wird auch als **Dual-Rail-Darstellung** bezeichnet (Ignatiev et al. 2021, S. 3809). Die Summe $\sum_{f\in\{1,\dots,n^F\}} w_f^p + w_f^n$ in der **Zielfunktion** berücksichtigt die Erhöhung, d. h. Verschlechterung, des optimalen Zielfunktionswertes des XRMP durch Hinzufügen des neuen Monoms aufgrund der in ihm enthaltenen positiven und negativen Literale. Die Summe $\sum_{i\in\{1,\dots,n^{Dp}\}} v_i^* a_{i,n^M+1}$ entspricht der Verringerung – d. h. Verbesserung – des optimalen Zielfunktionswerts des XRMP, die sich dadurch ergibt, dass das neue Monom hinzugefügt wird (siehe Kapitel 4.4.2.1.1). Sie berücksichtigt damit die Verringerung der Opportunitätskosten. **Nebenbedingung 1** sorgt dafür, dass a_{i,n^D+1} den Wert 0 annimmt, wenn der positive Datenpunkt i des Trainingsdatensatzes durch das neue Monom nicht akzeptiert wird. Dabei gibt die Summe $\sum_{f\in\{1,\dots,n^F\}} w_f^p(1 - x_{if}^p) + w_f^n x_{if}^p$ an, in wie vielen Variablen das neue Monom und Datenpunkt i nicht übereinstimmen. Eine Nichtübereinstimmung liegt vor, wenn im neuen Monom eine Variable als positives bzw. negatives Literal enthalten ist, aber Datenpunkt i für diese Variable den Wert 0 bzw. 1 aufweist. Nur wenn die Summe den Wert 0 annimmt, kann a_{i,n^M+1} den Wert 1 annehmen. **Nebenbedingung 2** stellt sicher, dass das neue Monom keinen negativen Datenpunkt des Trainingsdatensatzes akzeptiert, d. h., dass für jeden solchen Datenpunkt mindestens in einer Variablen eine Nichtübereinstimmung vorliegt. Auf Nebenbedingungen $w_f^p + w_f^n \leq 1$ zur Sicherstellung der Komplementarität von w_f^p und w_f^n für $f \in \{1, \dots, n^F\}$ kann verzichtet werden, weil durch einen Verstoß gegen eine solche Nebenbedingung der Zielfunktionswert verschlechtert würde, ohne damit eine zusätzliche Nebenbedingung 1 oder 2 zu erfüllen. Es existiert deshalb keine optimale Lösung für die $w_f^p + w_f^n = 2$ für ein $f \in \{1, \dots, n^F\}$ gilt. Abbildung 4.29 (1) zeigt das erste SP für den Beispielfall. Beispielsweise gibt der Zielfunktionskoeffizient von $a_{1,3}$ an, dass eine Verbesserung des optimalen Zielfunktionswerts des XRMP 1 um bis zu 2 möglich wäre, falls das neu hinzuzufügende Monom den positiven Datenpunkt 1 akzeptiert. Damit das neu hinzuzufügende Monom den positiven Datenpunkt 1 akzeptiert, darf es jedoch keines der Literale x_3, $\bar{x}_1$, $\bar{x}_2$ oder $\bar{x}_4$ enthalten, was durch Nebenbedingung Pos.1 sichergestellt wird. Damit das neu hinzuzufügende Monom darüber hinaus zulässig ist, muss es alle negativen Datenpunkte ausschließen, was durch die Nebenbedingungen Neg. 1, Neg. 2 und Neg. 3 gewährleistet wird.

(1) **Subproblem 1**

$$\begin{array}{lll}
min & w_1^p + w_2^p + w_3^p + w_4^p + w_1^n + w_2^n + w_3^n + w_4^n - 2\ a_{1,3} - 3\ a_{2,3} & \\
s.t. & w_3^p + w_1^n + w_2^n + w_4^n + 4\ a_{1,3} \leq 4 & (Pos.1) \\
 & w_1^p + w_2^p + w_3^n + w_4^n + 4\ a_{2,3} \leq 4 & (Pos.2) \\
 & w_1^p + w_2^p + w_3^p + w_4^p \geq 1 & (Neg.1) \\
 & w_1^p + w_3^p + w_4^p + w_2^n \geq 1 & (Neg.2) \\
 & w_1^p + w_2^p + w_3^p + w_4^n \geq 1 & (Neg.3) \\
 & w_1^p, w_2^p, w_3^p, w_4^p, w_1^n, w_2^n, w_3^n, w_4^n, a_{1,3}, a_{2,3} \in \{0,1\} &
\end{array}$$

(2) ➡ $(w_1^{p*}, w_2^{p*}, w_3^{p*}, w^{p*}_4) = (1, 0, 0, 0)$, $(w_1^{n*}, w_2^{n*}, w_3^{n*}, w_4^{n*}) = (0, 0, 0, 0)$, $(a_{1,3}^*, a_{2,3}^*) = (0, 1)$

Monom 3: x_3, Optimaler Zielfunktionswert: -2 (< 0) → Monom hinzufügen

Abbildung 4.29: Erstes Subproblem für den Beispielfall

4.4.2.1.3 Lösen des Subproblems

Das SP kann mittels ILP-Solvern gelöst werden. Die Anzahl der Variablen im SP entspricht $2n^F + n^{Dp}$ und die Anzahl der Nebenbedingungen entspricht $n^{Dp} + n^{Dn}$. Für Trainingsdatensätze mit vielen Literalen und Datenpunkten kann das SP damit groß und rechenintensiv werden. Das hat eine große Auswirkung auf die Rechenzeit, weil das SP im Zuge des Alg^{CG} i. d. R. mehrfach gelöst werden muss. Dies ist eine generelle Herausforderung der CG, weshalb Wilhelm (2001) den Einsatz problemspezifischer Heuristiken zur Lösung des SP empfiehlt. Im Rahmen der vorliegenden Arbeit wurde mit $Alg^{PricingHeuristik}$ eine solche **problemspezifische Heuristik** entwickelt, die sich in Anhang A5.3 findet. Für die heuristischen Lösungen des SP können zwei Fälle auftreten. Erstens kann sich $z^{SP^*} < 0$ ergeben. In diesem Fall wurde heuristisch eine Spalte gefunden, die dem XRMP hinzugefügt werden kann um eine Verbesserung des optimalen Zielfunktionswerts z^{XRMP^*} zu ermöglichen. Die CG kann fortgesetzt werden, ohne das SP exakt lösen zu müssen. Zweitens kann sich $z^{SP^*} \geq 0$ ergeben. In diesem Fall wurde heuristisch keine Spalte gefunden, die dem XRMP hinzugefügt werden kann um eine Verbesserung des optimalen Zielfunktionswerts z^{XRMP^*} zu ermöglichen. Da die heuristische Lösung keine Optimalität garantiert, kann nicht ausgeschlossen werden, dass eine Lösung des SP mit $z^{SP^*} < 0$ existiert. Um eine optimale Lösung des XRMP zu garantieren, muss in diesem Fall das SP exakt gelöst werden. Eine Heuristik zur Lösung des SP kann das exakte Lösen des SP somit nicht grundsätzlich ersetzen. Sie kann aber die Anzahl der Fälle reduzieren, in denen das SP exakt gelöst werden muss. Ergibt sich nach heuristischer oder optimaler Lösung des SP $z^{SP^*} < 0$ wird das XRMP aktualisiert, indem die entsprechende Spalte hinzugefügt wird. Für den Beispielfall

ergibt sich die in Abbildung 4.29 (2) gezeigte optimale Lösung des SP, die dem Monom x_3 entspricht[52]. Der optimale Zielfunktionswert des DP, -2, ist kleiner als 0, weshalb dem XRMP eine entsprechende Spalte hinzugefügt wird.

4.4.2.1.4 Aktualisierung des relaxierten reduzierten Master-Problems

Die Ausprägungen der Variablen w_f^p und w_f^n für $f \in \{1, \dots, n^F\}$ in der optimalen Lösung des SP ergeben das neu hinzuzufügende Monom in Dual-Rail-Darstellung. Prinzipiell entsprechen die Ausprägungen der Variablen a_{i,n^M+1} für $i \in \{1, \dots, n^{Dp}\}$ den Koeffizienten der neu hinzuzufügenden Spalte des XRMP. Allerdings werden diese für alle i, mit $v_i^* = 0$ im Zuge der Optimierung willkürlich gewählt, da sie keinen Einfluss auf den Zielfunktionswert haben. Um die **Koeffizienten der neuen Spalte** zu ermitteln, d. h. um zu ermitteln, welche positiven Datenpunkte das neue Monom akzeptiert, kann für jeden positiven Datenpunkt i die Bedingung $\sum_{f\in\{1,\dots,n^F\}} w_f^p(1 - x_{if}^p) + w_f^n x_{if}^p \geq 1$ ausgewertet werden. Der **Zielfunktionskoeffizient** des neuen Monoms im RMP ergibt sich als Anzahl der in ihm enthaltenen positiven und negativen Literale aus $\sum_{f\in\{1,\dots,n^F\}} w_f^p + w_f^n$. Damit kann dem XRMP eine neue Variable u_{n^M+1} mit Zielfunktionskoeffizient c_{n^M+1}und Nebenbedingungskoeffizienten a_{i,n^M+1} hinzugefügt werden. Aus dem aktualisierten XRMP kann durch Dualisierung ein neues DP abgeleitet werden[53] usw.

4.4.2.1.5 Terminierung

Der Algorithmus terminiert, sobald sich aus dem SP ergibt, dass durch Hinzufügen eines weiteren Monoms **keine Verbesserung** des optimalen Zielfunktionswerts mehr möglich ist. Die optimale Lösung des letzten XRMP entspricht der optimalen Lösung des XMP, wobei alle Variablen des XMP, die nicht im letzten XRMP enthalten sind, auf 0 gesetzt werden. Abbildung 4.30 (1) zeigt das aktualisierte XRMP und das aktualisierte DP für den Beispielfall. Im Beispielfall würde ein weiteres SP aufgestellt werden, aus dem sich das Monom x_1 ergibt. Nach der Lösung eines weiteren dualen Problems ergibt sich für das daraus folgende SP eine optimale Lösung mit Zielfunktionswert 0. Nach abschließender Lösung des XRMP ergibt sich eine ganzzahlige optimale Lösung und damit eine optimale Lösung des MP. Diese entspricht dem in Abbildung 4.30 (2)

---

[52] Neben der angegebenen optimalen Lösung besitzt das Problem in diesem Fall noch eine zweite optimale Lösung, die dem Monom x_1 entspricht, das später hinzugefügt wird. Die Wahl der optimalen Lösung hat keinen Einfluss auf die Komplexität des resultierenden booleschen Ausdrucks.

[53] Beim Einsatz von Solvern nach Stand der Technik, die das primale und duale Problem zugleich lösen, kann auf die explizite Formulierung des DP verzichtet werden.

gezeigten optimalen booleschen Ausdruck, der gegenüber der initialen Lösung 2 statt 5 Literale enthält. Eine Verzweigung ist in diesem Beispiel nicht notwendig. Ist die Lösung des XMP hingegen nichtganzzahlig ist diese Lösung durch zusätzliche Nebenbedingungen auszuschließen und es sind entsprechend neue XMPs aufzustellen. Dies geschieht im Rahmen des übergeordneten Algorithmus Alg$^{B\&P}$.

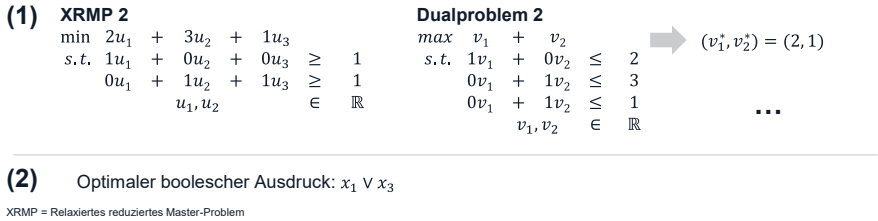


Abbildung 4.30: Zweites relaxiertes reduziertes Master-Problem und zugehöriges Dualproblem sowie finale Lösung des Master-Problems für den Beispielfall

4.4.2.2 Ermittlung eines optimalen booleschen Ausdrucks mittels Branch and Price (Alg$^{B\&P}$)

Abbildung 4.31 stellt das Vorgehen des Algorithmus Alg$^{B\&P}$ schematisch dar. Zunächst wird mittels Alg^{CG}, wie oben beschrieben, eine optimale Lösung des XMP bestimmt, wobei das in Schritt 1 ermittelte XRMP 1 als Ausgangsproblem dient. Ist die optimale Lösung des XMP nichtganzzahlig, wird sie ausgeschlossen und es werden neue, untergeordnete XMPs aufgestellt, woraus sich eine Verzweigung ergibt. Die optimalen Lösungen der neuen XMPs können erneut nichtganzzahlig sein, so dass weitere Verzweigungen entstehen können. Durch den auf diese Weise nach und nach entstehenden Baum kann das MP gelöst werden. Dabei wird Pruning mit Hilfe einer oberen Schranke $\hat{b}^o$ durchgeführt, um die Recheneffizienz zu erhöhen. Das Vorgehen wird im Folgenden erläutert.

4.4.2.2.1 Verzweigung

Um nichtganzzahlige Lösungen auszuschließen, können bekannte Verfahren der ILP eingesetzt werden, wie z. B. Schnittebenenverfahren oder Branch-and-Bound-Verfahren. Die vorliegende Arbeit nutzt ein **Branch-and-Bound-Verfahren**. Sei $\boldsymbol{u}^{XMP^*} = (u_1^*, \dots, u_{n^M}^*)$ die optimale Lösung des XMP, die, wie in Kapitel 4.4.2 beschrieben, der optimalen Lösung von XRMP L entspricht. Sei $S^{NGanzZ} = \{j \in \{1, \dots, n^M\} : u_j \notin \{0,1\}\}$ die

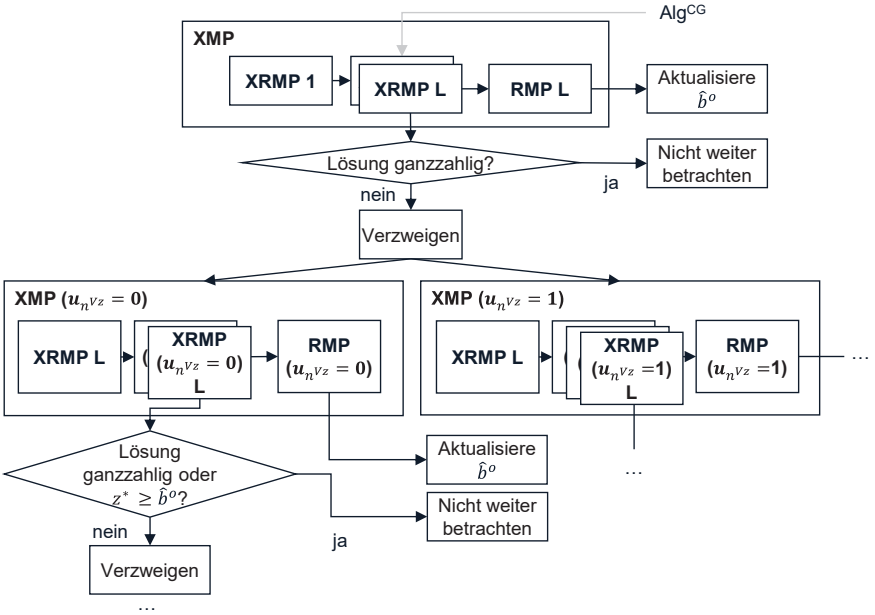


Abbildung 4.31: Schematischer Ablauf des Algorithmus Alg$^{B\&P}$

Menge aller Indizes nichtganzzahliger Entscheidungsvariablen in $\boldsymbol{u}^{XMP^*}$. Es wird der **erste Index** $n^{Vz} = min(\{i \in S^{NGanzZ}\})$ ausgewählt, für den die zugehörige Variable in der optimalen Lösung nichtganzzahlig ist. In diesem Index wird das XMP verzweigt (engl. to branch), indem jeweils ein XMP($u_{n^{Vz}} = 0$) und ein XMP($u_{n^{Vz}} = 1$) mit den zusätzlichen Nebenbedingungen $u_{n^{Vz}} = 0$ bzw. $u_{n^{Vz}} = 1$ erstellt werden[54]. Diese untergeordneten XMPs werden analog zum initialen XMP mittels Alg^{CG} gelöst. Hierfür kann XRMP L als initiales XRMP verwendet werden, wobei die Variable $u_{n^{Vz}}$ entsprechend fixiert wird.

[54] Bzgl. der hier vorgestellten Methode 4 besteht der Fokus der vorliegenden Arbeit darin, CG im Rahmen von Branch & Price für die datenbasierte Erstellung von Regeln in LLKM grundsätzlich nutzbar zu machen. Es ist nicht der Anspruch der vorliegenden Arbeit, das Potenzial, das Branch & Price für die effiziente Lösung entsprechender Optimierungsprobleme bietet, auszuschöpfen. Um die Recheneffizienz weiter zu erhöhen, besteht Potenzial für den Einsatz anspruchsvollerer Meta-Verfahren, wie z. B. Branch & Cut, anspruchsvollerer Modellierungen wie z. B. Dantzig-Wolfe-Reformulierungen und anspruchsvollere Verzweigungsstrategien wie z. B. eine Verzweigung mit Vorausschau. Eine Herausforderung besteht jeweils darin, diese Ansätze im SP zu berücksichtigen.

Für XRMPs mit Variablen, die auf 0 fixiert sind, sind die zugehörigen SPs zusätzlich zu beschränken. Es dürfen keine Spalten in das XRMP aufgenommen werden, die identisch zu Spalten sind, deren zugehörige Variablen im Zuge der Verzweigung auf 0 fixiert wurden – andernfalls würden Endlosschleifen entstehen. Es muss deshalb eine **Liste mit ausgeschlossenen Monomen** ($L^{ExklMonome}$) geführt werden, die beim Aufstellen des SP durch Nebenbedingungen ausgeschlossen werden. Sei $\mu \in L^{ExklMonome}$ ein zuvor ausgeschlossenes Monom und $(\boldsymbol{w}^{p\mu}, \boldsymbol{w}^{n\mu}) = (w_1^{p\mu}, \dots, w_{n^F}^{p\mu}, w_1^{n\mu}, \dots, w_{n^F}^{n\mu})$ dessen Dual-Rail-Darstellung. Dann müssen dem SP folgende Nebenbedingungen hinzugefügt werden:

$$\begin{aligned} &\sum_{f\in\{1,\dots n^F\}} w_f^{p\mu} w_f^p + \sum_{f\in\{1,\dots n^F\}} \left(1-w_f^{p\mu}\right)\left(1-w_f^p\right) \\ &+ \sum_{f\in\{1,\dots n^F\}} w_f^{n\mu} w_f^n + \sum_{f\in\{1,\dots n^F\}} \left(1-w_f^{n\mu}\right)\left(1-w_f^n\right) \end{aligned} \quad \leq \quad \begin{aligned} &2n^F - 1 \\ &\forall \mu \in L^{ExklMonome} \end{aligned} . \qquad 4.14$$

Die Terme $\sum_{f\in\{1,\dots,n^F\}} w_f^{p\mu} w_f^p$ und $\sum_{f\in\{1,\dots,n^F\}} w_f^{n\mu} w_f^n$ entsprechen den Anzahlen von Literalen, die sowohl für das ausgeschlossene Monom als auch für die Lösung in positiver bzw. negativer Form auftreten. Die Terme $\sum_{f\in\{1,\dots,n^F\}}\left(1-w_f^{p\mu}\right)\left(1-w_f^p\right)$ und $\sum_{f\in\{1,\dots,n^F\}}\left(1-w_f^{n\mu}\right)\left(1-w_f^n\right)$ entsprechen den Anzahlen von Literalen, die weder für das ausgeschlossene Monom noch für die Lösung in positiver Form auftreten. Insgesamt entspricht die linke Seite der Nebenbedingungen somit der Anzahl von Stellen, in denen die Lösung und das ausgeschlossene Monom übereinstimmen. Diese Übereinstimmung muss kleiner als $2n^F$ – die Anzahl aller Stellen – sein.

Bei der Fixierung von Variablen eines XRMP auf 0 können Fälle eintreten, für die das XRMP keine Lösung mehr besitzt. Ggf. sind Monome in das XRMP aufzunehmen, um die Lösbarkeit wiederherzustellen. Auf diese Fälle geht Anhang A5.4 ein.

Mit der vorgestellten, ggf. mehrfachen, **Verzweigung** kann eine ganzzahlige optimale Lösung des XMP und damit eine optimale Lösung des MP gefunden werden. Allerdings kann die Effizienz des Vorgehens erhöht werden, indem nicht alle Teilbäume des resultierenden Suchbaums vollständig betrachtet werden. Es wird deshalb – wie für Branch-and-Bound-Verfahren üblich – Pruning durchgeführt, indem die möglichen Zielfunktionswerte eines Teilbaums mit einer oberen Schranke (engl. bound) verglichen werden.

4.4.2.2.2 Pruning

XRMP L kann nicht nur als initiales XRMP für untergeordnete XMPs verwendet werden, sondern auch um eine obere Schranke $\hat{b}^o$ für z^{MP^*} – den optimalen Zielfunktionswert des MP – zu berechnen. Dazu wird XRMP L derelaxiert indem seine Entscheidungsvariablen auf ganzzahlige Werte beschränkt werden. Es ergibt sich ein RMP L, dessen optimale Lösung ganzzahlig und damit eine zulässige Lösung des MP ist. Da RMP L gegenüber dem MP reduziert ist, ist sein Lösungsraum eine Teilmenge des Lösungsraums des MP. Damit ist sein optimaler Zielfunktionswert z^{RMP^*} eine **obere Schranke** für den optimalen Zielfunktionswert z^{MP^*}. Eine solche obere Schranke ergibt sich auch für alle letzten XRMP aller untergeordneten XMPs. Die obere Schranke kann genutzt werden, um Teilprobleme, die sich aus der Verzweigung ergeben, von der Betrachtung auszuschließen.

Sei XMP(…) ein durch Variablenfixierungen beschränktes XMP, das im Suchbaum von $Alg^{B\&P}$ auftritt. Da die Zielfunktionskoeffizienten und die Entscheidungsvariablen des MP natürlich sind, ist der Zielfunktionswert des MP immer eine natürliche Zahl und somit ist auch die obere Schranke $\hat{b}^o$ eine natürliche Zahl. Gilt nun

$$\left\lceil z^{XMP(\dots)^*} \right\rceil \geq \hat{b}^o \tag{4.15}$$

gilt auch für das zu XMP(…) gehörige MP(…) mit Variablenfixierung

$$\left\lceil z^{MP(\dots)^*} \right\rceil \geq \hat{b}^o. \tag{4.16}$$

D. h. für das MP existiert unter den gegebenen Variablenfixierungen keine Lösung mit einem besseren Zielfunktionswert als $\hat{b}^o$. Damit können sich aus XMPs, die XMP(…) untergeordnet sind und deren Lösungsräume damit Teilmengen des Lösungsraums von XMP(…) sind, ebenfalls keine Lösungen mit besseren Zielfunktionswerten als $\hat{b}^o$ ergeben. Verzweigungen von XMP(…) müssen damit nicht betrachtet werden. Durch das Pruning mittels Kriterium 4.16 können optimale Lösungen des MP effizienter ermittelt werden.

4.4.2.2.3 Terminierung

Aus jedem betrachteten XMP ergibt sich durch Überführung des letzten XRMP in ein RMP wie oben beschrieben eine obere Schranke $\hat{b}^o$ für z^{MP^*}. Ist diese geringer als die geringste bisher gefundene obere Schranke $\hat{b}^o$ wird $\hat{b}^o$ entsprechend aktualisiert. Die Menge der aktiven Monome des RMP, d. h. der Monome, die zu Basisvariablen der optimalen Lösung gehören, werden als bester bisher gefundener boolescher Ausdruck

gespeichert. Sobald **alle XMPs des Suchbaums gelöst** oder im Zuge des Prunings von der Betrachtung ausgeschlossen wurden, entspricht der beste gefundene Ausdruck einem komplexitätsminimalen Ausdruck, der eine perfekte Trainingsgenauigkeit für den Trainingsdatensatz aufweist.

Damit können durch Methode 4 Regeln für Auswahlbedingungen in LLKMs datenbasiert erstellt werden. Für andere Arten von Regeln kann – wie oben erläutert – auf andere ML-Verfahren nach Stand der Forschung zurückgegriffen werden. Mittels Methode 4 erstellte Regeln ergeben zusammen mit einer mittels Methode 2 erstellten MSTL und zugehörigen, mittels Methode 3 erstellten MAPLs ein vollständiges LLKM. Falls nicht ausreichend viele Datenpunkte – d. h. nicht ausreichend viele Varianten mit zugehörigen VSTLs und VAPLs – zur Verfügung stehen, kann Methode 5 eingesetzt werden, um weitere Daten systematisch zu generieren.

4.5 Methode 5: Auswahl von repräsentativen Varianten zur Erweiterung der Datenbasis

Im Folgenden wird die im Rahmen der vorliegenden Arbeit entwickelte Methode 5 zur Lösung des Problems 5 – der Auswahl von repräsentativen Varianten zur Erweiterung der Datenbasis für die datenbasierte Erstellung von LLKMs – vorgestellt. Methode 5 behebt das in Kapitel 3.5.3 beschriebene Lösungsdefizit nach Stand der Forschung. Zulässige Varianten sind durch die Produktmerkmale, deren zulässige Ausprägungen sowie die Beschränkungen im HLKM beschrieben. Der sich daraus ergebende Konfigurationsraum ist typischerweise zu groß um ein poolbasiertes aktives Lernen (AL) einzusetzen. Durch die Ausprägung der definierten Produktmerkmale unter Berücksichtigung der Beschränkungen[55] des HLKM lassen sich jedoch mittels **Membership Query Synthesis** (MQS) Datenpunkte generieren, die zulässigen Varianten aus dem Konfigurationsraum entsprechen.

Prinzipiell können die in Kapitel 2.3.2 und 3.5.2 vorgestellten Kriterien des AL verwendet werden, um Varianten – beschrieben durch ihre Produktmerkmale – aus dem Konfigurationsraum auszuwählen. Dabei geht jede Variante als Datenpunkt in den Trainingsdatensatz ein, wie in Kapitel 4.1.3 erläutert. Die Varianten werden von einem Experten mit einer VSTL und VAPLs versehen, wie in Kapitel 4.1.2 beschrieben. Dadurch

[55] Sind die Beschränkungen nicht vollständig definiert können diese im Rahmen des in Kapitel 4.1.3 beschriebenen iterativen Prozesses reaktiv vervollständigt werden, indem gewählte Varianten, die unzulässig sind durch geeignete Beschränkungen ausgeschlossen werden.

entsteht ein annotierter Datenpunkt im Trainingsdatensatz. Zum Erzeugen mehrerer zusätzlicher Varianten kann dieses Vorgehen mehrfach wiederholt werden. Auf diese Weise ist Methode 5 in einen **iterativen Prozess** eingebettet.

Es wird weiterhin von binären Features ausgegangen, womit eine Variante als Binärvektor beschrieben werden kann. Aus den im Rahmen von Methode 5 generierten Daten sollen mittels Methode 2, Methode 3 und Methode 4 neben Regeln auch MSTLs und MAPLs erstellt werden. Methode 5 beschränkt sich jedoch darauf, Varianten auszuwählen, die einen hohen Informationsgewinn für die Erstellung von Regeln erwarten lassen. Außerdem muss das in Kapitel 4.1.1 dargestellte KM für die Anwendbarkeit der Methode eingeschränkt werden. Es wird davon ausgegangen, dass alle Parameter des LLKM direkt von der Vertriebskonfiguration abhängen. Dies beschränkt die Mächtigkeit des KM insofern nicht, als alle Parameter zwangsläufig indirekt von der Vertriebskonfiguration abhängen und somit schlicht indirekte Regeln durch direkte Regeln abgebildet werden. Im Folgenden entspricht ein **Datenpunkt** einer Variante. Die Produktmerkmale – ggf. binär codiert – entsprechen den **Features** des Datenpunkts und die zu prädizierenden Parameter der MSTL und der MAPLs entsprechen seinen **Labels**. Es liegt somit ein Multi-Label-Problem vor.

Die in der vorliegenden Arbeit verwendeten Klassifikationsmodelle in Form von booleschen Ausdrücken lassen keine Unsicherheit bzgl. Datenpunkten erkennen, werden nicht mittels gradientenbasierter Verfahren gelernt und induzieren keine Entscheidungsgrenze. Deshalb kommen die folgenden in Kapitel 2.3.2 eingeführten informationsbasierten Kriterien des AL für die Auswahl von Varianten nicht in Frage: Unsicherheit des Modells, erwartete Modellveränderung sowie Nähe zur Entscheidungsgrenze. Als informationsbasiertes Kriterium nach Tharwat & Schenck (2023) verbleibt die Heterogenität der Vorhersagen eines Komitees nach dem QBC-Ansatz. Der QBC-Ansatz hat jedoch den Nachteil, dass jedes Klassifikationsmodell des Komitees nur auf einer Teilmenge des Trainingsdatensatzes trainiert wird. Damit liegen zum einen immer suboptimale Modelle vor, da jeweils vorhandene Informationen unberücksichtigt bleiben. Zum anderen kann nicht gewährleistet werden, dass die Modelle eine perfekte Trainingsgenauigkeit aufweisen und damit überhaupt für die Problemstellung geeignet sind. Stattdessen wird in Schritt 1 der Methode 5 das bestehende Multi-Label-Problem mit n^L Labels zunächst nach dem Prinzip der Binary-Relevance in n^L Probleme mit jeweils einem Label aufgeteilt. Je Label wird ein Versionenraum (VR) erstellt, der Modelle in Form von booleschen Ausdrücken enthält, die jeweils auf allen Trainingsdaten trainiert

wurden und eine perfekte Trainingsgenauigkeit aufweisen. Die Heterogenität der Vorhersagen der Modelle für bestimmte Datenpunkte wird als eines der Auswahlkriterium verwendet – im Folgenden als **Separationskriterium** bezeichnet.

In dem Anwendungsfall der vorliegenden Arbeit entspricht die Gesamtheit der möglichen Datenpunkte dem Konfigurationsraum. Wird dieser als Pool im Sinne des AL aufgefasst, ist er zu groß, um Cluster oder Dichten zu bestimmen. Von den repräsentationsbasierten Kriterien nach Tharwat & Schenck (2023) können deshalb clusterbasierte und dichtebasierte Verfahren ausgeschlossen werden. **Diversität** wird hingegen als Auswahlkriterium für die Methode 5 genutzt. Auf Basis der Kriterien Separation und Diversität werden in Schritt 2 der Methode 5 Varianten aus dem Konfigurationsraum ausgewählt. Dabei werden die Beschränkungen im HLKM berücksichtigt. Abbildung 4.32 stellt die beiden Schritte der Methode 5 dar.

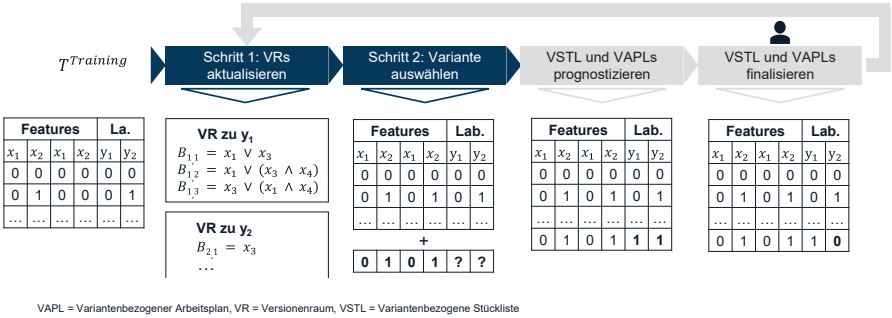


Abbildung 4.32: Überblick über die Schritte der Methode 5

4.5.1 Schritt 1: Versionenräume aktualisieren

Im Folgenden wird mit VR l der VR bezeichnet, der die Modelle für Label l enthält. Die Anzahl zulässiger Modelle für einen VR wächst exponentiell mit der Anzahl vorliegender Features. Deshalb werden nur die relevantesten n^{VR} Modelle in einen VR aufgenommen. Dabei ist n^{VR} ein Parameter der Methode, der die Größe der VRs angibt. Wie in Kapitel 2.3.2 dargelegt, sind Modelle umso relevanter, je geringer ihre Komplexität ist. Konkret enthält somit jeder VR l die n^{VR} **komplexitätsminimale**n **boolesche**n **Ausdrücke**, die für das Label l eine perfekte Trainingsgenauigkeit aufweisen.

Wie oben beschrieben, ist Methode 5 in einen iterativen Prozess eingebettet. Für alle Iterationen mit Ausnahme der ersten existieren aus vorherigen Iterationen bereits VRs. Da jedoch der Datensatz beim Abschluss der letzten Iteration um einen Datenpunkt erweitert wurde, ist zunächst zu überprüfen, ob die Modelle eines VR den zuletzt hinzugefügten Datenpunkt auf das gegebene Label abbilden. Falls nicht, werden sie aus dem VR entfernt. Nach dem **Entfernen unzulässiger Modelle** erfolgt das **Wiederauffüllen der VR** auf n^{VR} Modelle, indem neue Modelle berechnet werden. Dafür kann grundsätzlich Methode 4 einmal oder mehrmals angewandt werden. Es muss dabei gewährleistet werden, dass bereits im VR **enthaltene Modelle** nicht erneut erstellt werden. Hierfür werden zunächst alle Monome, die Bestandteil der r bereits im VR enthaltenen Modelle sind als Spalten in das initiale RMP und XRMP eingefügt – neben den Monomen der heuristischen Lösung (siehe Kapitel 4.4.1). Es kann ausgeschlossen werden, dass Modelle, die bereits im VR existieren erneut erzeugt werden, indem die Wahl der entsprechenden Monome im RMP beschränkt wird. Seien $\boldsymbol{u_k^{Vorh}} = \left(u_{k,1}^{Vorh}, \dots, u_{k,n^M}^{Vorh}\right) \forall\, k \in \{1, \dots, r\}$ diejenigen Lösungen des wie beschrieben aufgestellten, initialen XRMP. Diese entsprechen den im VR bereits vorhandenen Modellen. Dann können diese Lösungen durch folgende Nebenbedingungen im RMP und XRMP ausgeschlossen werden

$$\sum_{m\in\{1,\dots,n^M\}} u_{k,m}^{Vorh}(1-u_m) + (1-u_{k,m}^{Vorh})u_m \quad \geq \quad 1 \quad \forall k \in \{1, \dots, r\}. \tag{4.17}$$

Die linke Seite der Nebenbedingung entspricht der Anzahl von Stellen, in denen $\boldsymbol{u_k^{Vorh}}$ und $\boldsymbol{u}$ nicht übereinstimmen. Damit die beiden Vektoren nicht identisch sind, muss diese Anzahl größer oder gleich 1 sein. Umgeformt in Standardrepräsentation für Nebenbedingungen in Minimierungsproblemen ergibt sich

$$\sum_{m\in\{1,\dots,n^M\}} (1-2u_{k,m}^{Vorh})u_m \quad \geq \quad 1 - \sum_{m\in\{1,\dots n^M\}} u_{k,m}^{Vorh}\, \forall k \in \{1, \dots, r\}. \tag{4.18}$$

D. h. die entsprechenden Zeilen der Koeffizientenmatrix des RMP enthalten einen Eintrag -1 für alle Monome, die bereits Teil eines Ausdrucks im VR sind und 1 für alle anderen Monome. Monome, die bereits Teil eines Modells im VR sind, werden außerdem in die Liste $L^{ExklMonome}$ ausgeschlossener Monome für das SP (siehe Kapitel 4.4.2.1.2) aufgenommen, so dass sie nicht erneut in das RMP eingefügt werden. Damit ist ausgeschlossen, dass Modelle, die bereits Teil des VR sind durch Anwendung von Methode 4 erneut erstellt werden.

Zuletzt sind für die Anwendung von Alg^{CG} noch folgende **Details der Umsetzung** zu berücksichtigen. In Methode 4 entspricht jeder Nebenbedingung eine Variable des DP und somit eine Variable des SP (siehe Kapitel 4.4.2.1.2). Dies gilt nicht für die Nebenbedingungen 4.18, die im SP nicht berücksichtigt werden, da sie keinen Datenpunkten des Trainingsdatensatzes entsprechen. Außerdem werden die Koeffizienten dieser Nebenbedingungen nicht wie in Kapitel 4.4.2.1.4 beschrieben beim Einfügen einer neuen Spalte in das RMP berechnet. Stattdessen werden sie immer auf 1 gesetzt, da ein neu in das RMP aufgenommenes Monom, wie zuvor beschrieben, nicht Teil eines Modells des VR sein kann.

Abbildung 4.33 zeigt einen Trainingsdatensatz (1) für einen Beispielfall, der in Kapitel 4.5.2 aufgegriffen wird und die zugehörigen VRs (2). Nach Durchführung von Schritt 1 liegt ein vollständiger VR mit zulässigen Modellen je Label des Trainingsdatensatzes vor. In Schritt 2 wird auf Basis dessen eine Variante ausgewählt.

(1)

Features				Labels	
x_1	x_2	x_3	x_4	y_1	y_2
0	0	0	0	0	0
0	1	0	0	0	0
1	1	0	1	1	0
0	0	1	1	1	1
0	0	0	1	0	0

(2)

VR zu y_1

$B_{1,1} = x_1 \vee x_3$
$B_{1,2}' = x_1 \vee (x_3 \wedge x_4)$
$B_{1,3}' = x_3 \vee (x_1 \wedge x_4)$

VR zu y_2

$B_{2,1} = x_3$
$B_{2,2}' = x_3 \wedge x_4$
$B_{2,3}' = x_3 \wedge \sim x_2$

VR = Versionenraum

Abbildung 4.33: Trainingsdatensatz für einen Beispielfall (1) und zugehörige Versionenräume (2)

4.5.2 Schritt 2: Variante auswählen

Bei der Auswahl von Varianten sind die Beschränkungen des HLKM zu berücksichtigen. Unter allen zulässigen Varianten ist diejenige auszuwählen, die zu einer hohen Heterogenität der Vorhersagen der Modelle des VR führt (Separationskriterium) und einen großen Abstand zu bereits zuvor ausgewählten Varianten, d. h. Datenpunkten im Trainingsdatensatz, aufweist (Diversitätskriterium). Es kann somit ein **multikriterielles Optimierungsproblem** aufgestellt werden, das im Folgenden hergeleitet wird. Zunächst wird die Modellierung des Separationskriteriums erläutert.

4.5.2.1 Modellierung des Separationskriteriums

Um die Heterogenität der Vorhersagen der Modelle eines VR, bezogen auf eine Variante mit den Featureausprägungen $\boldsymbol{x}$, analytisch zu modellieren, wird ein Maß benötigt,

das analytisch beschrieben werden kann. Seien die booleschen Ausdrücke $B_{l,j}(\boldsymbol{x})$ und $B_{l,k}(\boldsymbol{x})$ die Modelle j und k des VR l. Deren Funktionswerte hängen von der gewählten Variante $\boldsymbol{x}$ ab. Die beiden Ausdrücke stimmen nicht überein, falls ihre Kontravalenz

$$B_{l,j}(\boldsymbol{x}) \oplus B_{l,k}(\boldsymbol{x}) \tag{4.19}$$

wahr ist, was

$$B_{l,j}(\boldsymbol{x}) \wedge \neg B_{l,k}(\boldsymbol{x}) \vee \neg B_{l,j}(\boldsymbol{x}) \wedge B_{l,k}(\boldsymbol{x}) \tag{4.20}$$

entspricht. Der resultierende boolesche Ausdruck (Formel 4.19) wird im Folgenden als **Modellseparationsformel** (MSF) $B_{l,j,k}^{MS}(\boldsymbol{x})$ bezeichnet. Eine MSF $B_{l,j,k}^{MS}$ gibt an, für welche Varianten die Vorhersagen der Modelle $B_{l,j}(\boldsymbol{x})$ und $B_{l,k}(\boldsymbol{x})$ nicht übereinstimmen, d. h. die entsprechenden Modelle separiert werden. Sind zwei Modelle durch eine Variante separiert, kann eines der beiden ausgeschlossen werden, sobald die Labels der Variante bekannt sind. Varianten, die möglichst viele MSF eines VR erfüllen sind somit tendenziell zu bevorzugen.

Es existiert in einem VR eine MSF für jedes Paar an Modellen, woraus sich $\frac{n^{VR}*(n^{VR}-1)}{2}$ MSF je VR ergeben. Nicht alle MSF eines VR können zugleich erfüllt sein. Die maximale Anzahl von Modellseparationen für n^{VR} Modelle tritt auf, wenn die Modelle des VR eine Variante in möglichst gleich vielen Fällen auf wahr und auf falsch abbilden. Die **Anzahl von Separationen** ist dann $\left\lfloor \frac{n^{VR}}{2} \right\rfloor \left\lceil \frac{n^{VR}}{2} \right\rceil$. Über alle VRs hinweg ergeben sich somit bei n^L Labels $n^L \left\lfloor \frac{n^{VR}}{2} \right\rfloor \left\lceil \frac{n^{VR}}{2} \right\rceil$ mögliche Separationen. Im besten Fall schließt eine Variante aufgrund ihrer Labels über alle VRs hinweg $n^F \left\lfloor \frac{n^{VR}}{2} \right\rfloor \left\lceil \frac{n^{VR}}{2} \right\rceil$ Modelle aus. Um die MSF für eine Variante möglichst effizient auswerten zu können, werden sie in Konjunktive Normalform (KNF), d. h. konjunktiv verknüpfte Klauseln, überführt. Wird eine Variante $\boldsymbol{x}$ von allen Klauseln akzeptiert, d. h. auf wahr abgebildet, ist die MSF erfüllt. Abbildung 4.34 zeigt die MSFs für den Beispielfall.

$$\begin{aligned} B_{1,1,2}^{MSF} &= B_{1,1} \oplus B_{1,2} &&= \sim x_1 \wedge x_3 \wedge \sim x_4 \\ B_{1,1,3}^{MSF} &= B_{1,1} \oplus B_{1,3} &&= x_1 \wedge \sim x_3 \wedge \sim x_4 \\ B_{1,2,3}^{MSF} &= B_{1,2} \oplus B_{1,3} \\ &= \sim x_4 \wedge (x_1 \vee x_3) \wedge (\sim x_1 \vee \sim x_3) \end{aligned}$$

$$\begin{aligned} B_{2,1,2}^{MSF} &= B_{2,1} \oplus B_{2,2} &&= x_3 \wedge \sim x_4 \\ B_{2,1,3}^{MSF} &= B_{2,1} \oplus B_{2,3} &&= x_2 \wedge x_3 \\ B_{2,2,3}^{MSF} &= B_{2,2} \oplus B_{2,3} \\ &= x_3 \wedge (x_2 \vee \sim x_4) \wedge (x_4 \vee \sim x_2) \end{aligned}$$

Abbildung 4.34: Modellseparationsformeln für den Beispielfall

Es existieren insgesamt $n^L * \frac{n^{VR}*(n^{VR}-1)}{2}$ MSF von denen, wie oben beschrieben, durch eine Variante nur ein Teil erfüllt werden kann. Es muss folglich entschieden werden, welche MSFs erfüllt werden sollen. Hierfür wird auf Basis der folgenden Überlegung eine Gewichtung der MSFs eingeführt.

4.5.2.2 Gewichtung von Modellseparationsformeln

Durch die sukzessive Erweiterung des Trainingsdatensatzes werden sukzessive Modelle, die für einen neuen Datenpunkt nicht gültig sind, ausgeschlossen und durch neue zulässige Modelle mit minimaler Komplexität ersetzt. Die Komplexität von Modellen, die in den VR aufgenommen werden, ist deshalb mindestens so groß wie die der Modelle, die zuvor ausgeschlossen wurden. Die Komplexität der Modelle im VR nimmt damit im Laufe der Iterationen tendenziell zu. Sobald die Komplexität der Modelle im VR der Komplexität des tatsächlich gültigen Modells entspricht, gelangt das tatsächlich gültige Modell in den VR. Das tatsächlich gültige Modell verbleibt für alle Iterationen im VR. Die Komplexität der hinzukommenden Modelle im VR steigt weiterhin. Dadurch unterscheidet sich die Komplexität des Modells mit der geringsten Komplexität im VR und den anderen Modellen im VR zunehmend. Je größer diese Differenz, desto wahrscheinlicher ist es, dass es sich bei dem Modell mit der geringsten Komplexität um das tatsächlich gültige Modell handelt. Modelle auszuschließen, die bereits eine hohe **Komplexitätsdifferenz** zum komplexitätsminimalen Modell im VR aufweisen, ist daher weniger relevant als Modelle mit geringer Komplexitätsdifferenz auszuschließen. MSFs, die sich auf Modelle mit hoher Komplexitätsdifferenz beziehen, erhalten deshalb ein geringeres Gewicht. Sei $a_{l,j}^{Komp}$ die absolute Komplexitätsdifferenz eines Modells j aus VR l, d. h. die Differenz von dessen Literalanzahl zur Literalanzahl des komplexitätsminimalen Modells im VR. Dann erhält die MSF jk des VR l das Gewicht

$$r_{l,j,k}^{MS} = 2^{-\max(a_{l,j}^{Komp}, a_{l,k}^{Komp})} \qquad 4.21$$

falls $\max\left(a_{l,j}^{Komp}, a_{l,k}^{Komp}\right) \leq \hat{a}^{Komp}$ und 0 sonst. $\hat{a}^{Komp}$ stellt eine Grenze dar, ab der MSF nicht mehr betrachtet werden. Dies verringert die Komplexität des in Kapitel 4.5.2.5 vorgestellten Optimierungsproblems.

4.5.2.3 Modellierung des Diversitätskriteriums

Das Kriterium der Diversität lässt sich als geringster Abstand einer Variante zu allen n^D im Trainingsdatensatz bereits vorhandenen Datenpunkten, d. h. Varianten, beschreiben. Es gilt für den Abstand zu den bereits im Datensatz vorhandenen Datenpunkten

$$d^{Dist} = \min(d_1^{Dist}, \dots, d_{n^D}^{Dist}) , \tag{4.22}$$

wobei d_i^{Dist} den Abstand zu einem der n^D Datenpunkte bezeichnet. Da, wie zuvor erläutert, von binären Features ausgegangen wird, wird als Maß für den Abstand einer Variante zu einer zuvor gewählten Variante die **Hamming-Distanz** verwendet. Diese gibt an, in wie vielen Stellen sich zwei binäre Vektoren unterscheiden. Damit ist dieses Maß durch die Anzahl n^F an Features nach oben beschränkt.

4.5.2.4 Modellierung der Beschränkungen des High-Level-Konfigurationsmodells

Es bleibt sicherzustellen, dass eine Variante den Beschränkungen des HLKM genügt. Jede Beschränkung des HLKM lässt sich als boolescher Ausdruck beschreiben. Eine Variante ist zulässig, wenn sie alle diese Ausdrücke erfüllt. Sie muss also einen booleschen Ausdruck $B^{HL}(\boldsymbol{x})$ erfüllen, der eine konjunktive Verknüpfung dieser Ausdrücke darstellt und im Folgenden als **High-Level-Formel** (HLF) bezeichnet wird. Um die HLF effizient auswerten zu können, wird sie in KNF überführt. Eine Variante ist somit zulässig, wenn sie alle Klauseln der HLF erfüllt.

Vor dem zuvor beschriebenen Hintergrund kann das Optimierungsproblem zur Auswahl einer Variante aufgestellt werden.

4.5.2.5 Aufstellen des Optimierungsproblems zur Auswahl einer Variante

Das Optimierungsproblem enthält die in

Tabelle 4.5 beschriebenen Variablen und Parameter und wird im Folgenden näher erläutert.

Tabelle 4.5: Variablen und Parameter des Optimierungsproblems zur Berechnung der optimalen zu wählenden Variante

$d^{Dist} \in \mathbb{N}$	Entscheidungsvariable, die den kleinsten Abstand einer Lösung, d. h. Variante, zu allen bereits im Trainingsdatensatz befindlichen Varianten angibt
$d_{l,j,k}^{MS} \in \{0,1\}$	Entscheidungsvariable, die angibt, ob sich die Vorhersagen der Modelle j und k des VR des Labels l für eine Lösung, d. h. eine Variante, unterscheiden (1) oder nicht (0)

$n^D \in \mathbb{N}$	Anzahl der Datenpunkte im Trainingsdatensatz
$n^F \in \mathbb{N}$	Anzahl der Features des Trainingsdatensatzes
$n^{Kl,HL} \in \mathbb{N}$	Anzahl der Klauseln in der HLF
$n_{l,j,k}^{Kl,MS} \in \mathbb{N}$	Anzahl der Klauseln in der MSF jk eines VR l
$n^L \in \mathbb{N}$	Anzahl der Labels im Trainingsdatensatz
$n^{VR} \in \mathbb{N}$	Definierte Größe der VR
$r_{l,j,k}^{MS}$	Gewichtung der MSF, die den Modellen j und k im VR l zugeordnet ist
$w^{MS} \in [0,1]$	Gewichtung des Kriteriums Modellseparation
$w_{f,l,j,k,m}^{p,MS} \in \{0,1\}$	Parameter, der angibt, ob Klausel m der Modellseparationsformel jk des VR des Labels l das Feature f als positives Literal enthält (1) oder nicht (0)
$w_{f,l,j,k,m}^{n,MS} \in \{0,1\}$	Parameter, der angibt, ob Klausel m der Modellseparationsformel jk des VR i das Feature f als negatives Literal enthält (1) oder nicht (0)
$w_{f,m}^{p,HL} \in \{0,1\}$	Parameter, der angibt, ob Klausel m der HLF das Feature f als positives Literal enthält (1) oder nicht (0)
$w_{f,m}^{n,HL} \in \{0,1\}$	Parameter, der angibt, ob Klausel m der HLF das Feature f als negatives Literal enthält (1) oder nicht (0)
$x_f \in \{0,1\}$	Ausprägung von Feature f in einer zu wählenden Variante
$x_{i,f}^{Vorh} \in \{0,1\}$	Ausprägung von Feature f in der bereits im Trainingsdatensatz vorhandenen Variante i

Optimierungsproblem zur Auswahl einer Variante 4.23

$$max \quad w^{MS} \frac{1}{n^L \left\lfloor \frac{n^{VR}}{2} \right\rfloor \left\lceil \frac{n^{VR}}{2} \right\rceil} \sum_{i \in \{1,\dots,n^L\}} \sum_{j \in \{1,\dots,n^{VR}\}} \sum_{k \in \{j+1,\dots,n^{VR}\}} r_{l,j,k}^{MS} d_{l,j,k}^{MS} + (1 - w^{MS}) \frac{1}{n^F} \sum_{i \in \{1,\dots,n^D\}} d^{Dist}$$

$s.t.$:

$$\sum_{f \in \{1,\dots,n^F\}} w_{f,l,j,k,m}^{p,MS} x_f + \sum_{f \in \{1,\dots,n^F\}} w_{f,l,j,k,m}^{n,MS} (1 - x_f) \geq d_{j,k,l}^{MS} \quad \begin{matrix} \forall l \in \{1,\dots,n^L\}, \\ \forall j \in \{1,\dots,n^{VR}\}, \\ \forall k \in \{j+1,\dots,n^{VR}\}, \\ \forall m \in \{1,\dots,n_{l,j,k}^{Kl,MS}\} \end{matrix} \quad (1)$$

$$\sum_{f \in \{1,\dots,n^F\}} x_{i,f}^{Vorh} (1 - x_f) + \sum_{f \in \{1,\dots,n^F\}} (1 - x_{i,f}^{Vorh}) x_f \geq d^{Dist} \quad \forall i \in \{1,\dots,n^D\} \quad (2)$$

$$\sum_{f\in\{1,\dots,n^F\}} w_{f,m}^{p,HL} x_f + \sum_{f\in\{1,\dots,n^F\}} w_{f,m}^{n,HL}(1-x_f) \quad \geq 1 \quad \forall m \in \{1,\dots,n^{Kl,HL}\} \quad (3)$$

$$d_{l,j,k}^{MS} \in \{0,1\} \quad \forall l \in \{1,\dots,n^L\},\ \forall j \in \{1,\dots,n^{VR}\},\ \forall k \in \{j+1,\dots,n^{VR}\} \quad (4)$$

$$d^{Dist} \in \mathbb{N} \quad (5)$$

$$x_f \in \{0,1\} \quad \forall f \in \{1,\dots,n^F\} \quad (6)$$

Der Term $\sum_{i\in\{1,\dots,n^L\}}\sum_{j\in\{1,\dots,n^{VR}\}}\sum_{k\in\{j+1,\dots,n^{VR}\}} r_{l,j,k}^{MS} d_{l,j,k}^{MS}$ der **Zielfunktion** gibt die gewichtete Anzahl von Separationen für die gewählte Variante an. Er wird normiert, indem er durch die Anzahl maximal möglicher Separationen geteilt wird. Ebenso wird der Abstand d^{Dist} normiert, indem er durch den maximal möglichen Abstand n^F geteilt wird. Durch den Gewichtungsfaktor w^{MS} wird eine Gewichtung der beiden Kriterien vorgenommen. Dieser Gewichtungsfaktor ist experimentell zu bestimmen. **Nebenbedingung 1** stellt sicher, dass $d_{l,j,k}^{MS}$ nur auf 1 gesetzt werden darf, wenn tatsächlich MSF jk in VR l erfüllt ist. Hierfür wird jede Klausel der MSF in eine Dual-Rail-Darstellung überführt. Der Term $\sum_{f\in\{1,\dots,n^F\}} w_{f,l,j,k,m}^{p,MS} x_f$ gibt an, an wie vielen Stellen f die Klausel m ein positives Literal enthält und x_f wahr ist. Gilt dies für eine der Stellen, akzeptiert die Klausel die Variante. Dasselbe gilt für den Term $\sum_{f\in\{1,\dots,n^F\}} w_{f,l,j,k,m}^{n,MS}(1-x_f)$ und negative Literale. Damit $d_{l,j,k}^{MS}$ auf 1 gesetzt werden kann, müssen alle Klauseln der MSF die Variante akzeptieren. **Nebenbedingung 2** beschränkt d^{Dist} jeweils durch den Abstand zu den Datenpunkten im Datensatz, d. h. d^{Dist} kann nicht größer gewählt werden, als der geringste Abstand der gewählten Variante zu Datenpunkten im Datensatz. Die Linke Seite gibt die Anzahl von Stellen an, in denen die gewählte Variante und eine bestehende Variante nicht übereinstimmen. **Nebenbedingung 3** stellt sicher, dass die gewählte Variante alle Klauseln der HLF erfüllt und nutzt hierfür ebenfalls eine Dual-Rail-Darstellung. Die **Nebenbedingungen 4, 5 und 6** ergeben sich aus der Definition der Variablen. In Anhang A6 wird auf zwei Aspekte der hier gewählten Modellierung, den Verzicht auf eine dynamische Gewichtung der Kriterien sowie die zu wählende Codierung, eingegangen.

Abbildung 4.35 zeigt das Optimierungsproblem zur Auswahl einer Variante für den Beispielfall sowie die zugehörige optimale Lösung. Dabei wird beispielhaft davon ausgegangen, dass im HLKM die Beschränkung $\neg x_2 \vee \neg x_3$ gilt, d. h. x_2 und x_3 nicht zugleich gewählt werden können. Nach Schritt 2 liegt eine Variante mit potenziell hohem Informationsgehalt für die datenbasierte Erstellung von Regeln vor.

$$\max \quad w^{MS}\frac{1}{4}\left(\frac{1}{2}d^{MS}_{1,1,2}+\frac{1}{2}d^{MS}_{1,1,3}+\frac{1}{2}d^{MS}_{1,2,3}+\frac{1}{2}d^{MS}_{2,1,2}+\frac{1}{2}d^{MS}_{2,1,3}+\frac{1}{2}d^{MS}_{2,2,3}\right) + (1-w^{MS})\frac{1}{4}d^{dist}$$

$$s.t. \quad \begin{array}{rcll} (1-x_2)+(1-x_3) & \geq & 1 & \text{HLKM} \\ (1-x_1) & \geq & d^{MS}_{1,1,2} & \\ x_3 & \geq & d^{MS}_{1,1,2} & B^{MSF}_{1,1,2} \\ (1-x_4) & \geq & d^{MS}_{1,1,2} & \\ \ldots & & \ldots & \\ x_1+x_2+x_3+x_4 & \geq & d^{Dist} & \text{Instanz 1} \\ \ldots & & & \ldots \\ d^{MS}_{1,1,2}, d^{MS}_{1,1,3}, d^{MS}_{1,2,3}, d^{MS}_{2,1,2}, d^{MS}_{2,1,3}, d^{MS}_{2,2,3} & \in & \{0,1\} & \\ d^{Dist} & \in & \mathbb{N} & \\ x_1, x_2, x_3, x_4 & \in & \{0,1\} & \end{array}$$

HLKM = High-Level-Konfigurationsmodell

Aus HLKM:
$\sim x_2 \vee \sim x_3$

Optimale Lösung für $w^{MS} = 0{,}5$:

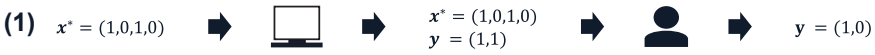
$\boldsymbol{x}^* = (1,0,1,0)$

$\boldsymbol{d}^{MS^*} = (0,0,0,0,1,1)$

$d^{Dist^*} = 2$

Abbildung 4.35: Optimierungsproblem zur Auswahl einer Variante sowie zugehörige optimale Lösung

Wie in Kapitel 4.1.3 beschrieben, kann diese Variante – unterstützt durch das LLKM – mit einer VSTL und VAPLs versehen und für das **Training eines neuen, genaueren LLKM** genutzt werden. Abbildung 4.36 zeigt schematisch die Erstellung einer VSTL und eines VAPL als Labels des neuen Datenpunkts (1), das Hinzufügen des neuen, annotierten Datenpunkts zum Trainingsdatensatz (2) sowie die Aktualisierung der VRs



(2)

Features				Labels	
x_1	x_2	x_3	x_4	y_1	y_2
0	0	0	0	0	0
0	1	0	0	0	0
…					
1	0	1	0	1	0

VR = Versionenraum

(3)

VR zu y_1

$B_{1,1} = x_1 \vee x_3$
$B'_{1,2} = x_1 \vee (x_3 \wedge x_4)$
$B'_{1,3} = x_3 \vee (x_1 \wedge x_4)$

VR zu y_2

~~$B_{2,1} = x_3$~~
$B'_{2,1} = x_3 \wedge \sim x_1$
$B'_{2,2} = x_3 \wedge x_4$
~~$B'_{2,3} = x_3 \wedge \sim x_2$~~
$B'_{2,3} = x_3 \wedge x_4 \wedge \sim x_1$

Abbildung 4.36: Abschluss einer Iteration durch die Erstellung von variantenbezogenen Stücklisten und Arbeitsplänen (1) sowie Überführung in den Datensatz (2) und Beginn einer neuen Iteration mit der Aktualisierung der Versionenräume (3)

in der nächsten Iteration (3) für den Beispielfall. Wie die Abbildung zeigt, können durch den neuen Datenpunkt zwei Modelle aus VR 2 ausgeschlossen und durch neue Modelle ersetzt werden.

Methode 5 schließt die im Rahmen der vorliegenden Arbeit entwickelte Methodik zur datenbasierten Erstellung von LLKM ab. Die Im folgenden vorgestellte Methode 6 ist der datenbasierten Überprüfung von LLKM zuzuordnen.

4.6 Methode 6: Datenbasierte Überprüfung von Regeln

Im Folgenden wird die im Rahmen der vorliegenden Arbeit entwickelte Methode 6 zur Lösung des Problems 6 – der datenbasierten Überprüfung von Regeln – vorgestellt. Methode 6 behebt das in Kapitel 3.6.3 beschriebene Lösungsdefizit nach Stand der Forschung. Die Methode kann auf bestehende LLKMs angewandt werden, unabhängig davon, ob diese manuell oder datenbasiert, wie in den vorangegangenen Kapiteln beschrieben, erstellt wurden.

Wie in Kapitel 3.6.2 dargestellt, existieren mit der statischen Verifikation und dem empirischen Testen bereits zwei Ansätze zur Überprüfung von KMs, die auch zur Überprüfung von LLKMs eingesetzt werden können. Methode 6 komplementiert diese Ansätze, indem sie Anomalieerkennung im Sinne des UL nutzt, um Hinweise auf Fehler in LLKMs zu finden. Wie in Kapitel 3.5.2 begründet, ist für eine ganzheitliche Überprüfung eine Kombination dieser Methode mit den bestehenden Ansätzen sinnvoll. Zunächst sollte eine statische Verifikation durchgeführt werden, da diese eine beschränkte Anzahl starker Hinweise auf Fehler gibt, die immer vollständig überprüft werden sollten. Sowohl die im Folgenden vorgestellte Methode zur Anomalieerkennung als auch das empirische Testen sind beliebig skalierbar. Die Frequenz, mit der jeweils Fehler gefunden werden, nimmt mit steigender Anzahl von Überprüfungen ab. Für die Anomalieerkennung besteht, wie im Folgenden gezeigt wird, ein Indikator dafür, wann eine weitere Überprüfung ineffizient wird. Damit ist es möglich, eine Anomalieerkennung bis zu diesem Punkt durchzuführen und anschließend bei Bedarf mit empirischem Testen fortzufahren.

Die **grundlegende Idee** der im Folgenden vorgestellten Methode zur Anomalieerkennung für LLKMs besteht darin, eine Menge S^{Reg} von Regeln eines LLKM in einen **nicht-annotierten Datensatz** zu überführen. Dieser kann anschließend genutzt werden, um darin mit **Verfahren des UL** Anomalien zu ermitteln (siehe Kapitel 2.3.3). Hieraus können Rückschlüsse auf potenzielle Fehler in den Regeln aus S^{Reg} gezogen werden. Es

wird von Regeln des LLKM in Form von booleschen Ausdrücken ausgegangen, deren Wahrheitswert den Wahrheitswert einer abhängigen booleschen Variablen bestimmt, d. h. von Regeln, wie sie für Auswahlbedingungen in LLKMs genutzt werden (siehe Kapitel 2.2.2.4). Für das UL werden abhängigkeitsbasierte Ansätze der Anomalieerkennung (siehe Kapitel 2.3.3) verwendet, um Muster in den Features des Datensatzes zu erkennen und daraus auf Anomalien zu schließen. Abbildung 4.37 gibt einen Überblick über die fünf Schritte der Methode 6. In Schritt 1 werden die Regeln aus S^{Reg} in eine tabellarische Form gebracht, die als Datensatz für UL geeignet ist. In Schritt 2 wird eines der Features des Datensatzes für die Überprüfung ausgewählt. In Schritt 3 werden SL-Modelle trainiert, um einen Zusammenhang zwischen den anderen Features des Datensatzes dem ausgewählten Feature zu ermitteln[56]. In Schritt 4 werden auf Basis der Güte des ermittelten Zusammenhangs Anomalien im Datensatz bewertet und Alternativvorschläge berechnet. Die Schritte 2 bis 4 werden für jedes Feature des Datensatzes wiederholt. In Schritt 5 werden die im Datensatz erkannten Anomalien auf die Regeln aus S^{Reg} abgebildet und dem Anwender für eine systematische Überprüfung der Regeln zur Verfügung gestellt. Die Schritte der Methode werden im Folgenden erläutert.

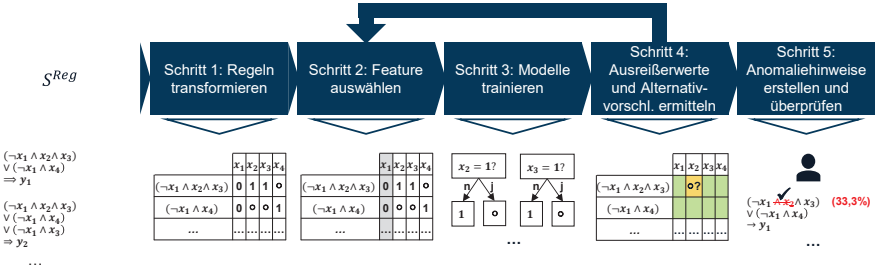


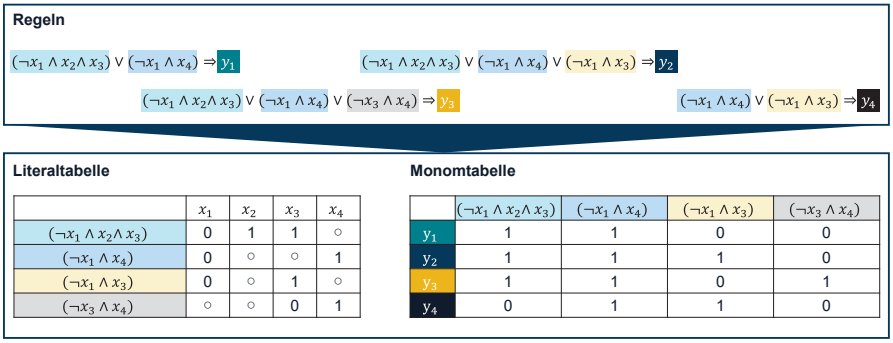
Abbildung 4.37: Überblick über die Schritte der Methode 6

4.6.1 Schritt 1: Regeln transformieren

Damit die Regeln aus S^{Reg} mit Methoden des UL verarbeitet werden können, müssen sie in eine geeignete, tabellarische Form gebracht werden. Regeln in DNF lassen sich

[56] Das Problem wird – wie bei abhängigkeitsbasierten Ansätzen des UL üblich – auf Teilprobleme zurückgeführt, die dem SL zuzuordnen sind. Auch wenn hierfür Features temporär als Labels interpretiert werden, liegen im Trainingsdatensatz selbst keine Labels vor. Das Problem ist somit dem UL zuzuordnen (siehe Kapitel 2.3.1).

in eine **Literaltabelle** und eine **Monomtabelle** überführen, wie in Abbildung 4.38 exemplarisch dargestellt.



	x_1	x_2	x_3	x_4
$(\neg x_1 \wedge x_2 \wedge x_3)$	0	1	1	○
$(\neg x_1 \wedge x_4)$	0	○	○	1
$(\neg x_1 \wedge x_3)$	0	○	1	○
$(\neg x_3 \wedge x_4)$	○	○	0	1

	$(\neg x_1 \wedge x_2 \wedge x_3)$	$(\neg x_1 \wedge x_4)$	$(\neg x_1 \wedge x_3)$	$(\neg x_3 \wedge x_4)$
y_1	1	1	0	0
y_2	1	1	1	0
y_3	1	1	0	1
y_4	0	1	1	0

Abbildung 4.38: Überführung von Regeln in eine tabellarische Form für einen Beispielfall

Die Literaltabelle gibt für jedes Monom an, ob eine Variable als **positives Literal (1)**, **negatives Literal (0)** oder **überhaupt nicht (○)** auftritt. Die Monomtabelle gibt an, ob ein Monom in einer Regel auftritt (1) oder nicht (0). Literal- und Monomtabelle enthalten alle Informationen über eine Menge S^{Reg} von Regeln. Beide Tabellen lassen sich jeweils als Datensätze im Sinne des ML interpretieren. Die Zeilen der Tabellen, d. h. die Monome bzw. Regeln, entsprechen Datenpunkten und die Spalten, d. h. die Variablen bzw. Monomen, entsprechen Features im Sinne des ML[57]. Labels liegen im Datensatz zunächst nicht vor. Zum Teil nutzen Unternehmen solche oder ähnliche Tabellen, um Regeln manuell in KMs einzugeben. Diese können neben Regeln in DNF auch Regeln in KNF oder in anderen Formen (siehe Anhang A12.1) repräsentieren. Ggf. können die bereits existierenden Tabellen unmittelbar verwendet werden. Die folgende Erklärung beschränkt sich auf Regeln in DNF. Die Anwendung auf andere Darstellungsformen kann jedoch analog erfolgen.

UL-Verfahren zur Anomalieerkennung werden eingesetzt, um anomale Datenpunkte zu ermitteln (siehe Kapitel 2.3.3). Ein Datenpunkt besteht aus einem Eintrag je Feature.

[57] Datensätze die für die datenbasierte Überprüfung von Regeln verwendet werden sind nicht mit Datensätzen zu verwechseln, die für die datenbasierte Erstellung von Konfigurationsmodellen verwendet werden. Diese weisen einen anderen Aufbau und insbesondere andere Arten von Datenpunkten und Features auf (siehe Kapitel 4.1 und Kapitel 4.4).

Anomalien bestimmten Einträgen von Datenpunkten zuzuordnen, ist in vielen Ansätzen in der Literatur nicht das Ziel. Ohne eine solche Zuordnung zu Einträgen entsteht für den vorliegenden Fall jedoch ein hoher Überprüfungsaufwand, weil Datenpunkte in den betrachteten Tabellen für komplexe KMs über viele Einträge verfügen können. Um den manuellen Überprüfungsaufwand gering zu halten, wird deshalb eine Methode der Anomalieerkennung benötigt, durch die Anomalien in Einträgen von Datenpunkten ermittelt werden können. Hierfür sind prinzipiell Ansätze der **abhängigkeitsbasierte**n **Anomalieerkennung** geeignet[58]. Dabei wird je Feature ein Modell trainiert, das den Zusammenhang zwischen den anderen Features und dem betrachteten Feature im Datensatz wiedergibt. Einträge des betrachteten Features werden jeweils mit einem sog. **Ausreißerwert** (engl. Outlier Score) versehen, der umso höher ist, je weniger sie dem ermittelten Zusammenhang entsprechen. Die Ausreißerwerte werden über alle Einträge eines Datenpunkts aggregiert. Auf diese Weise wird jedem Datenpunkt ein Ausreißerwert zugeordnet. Damit stellt die Ermittlung von anomalen Einträgen ein Zwischenergebnis dieser Ansätze dar. **Bestehende Ansätze** sind jedoch entweder für Regressionsprobleme entwickelt worden (siehe Li & van Leeuwen 2023, Xie et al. 2021, Lu et al. 2020, Paulheim & Meusel 2015, Wagstaff et al. 2013) oder setzen voraus, dass ein Teil des Datensatzes als fehlerfrei bekannt ist (siehe Noto et al. 2010). Ersteres entspricht nicht dem vorliegenden Fall, weil ein Klassifikationsproblem vorliegt. Zweiteres kann für Anwendungsfälle in der Industrie im Allgemeinen nicht vorausgesetzt werden[59]. Im Folgenden wird deshalb eine im Rahmen der vorliegenden Arbeit entwickelte Methode des UL vorgestellt, die das Vorgehen der genannten Arbeiten mit einem Ansatz von Sluban et al. (2010) zur Ermittlung von Rauschen in Labels für annotierte Datensätze verbindet.

4.6.2 Schritt 2: Feature auswählen

Wie beschrieben, wird je Feature ein Modell trainiert. Deshalb sind die Features nacheinander auszuwählen. Die Reihenfolge, in der die Features betrachtet werden, ist nicht relevant. Gewählt wird in jeder Iteration ein Feature der betrachteten Tabelle, das zuvor

[58] Die vom Autor der vorliegenden Arbeit angeleitete Bachelorarbeit A_Kahn (2023) baut auf Frey et al. (2023) auf und beschreibt erstmals die Verwendung abhängigkeitsbasierter Verfahren des UL für diesen Zweck. Die hier vorgestellte Methode unterscheidet sich jedoch von A_Kahn (2023). Es werden nicht Ausreißer absolut bestimmt, sondern es wird jedem Eintrag ein Ausreißerwert zugeordnet (siehe Kapitel 4.6.4). Dies bildet die Basis für eine skalierbare Überprüfung der Regeln (siehe Kapitel 4.6.5). Dieses abweichende Ziel begründet auch die Verwendung eines anderen Verfahrens des SL (siehe Kapitel 4.6.3), das es ermöglicht Ausreißerwerte zu berechnen.

[59] Es kann Fälle geben, in denen ein Teil der Regeln als korrekt angenommen werden kann und ein anderer Teil überprüft werden soll, z. B. wenn das LLKM nach einiger Zeit erweitert wird. An dieser Stelle wird jedoch nur der allgemeine Anwendungsfall betrachtet.

noch nicht betrachtet worden ist. Wurden bereits alle Features betrachtet, wird die Methode mit Schritt 5 fortgesetzt. Die Einträge des in einer Iteration gewählten Features werden in dieser Iteration als **Labels des Datensatzes** betrachtet. Somit liegt ein annotierter Datensatz vor und es können SL-Modelle trainiert werden, um den Zusammenhang zwischen den anderen Features und dem Label, d. h. dem ausgewählten Feature darzustellen. Abbildung 4.39 zeigt die iterative Auswahl von Features für die Literaltabelle des Beispielfalls.

Literaltabelle	x_1 ✓	x_2 ✓	x_3 ✓	x_4
$(\neg x_1 \wedge x_2 \wedge x_3)$	0	1	1	○
$(\neg x_1 \wedge x_4)$	0	○	○	1
$(\neg x_1 \wedge x_3)$	0	○	1	○
$(\neg x_3 \wedge x_4)$	○	○	0	1

Abbildung 4.39 Iterative Auswahl von Features

4.6.3 Schritt 3: Modelle trainieren

Nachdem ein Feature ausgewählt wurde liegt ein annotierter Datensatz vor, wobei die Ausprägungen des ausgewählten Feature die Labels der Datenpunkte darstellen. Der Ansatz von Sluban et al. (2010) kann nun eingesetzt werden um Rauschen in diesen Labels zu ermitteln. Dadurch können Anomalien in den Einträgen der initialen Tabelle ermittelt werden, die zu dem ausgewählten Feature gehören[60]. Um Rauschen in Labels zu ermitteln, trainieren Sluban et al. (2010) für das Label des Datensatzes ein **Random-Forest-Modell**. Dabei handelt es sich um ein Ensemble von Entscheidungsbäumen, wobei jeder Entscheidungsbaum von der Realisation einer oder mehrerer Zufallsvariablen abhängt (Breiman 2001, S. 5). Sluban et al. (2010) nutzen die Random-Forest-Implementierung der Orange Data Mining-Bibliothek[61] (Sluban et al. 2014, S. 271). Diese erstellt Entscheidungsbäume nach der von Breiman (2001, S. 10–11) beschriebenen Methode, wonach beim Training der Entscheidungsbäume zufällige Features zum Bestimmen eines Splits gleichmäßig verteilt ausgewählt werden. Rauschen

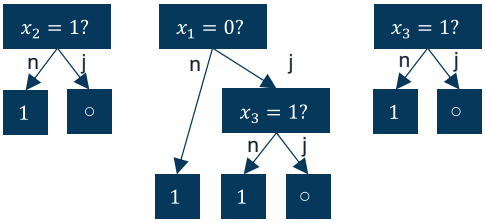


Abbildung 4.40: Ensemble von Entscheidungsbäumen für Feature x_4 des Beispielfalls

[60] Der Ansatz von Sluban et al. (2010) wurde zuvor in der vom Autor der vorliegenden Arbeit angeleiteten Bachelorarbeit A_Zacateco Herrera (2023) zur Überprüfung von variantenbezogenen Stücklisten und Arbeitsplänen eingesetzt – ein Anwendungsfall, der in der vorliegenden Arbeit nicht betrachtet wird.

[61] Siehe: https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/randomforest.html (zuletzt überprüft am 07.06.2025)

im Label ist nach Sluban et al. (2010, S. 1105) für diejenigen Datenpunkte zu vermuten, für die die Vorhersagen der Entscheidungsbäume heterogen sind, d. h. für die eine hohe Vorhersageunsicherheit besteht und damit kein klares Muster erkennbar ist. Sluban et al. (2010) zeigen, dass Random-Forest-Modelle für Klassifikationsprobleme besser geeignet sind, um Rauschen in Labels zu identifizieren als andere Ensemble-Methoden. Deshalb werden sie auch in der vorliegenden Arbeit eingesetzt. Es wird ein Ensemble von Entscheidungsbäumen trainiert, die das ausgewählte Feature auf Basis der anderen Features vorhersagen. Abbildung 4.40 zeigt ein beispielhaftes Ensemble zur Vorhersage von Feature x_4 und damit die vierte Iteration des Beispielfalls. Auf Basis des Ensembles können für die Einträge des ausgewählten Features Ausreißerwerte und Alternativvorschläge ermittelt werden.

4.6.4 Schritt 4: Ausreißerwerte und Alternativvorschläge ermitteln

Analog zu Sluban et al. (2010) wird je Label, d. h. Eintrag des betrachteten Features, ein Ausreißerwert bestimmt, der die **Vorhersageunsicherheit** des Random-Forests für jeden Datenpunkt des Datensatzes angibt. Die Vorhersageunsicherheit für einen Datenpunkt ergibt sich als Anteil der Entscheidungsbäume, die das falsche Label vorhersagen. Die am häufigsten vorhergesagte Klasse, die nicht der tatsächlichen Ausprägung des Labels entspricht, stellt den Alternativvorschlag zur Ausprägung des Labels dar. Die Unsicherheit von Datenpunkt $(\neg x_1 \wedge x_3)$ in Abbildung 4.41 (1) beträgt z. B. 33,3 %, weil von drei Entscheidungsbäumen im Random-Forest eine Vorhersage nicht der tatsächlichen Ausprägung des Labels entspricht. Der **Alternativvorschlag** ist die am häufigsten vorhergesagte nichtzutreffende Klasse „1“ (2). Die Unsicherheiten je Eintrag

(1)

Berechnung	x_1	x_2	x_3	Label	DT1	DT2	DT3	Uns.
$(\neg x_1 \wedge x_2 \wedge x_3)$	0	1	1	○	○	○	○	0%
$(\neg x_1 \wedge x_4)$	0	○	○	1	1	1	1	0%
$(\neg x_1 \wedge x_3)$	0	○	1	○	1	○	○	33,3%
$(\neg x_3 \wedge x_4)$	○	○	0	1	1	1	1	0%

(2)

Ausreißerwerte	x_1	x_2	x_3	x_4
$(\neg x_1 \wedge x_2 \wedge x_3)$	0%	33,3%	0%	**0%**
$(\neg x_1 \wedge x_4)$	0%	0%	0%	**0%**
$(\neg x_1 \wedge x_3)$	0%	0%	0%	**33,3%**
$(\neg x_3 \wedge x_4)$	66,6%	0%	0%	**0%**

Alternativvorschläge	x_1	x_2	x_3	x_4
$(\neg x_1 \wedge x_2 \wedge x_3)$		○		
$(\neg x_1 \wedge x_4)$				
$(\neg x_1 \wedge x_3)$				**1**
$(\neg x_3 \wedge x_4)$	1			

Abbildung 4.41: Berechnung von Ausreißerwerten und Alternativvorschlägen für Feature x_4 des Beispielfalls. Die Spalten DT (Decision Tree) entsprechen den Vorhersagen der Entscheidungsbäume des Ensembles.

sowie die Alternativvorschläge werden über alle Features hinweg in einer Tabelle $T^{Ausreißer}$ bzw. einer Tabelle $T^{Alternativ}$ gespeichert. Die beiden Tabellen werden abschließend genutzt, um Anomaliehinweise zu erstellen.

4.6.5 Schritt 5: Anomaliehinweise erstellen und überprüfen

Auf Basis der Tabellen $T^{Ausreißer}$ und $T^{Alternativ}$ werden sukzessive Anomaliehinweise generiert. Dafür wird derjenige noch nicht betrachtete Eintrag in $T^{Ausreißer}$ mit dem höchsten Wert ausgewählt. Diesem Eintrag steht ein Eintrag der Literaltabelle bzw. Monomtabelle gegenüber, der wiederum mit einem Literal in einem Monom bzw. einem Monom in den Regeln korrespondiert. Ein hoher Ausreißerwert ist ein Hinweis auf einen potenziellen Fehler in diesem Literal bzw. diesem Monom. Wird über die Literaltabelle ein Literal eines bestimmten Monoms als Anomalie identifiziert, sind davon alle Vorkommen dieses Monoms in den Regeln betroffen. Wird über die Monomtabelle ein Monom einer bestimmten Regel als Anomalie identifiziert, sind davon alle Literale des bestimmten Monoms in der bestimmten Regel betroffen. Durch $T^{Alternativ}$ ist eine potenziell korrekte Formulierung bekannt. Beides kann zusammen, wie in Abbildung 4.42 dargestellt, zur Überprüfung an einen **Domänenexperte**n gegeben werden.

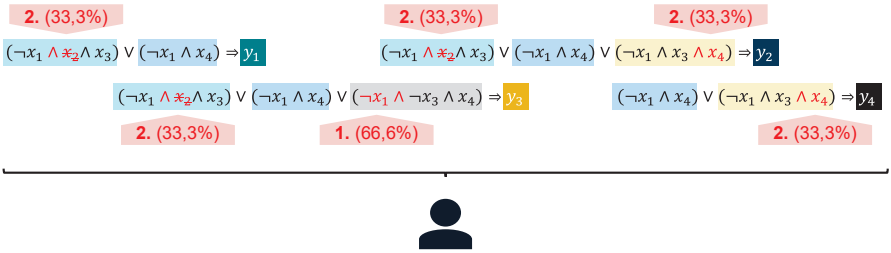


Abbildung 4.42: Darstellung von Anomaliehinweisen zur Überprüfung durch einen Domänenexperten

Die Erstellung und Überprüfung von Hinweisen wird so lange fortgesetzt, bis ein **Abbruchkriterium** erreicht ist. Wie in Kapitel 5.6.2 gezeigt wird, besteht zunächst ein ungefähr linearer Zusammenhang zwischen der Anzahl von Überprüfungen und der Anzahl gefundener Fehler. Dieser lineare Zusammenhang geht ab einer gewissen Anzahl gefundener Fehler in einen exponentiellen Zusammenhang über. Sobald also der Nutzer bei der Durchführung von Schritt 5 eine Abweichung von einem linearen Zusammenhang erkennt, wird Schritt 5 abgebrochen. Bei Bedarf wird mit empirischem Testen

fortgefahren. Neben einer Einschätzung durch den Nutzer können hierfür auch statistische Verfahren eingesetzt werden.

5 Demonstration

Im Folgenden werden die in Kapitel 4 entwickelten Methoden 1 bis 6 demonstriert. Dies entspricht dem vierten Schritt des Design Science Research Process (DSRP, siehe Kapitel 1.3). Es werden Metriken für die Demonstration eingeführt, um die Effektivität der Methoden zu quantifizieren. Die Ergebnisse hinsichtlich dieser Metriken dienen als Grundlage für eine abschließende Bewertung der entwickelten Methoden in Kapitel 6. Im Folgenden wird zunächst das Vorgehen der Demonstration im Allgemeinen sowie der hierfür verwendete Anwendungsfall beschrieben (Kapitel 5.1). In den darauffolgenden Unterkapiteln wird das Vorgehen zur Demonstration der Methoden 2 bis 6 genauer beschrieben und es werden die Ergebnisse der Demonstration präsentiert (Kapitel 5.2 bis 5.6).

Methode 1 integriert die Methoden 2 bis 5 und ist genau dann in der Lage Problem 1 zu lösen, wenn die Methoden 2 bis 5 in der Lage sind, die jeweiligen Probleme 2 bis 5 zu lösen. Die Effektivität der Methode 1 ergibt sich aus der Effektivität der Methoden 2 bis 5. Deshalb wird im Folgenden auf ein separates Unterkapitel zur Demonstration von Methode 1 verzichtet.

5.1 Anwendungsfall und Vorgehen der Demonstration

Zur Demonstration der im Rahmen der vorliegenden Arbeit entwickelten Methoden 4, 5 und 6 wurden **Konfigurationsmodelle** (KMs) **eines Industriepartners** verwendet. Der Industriepartner ist ein international tätiger Anbieter von Messgeräten für die Prozessindustrie und setzt bereits seit langem Konfigurationssysteme (KSs) im Rahmen der Auftragsabwicklung ein. Insbesondere werden diese für die Erstellung von variantenbezogenen Stücklisten (VSTLs) und variantenbezogenen Arbeitsplänen (VAPLs) genutzt. Verwendet werden beschränkungsbasierte High-Level-Konfigurationsmodelle (HLKMs) und regelbasierte Low-Level-Konfigurationsmodelle (LLKMs) im Rahmen von SAP LO-VC, wie in Kapitel 2.2.2.4 beschrieben. Das Unternehmen hat für die vorliegende Arbeit HLKMs und LLKMs zu 7 verschiedenen konfigurierbaren Produkten, d. h. 7 Produktfamilien, zur Verfügung gestellt. Um mit vertretbarem Rechenaufwand eine umfassende Demonstration durchführen zu können, wurden hieraus drei Produkte ausgewählt, welche im Folgenden als Produkte A, B und C bezeichnet werden. Dabei handelt es sich um **Messgeräte zur Absolut- und Differenzdruckmessung**. Diese unterscheiden sich in ihren physikalischen Messprinzipien und dadurch auch in ihren Stücklisten (STLs) und Arbeitsplänen (APLs) maßgeblich, so dass sie ein breites Spektrum

an Varianten abbilden. Für die Demonstration im engeren Sinne wurden ausschließlich die Produkte A und B verwendet, wohingegen Produkt C für Voruntersuchungen, wie z. B. Parameterstudien genutzt wurde. Die KMs wurden für die Demonstration durch eigens entwickelte Programme emuliert, so dass sich daraus VSTLs und VAPLs konfigurieren lassen.

Die Produkte verfügen zum einen über **kategorische Merkmale**, wie z. B. Zulassung, Druckart oder elektrischer Anschluss. Für diese Merkmale ist genau eine Ausprägung zu wählen. Ein Spezialfall dieser Merkmale stellen Merkmale dar, für die darüber hinaus auch keiner der Werte gewählt werden kann, was letztlich als weitere mögliche Ausprägung verstanden werden kann. Zum anderen verfügen die Produkte über **mehrwertige Merkmale**, für die mehrere Ausprägungen gewählt werden können, wie z. B. Zubehör oder Kennzeichnungen des Produkts. Diese Merkmale wurden als Menge von booleschen Merkmalen interpretiert, die jeweils gewählt oder nicht gewählt werden können. Für die Demonstration der Methoden 4 und 5 wurden die Produktmerkmale per One-Hot-Codierung codiert. Für die Demonstration der Methode 6 wurde eine in Kapitel 5.6 beschriebene Transformation der KMs durchgeführt.

Es gibt in den KMs des Industriepartners keine Unterscheidung zwischen Produktmerkmalen im HLKM und den Parametern des Endprodukts im LLKM. Das HLKM enthält ausschließlich paarweise Ausschlussbeziehungen zwischen Ausprägungen verschiedener Merkmale und keine Einschlussbeziehungen. Die **Maximalstücklisten** (MSTLs) aller betrachteten Produkte sind **einstufig**, d. h. alle Komponenten gehen unmittelbar in das Endprodukt ein. Entsprechend liegt auch nur **ein Maximalarbeitsplan** (MAPL) vor. Der MAPL ist linear, d. h. enthält keine Parallelitäten. Das LLKM enthält eine Regel für jede Komponente der MSTL und jeden Arbeitsvorgang (AVO) des MAPL, wobei jedes der Elemente ausschließlich von den Produktmerkmalen abhängig ist. Tabelle 5.1 gibt eine Übersicht über die Kennzahlen der betrachteten KMs. Die Variantenanzahlen unter Berücksichtigung von Beschränkungen wurden im Rahmen der vorliegenden Arbeit erstmals fundiert approximiert, worauf Anhang A7.1 eingeht. Es zeigte sich, dass insbesondere Konfigurationsmodell B mit über 10^{22} Varianten einen Konfigurationsraum aufweist, der nahe an die größten in der Literatur beschriebenen Konfigurationsräume heranreicht (siehe Kapitel 1.1). Es weist darüber hinaus mehr Komponentenklassen (KKs) in der MSTL und mehr Arbeitsvorgangsklassen (AVKs) im MAPL als die anderen betrachteten Produkte auf. In der MSTL und im MAPL des Industriepartners liegen Standardkomponenten bzw. Standardarbeitsvorgänge vor, die in jeder

Variante auftreten. Diese sind immer aktiv, d. h. die Regel ihres Aktivitätszustands entspricht trivial dem Wahrheitswert wahr. Tabelle 5.1 zeigt den Anteil trivialer Regeln für die MSTL und den MAPL. Darüber hinaus zeigt sie die Komplexität der Regeln je Konfigurationsmodell, gemessen an der mittleren Anzahl enthaltener Monome in disjunktiver Normalform (DNF). Insbesondere für die Produkte B und C existieren einige wenige Regeln mit herausragender Komplexität. Zur besseren Einschätzung ist in Tabelle 5.1 deshalb auch die mittlere Komplexität der Regeln ohne Berücksichtigung der komplexesten 5% der Regeln angegeben. Es kann festgehalten werden, dass die mittleren Komplexitäten der Regeln für Produkt B deutlich über denen von Produkt A liegen. Für alle drei Produkte ist außerdem die Komplexität der Regeln der MSTL höher als die der Regeln des MAPL. Dies lässt sich u. a. dadurch begründen, dass es Merkmale gibt, die keinen Einfluss auf die VSTL, jedoch auf den VAPL haben (siehe hierzu auch Kapitel 2.2.2.3).

Tabelle 5.1: Kennzahlen der betrachteten Konfigurationsmodelle des Industriepartners

		Produkt		
Kategorie	**Kennzahl**	**A**	**B**	**C**
HLKM	Anzahl boolescher Merkmale	71	82	68
	Anzahl kategorischer Merkmale	20	25	16
	Anzahl der Varianten ohne Berücksichtigung von Beschränkungen	$1{,}0 * 10^{35}$	$4{,}8 * 10^{43}$	$7{,}4 * 10^{33}$
	Approximierte Anzahl der Varianten unter Berücksichtigung von Beschränkungen	$1{,}4 * 10^{18}$	$5{,}8 * 10^{22}$	$1{,}3 * 10^{18}$
LLKM, MSTL	Anzahl der KKs in der MSTL	690	1659	656
	Anteil trivialer Regeln	49,7%	30,6%	41,5%
	Mittlere Anzahl der Monome in den Regeln in DNF, ohne triviale Regeln	3,5	64,2	48,0
	Mittlere Anzahl der Monome in den Regeln in DNF, ohne triviale Regeln, ohne obere 5%	2,9	4,8	6,3
LLKM, MAPL	Anzahl der AVKs im MAPL	162	330	165
	Anteil trivialer Regeln	26,5%	23,3%	24,2%
	Mittlere Anzahl der Monome in den Regeln in DNF, ohne triviale Regeln	7,6	163,4	98,1
	Mittlere Anzahl der Monome in den Regeln in DNF, ohne triviale Regeln, ohne obere 5%	5,3	20,7	12,1

Die Grenzen des Anwendungsfalls für die Demonstration der im Rahmen der vorliegenden Arbeit entwickelten Methoden liegen in den einfachen Strukturen der MSTL und

des MAPL. Dadurch lässt sich am Anwendungsfall nicht umfänglich demonstrieren, dass mit den entwickelten Methoden 2 und 3 auch allgemeine MSTLs und MAPLs datenbasiert erstellt werden können. Um übertragbare Erkenntnisse zu ermöglichen, werden deshalb zur Demonstration der Methoden 2 und 3 gleichmäßig zufällig erstellte **synthetische MSTLs und MAPLs** verwendet. Darauf wird in den entsprechenden Unterkapiteln eingegangen.

Die beschriebenen Methoden 2 bis 6 wurden im Rahmen der vorliegenden Arbeit vollständig in **Programmcode** implementiert, welcher die Grundlage der Demonstration bildet und online verfügbar ist[62]. Damit können alle Ergebnisse, soweit sie nicht auf vertraulichen Daten basieren, nachvollzogen werden. Alle Experimente für die im Folgenden Rechenzeiten oder Zeitbeschränkungen angegeben werden, wurden auf Rechensystemen mit 16 Prozessorkernen, mit einer jeweiligen Taktrate von 2,4 Gigahertz und mit 32 Gigabyte Arbeitsspeicher durchgeführt.

5.2 Methode 2: Datenbasierte Erstellung von Maximalstücklisten

Im Folgenden wird die Demonstration der Methode 2 beschrieben. In Anhang A8.4 finden sich darüber hinaus Zeitstudien, die den Effekt der in Kapitel 4.2.1 vorgestellten Funktionen zur Erhöhung der Recheneffizienz von Schritt 1 der Methode 2 quantifizieren. Sie zeigen, dass die im Rahmen der vorliegenden Arbeit entwickelten Funktionen maßgeblich zur Recheneffizienz von Methode 2 beitragen.

5.2.1 Vorgehen

Die grundlegende Idee der Demonstration besteht darin, eine synthetische MSTL als Referenz zu erstellen, aus dieser MSTL eine Menge S^{VSTL} von VSTLs zu konfigurieren und auf Basis von S^{VSTL} mittels Methode 2 die **Referenz-MSTL** zu **rekonstruieren**. Die Referenz-MSTL repräsentiert die für einen praktischen Anwendungsfall zu erstellende, korrekte MSTL. Die Menge S^{VSTL} repräsentiert die in einem Anwendungsfall vorliegenden VSTLs. Durch die Demonstration wird überprüft, ob Methode 2 grundsätzlich in der Lage ist, eine geeignete MSTL aus S^{VSTL} zu erstellen und wie effektiv dies in Abhängigkeit der Kardinalität n^{VSTL} von S^{VSTL} möglich ist. Um die Effektivität quantifizieren zu können, wird die relative Abweichung zwischen der durch Methode 2 erstellten Ergebnis-MSTL und der Referenz-MSTL nach der in Kapitel A8.3 definierten, normierten

[62] https://github.com/alexmfrey/creation_and_validation_of_configuration_models.git (zuletzt überprüft am 07.06.2025)

Metrik $d^{MSTL} \in [0,1]$ ermittelt. Diese quantifiziert die Unterschiedlichkeit der Baugruppen der Ergebnis- und der Referenz-MSTL.

Es ist davon auszugehen, dass bestimmte Eigenschaften der Referenz-MSTL Auswirkungen auf die Effektivität der Methode 2 haben. Deshalb muss eine systematische Verzerrung bei der **Erzeugung der Referenz-MSTL** vermieden werden. Im Rahmen der vorliegenden Arbeit wurde deshalb eine Methode zur Erzeugung von MSTLs entwickelt, die MSTLs aus dem Raum aller möglichen MSTLs mit bestimmten Eigenschaften gleichmäßig zufällig auswählt. Die Methode ist in Anhang A8.1 beschrieben. Es wird ferner davon ausgegangen, dass die Anzahl möglicher Zukaufkomponenten (ZKs) mit identischer Bezeichnung in einer VSTL unabhängig von der gültigen Struktur ist (siehe Kapitel 4.2.2.2), d. h. alle Strukturoptionen (STOs) der Referenz-MSTL enthalten dieselbe Anzahl von Zukaufkomponentenklassen (ZKKs). Außerdem werden nur MSTLs betrachtet, deren ZKKs entweder genau einer STO oder allen STOs zugeordnet sind[63].

Die **Parameter der Erstellung** sind

- r^{BGK}, das Verhältnis der Anzahl von Baugruppenklassen (BGKs) zu ZKKs in der Referenz-MSTL,
- n^{STO}, die Anzahl der STOs in der Referenz-MSTL,
- n^{ZKK}, die Anzahl der ZKKs je STO,
- r^{Mult}, der Anteil mehrfach auftretender ZKK-Bezeichnungen je STO
- und r^{Abh}, der Anteil von ZKKs der Referenz-MSTL, die von der gültigen STO abhängen.

Aus einer Referenz-MSTL können, wie im Folgenden beschrieben, VSTLs konfiguriert werden. Sind für die MSTL STOs hinterlegt, wird zunächst die gültige STO zufällig gewählt. Es wird berücksichtigt, dass die STOs nicht dieselben Auftretenswahrscheinlichkeiten haben müssen. Hierfür wird zunächst jeder STO gleichmäßig zufällig ein Wert aus dem Intervall $0{,}5 \pm r^{RadSTO}$ mit $r^{RadSTO} \in [0; 0{,}5)$ zugewiesen. Diese Werte werden anschließend normiert, so dass sich die Wahrscheinlichkeitsmasse zu 1 kumuliert. Je größer r^{RadSTO}, desto höher ist somit die Varianz der Auftretenswahrscheinlichkeiten. Die gültige STO wird je konfigurierter VSTL entsprechend dieser Wahrscheinlichkeitsverteilung gewählt. r^{RadSTO} ist ein Parameter der Demonstrationsexperimente.

[63] Wie in Anhang AA8.1 erläutert, ist diese Einschränkung notwendig um eine gleichmäßig zufällige Auswahl zu gewährleisten.

Nachdem die STO feststeht, wird für jede ZKK, die unter dieser STO instanziiert werden kann, zufällig bestimmt, ob diese instanziiert wird. Auch hier werden unterschiedliche Auftretenswahrscheinlichkeiten der zugehörigen ZKKs berücksichtigt, indem den ZKKs gleichmäßig zufällig Wahrscheinlichkeiten aus dem Intervall $0{,}5 \pm r^{RadZKK}$ mit $r^{RadZKK} \in [0; 0{,}5]$ zugeordnet und normiert werden. r^{RadZKK} ist ebenfalls ein Parameter der Demonstrationsexperimente. Die **Parameter eines Experiments** sind damit n^{VSTL}, r^{BGK}, n^{STO}, n^{ZKK}, r^{Mult}, r^{Abh}, r^{RadSTO} und r^{RadZKK}.

Das Vorhandensein von Multipositionen und STOs wirkt sich erheblich auf die Rechenzeit der Experimente aus und schränkt damit die Anzahl in vertretbarer Zeit durchführbarer Experimente ein. Es werden deshalb zum einen Experimente für Referenz-MSTL ohne Multipositionen und ohne STOs, d. h. mit $n^{STO} = 1$, $r^{Mult} = 0$, $r^{Abh} = 0$ und $r^{RadSTO} = 0$ durchgeführt. Dabei wird ein **Referenzfall** mit einem definierten Anteil von BGKs von $r^{BGK} = 0{,}5$ und einem definierten Radius der Auftretenswahrscheinlichkeiten der Komponenten von $r^{RadZKK} = 0{,}25$ gewählt. Die beiden Parameter werden gegenüber diesem Referenzfall variiert, um ihren Einfluss auf d^{MSTL} zu ermitteln. Zum anderen werden für den ausgewählten Referenzfall MSTLs mit Multipositionen und STOs untersucht, um wiederum den Einfluss dieser Aspekte zu bestimmen. Jedes Experiment wird in **zehn Durchläufe**n wiederholt. Sowohl für Schritt 1 als auch für Schritt 2 der Methode 2 wird eine **Zeitbeschränkung von 1800 Sekunden** definiert.

5.2.2 Ergebnisse

5.2.2.1 Experimente an Referenz-Maximalstücklisten ohne Multipositionen und Strukturoptionen

Abbildung 5.1 zeigt die jeweils über 10 Durchläufe gemittelten Ergebnisse für d^{MSTL} sowie die Korrelationskoeffizienten ausgewählter Parameter der Experimente in Prozent. Für den **Referenzfall** (Abbildung 5.1, 2) mit $r^{Mult} = 0{,}5$ und $r^{RadZKK} = 0{,}25$ ergeben sich zunächst für $n^{VSTL} = 10$ Abweichungen zwischen Ergebnis- und Referenz-MSTL von bis zu 19,2 %. Bei geringen Datenmengen hängt das Ergebnis in hohem Maße von den zufällig ausgewählten Datenpunkten ab, weshalb für die verschiedenen Ausprägungen von n^{ZKK} unsystematische Streuungen auftreten. Bereits ab $n^{VSTL} = 40$ sind die Abweichungen zwischen Ergebnis- und Referenz-MSTL weitgehend vernachlässigbar, d. h. die Referenz-MSTL kann robust rekonstruiert werden. Für $n^{VSTL} = 180$ und $n^{VSTL} = 200$ in Verbindung mit $n^{ZKK} = 100$, d. h. eine große Menge an VSTLs aus

einer großen Referenz-MSTL, treten allerdings erneut Abweichungen auf. Das ist auf die vorgegebene Zeitbeschränkung für die Schritte 1 und 2 der Methode 2 zurückzuführen. Für $n^{VSTL} = 200$ wurde in 6 der 10 Durchläufe mindestens einer der beiden Zeitbeschränkungen erreicht. Eine genauere Analyse dieser Fälle zeigte, dass aufgrund des vorzeitigen Abbruchs jeweils eine MSTL mit höherer Komplexität als die Referenz-MSTL erstellt wurde.

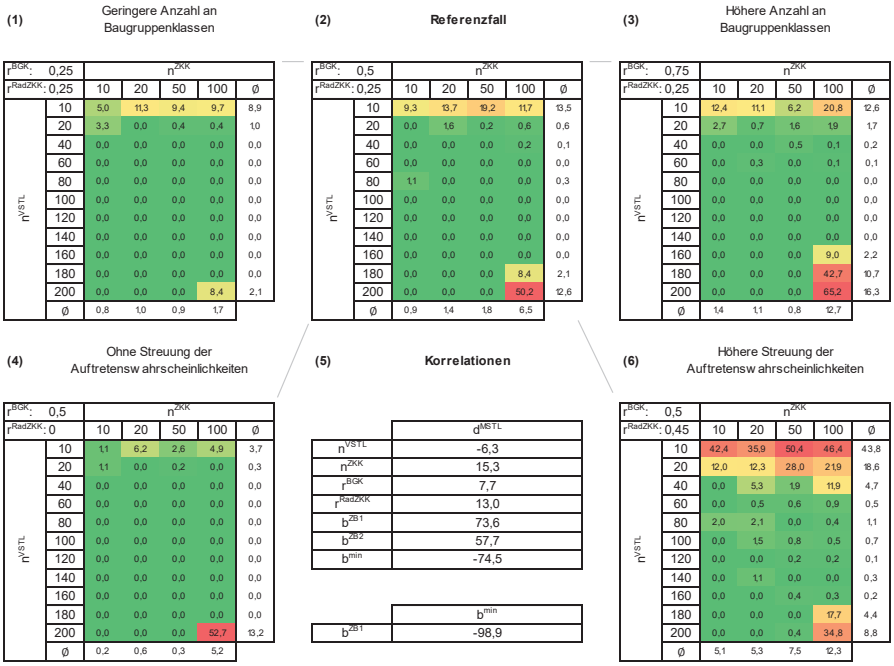


Abbildung 5.1: Ergebnisse der Demonstration der Methode 2 an Referenz-Maximalstücklisten ohne Multipositionen und Strukturoptionen; alle Angaben in Prozent

Die Ergebnisse für geringere (Abbildung 5.1, 1) und höhere (Abbildung 5.1, 3) **Anzahlen von BGKs** sind mit denen für den Referenzfall vergleichbar. Der Effekt der Zeitbeschränkung wirkt sich jedoch aufgrund des weniger komplexen bzw. komplexeren Problems weniger stark bzw. stärker aus. Die Ergebnisse für die Fälle ohne und mit höheren **Streuungen der Auftretenswahrscheinlichkeiten von ZKKs** (Abbildung 5.1, 4 bzw. 6) unterscheiden sich hingegen maßgeblich von denen für den Referenzfall. Für $r^{RadZKK} = 0$ liegt die Abweichung d^{MSTL} bereits für $n^{VSTL} = 10$ unabhängig von n^{ZKK}

unterhalb von 10 %. Für $r^{RadZKK} = 0{,}45$ treten hingegen bis zu $n^{VSTL} = 40$ noch Abweichungen von mehr als 10 % auf. Bei großen Streuungen der Auftretenswahrscheinlichkeiten kann es Komponenten geben, die in den VSTLs aus S^{VSTL} nur gering repräsentiert sind. Dies erschwert es diese Komponenten mittels Methode 2 korrekt in die MSTL einzuordnen.

In Abbildung 5.1 (5) sind die **Korrelationskoeffizienten** der Parameter des Experiments sowie der Variablen b^{ZB1}, b^{ZB2} und b^{min} zu sehen. Die Variablen geben an, ob bei der Durchführung von Schritt 1 oder Schritt 2 der Methode 2 die **Zeitbeschränkung** erreicht wurde bzw. ob eine MSTL mit einer minimalen Komplexität gefunden wurde[64]. Das Auftreten komplexitätsminimaler MSTLs ist stark negativ mit d^{MSTL} korreliert, d. h. die Abweichung zur Referenz-MSTL ist tendenziell für komplexitätsminimale MSTLs geringer als für nichtkomplexitätsminimale. Dies spricht für die Wahl dieses Optimierungskriteriums. Der vorzeitige Abbruch von Schritt 2 ist in allen Fällen eine Konsequenz des vorzeitigen Abbruchs von Schritt 1. Liegt keine komplexitätsminimale MSTL vor, kann das in Schritt 2 aufgestellte Optimierungsproblem sehr groß werden, was zu einem Abbruch nach Überschreitung der Zeitbeschränkung führt. Deshalb ist es plausibel, dass nicht nur b^{ZB1} und d^{MSTL}, sondern auch b^{ZB2} und d^{MSTL} stark positiv korreliert sind. Zuletzt bestätigen die Korrelationskoeffizienten den oben beschriebenen Einfluss von r^{RadZKK} und implizieren, dass d^{MSTL} mit zunehmender Größe n^{ZKK} der MSTL zu- und mit zunehmender Anzahl n^{VSTL} an vorliegenden VSTLs abnimmt.

5.2.2.2 Experimente an Referenz-Maximalstücklisten mit Multipositionen oder Strukturoptionen

Abbildung 5.2 zeigt die jeweils über 10 Durchläufe gemittelten Ergebnisse für d^{MSTL} in Prozent sowie die Korrelationskoeffizienten ausgewählter Parameter der Experimente in Prozent. Liegen ausschließlich **Multipositionen und keine STOs** vor (Abbildung 5.2, 1), können kleine MSTLs mit $n^{ZKK} = 20$ bereits ab $n^{VSTL} = 20$ genau reproduziert werden. Ab $n^{VSTL} = 100$ ergeben sich jedoch wieder schlechtere Ergebnisse, was auf den in Kapitel 5.2.2.1 beschriebenen **Effekt der Zeitbeschränkung** zurückzuführen ist. Mit zunehmender Größe der Datenmenge nimmt die Abweichung d^{MSTL} also zunächst aufgrund abnehmender Varianz im Sinne des maschinellen Lernens (ML) ab. Ab einer

[64] Im Falle eines vorzeitigen Abbruchs wird angenommen, dass eine minimale Komplexität vorliegt, wenn diese nicht höher ist als die der Referenz-MSTL.

gewissen Größe der Datenmenge wird dieser Effekt jedoch durch den vorzeitigen Abbruch der Schritte 1 und 2 der Methode 2 überkompensiert. Dadurch ergibt sich ein **u-förmiger Verlauf** von d^{MSTL} in Abhängigkeit von n^{VSTL}. Dieser ist auch für $n^{ZKK} = 100$ und für die über alle n^{ZKK} gemittelten Randwerte deutlich zu erkennen. Experimente mit Multipositionen weisen tendenziell einen höheren Rechenaufwand als Experimente ohne Multipositionen auf. Dadurch kann erklärt werden, dass der Effekt der Zeitbeschränkung stärker ausgeprägt ist als für die in Kapitel 5.2.2.1 beschriebenen Experimente.

(1) Multipositionen

r^{ZKMult} 0,1 / r^{AbhZKK} 0 / n^{STO} 1 / r^{RadSTO} 0	n^{ZKK} 20	50	100	Ø
n^{VSTL} 10	8,1	23,2	32,2	21,2
20	0,0	11,3	18,6	10,0
50	0,0	6,3	5,4	3,9
100	1,0	3,6	8,7	4,4
200	8,0	3,7	61,7	24,5
Ø	3,4	9,6	25,3	

(2) STO ohne Streuung der Auftretenswahrscheinlichkeiten

r^{ZKMult} 0 / r^{AbhZKK} 0,1 / n^{STO} 2 / r^{RadSTO} 0	n^{ZKK} 10	20	50	100	Ø
n^{VSTL} 10	14,7	15,2	24,0	33,3	21,8
20	7,0	8,1	9,9	11,5	9,1
50	0,0	6,9	2,5	5,8	3,8
100	6,1	4,0	24,2	20,3	13,7
200	4,0	24,1	38,7	72,9	34,9
Ø	6,4	11,7	19,9	28,8	

(3) Multipositionen und STO ohne Streuung der Auftretenswahrscheinlichkeiten

r^{ZKMult} 0,1 / r^{AbhZKK} 0,1 / n^{STO} 2 / r^{RadSTO} 0	n^{ZKK} 20	50	100	Ø
n^{VSTL} 10	28,2	30,8	41,1	33,4
20	4,9	20,2	28,4	17,8
50	10,3	13,4	15,4	13,0
100	15,4	23,0	33,3	23,9
200	13,7	44,7	80,8	46,4
Ø	14,5	26,4	39,8	

(4) Korrelationen

	d^{MSTL}
n^{VSTL}	19,8
n^{ZKK}	34,9
r^{ZKMult}	18,4
r^{AbhZKK}	26,0
r^{AbhZKK}	26,0
r^{RadSTO}	14,7
b^{ZB1}	42,5
b^{ZB2}	64,7
b^{min}	-58,3

	b^{Min}
b^{ZB1}	-48,0

(5) STO mit Streuung der Auftretenswahrscheinlichkeiten

r^{ZKMult} 0 / r^{AbhZKK} 0,1 / n^{STO} 2 / r^{RadSTO} 0,25	n^{ZKK} 10	20	50	100	Ø
n^{VSTL} 10	36,2	21,1	28,5	34,6	30,1
20	4,6	6,7	13,3	19,0	10,9
50	12,8	3,1	10,2	8,1	8,6
100	1,3	12,8	12,5	13,9	10,1
200	1,0	25,6	26,9	76,1	32,4
Ø	11,2	13,9	18,3	30,4	

(6) Multipositionen und STO mit Streuung der Auftretenswahrscheinlichkeiten

r^{ZKMult} 0,1 / r^{AbhZKK} 0,1 / n^{STO} 2 / r^{RadSTO} 0,25	n^{ZKK} 20	50	100	Ø
n^{VSTL} 10	22,1	41,9	42,5	35,5
20	9,7	30,9	32,7	24,4
50	1,7	19,4	19,1	13,4
100	21,4	23,6	20,4	21,8
200	18,2	41,8	61,6	40,6
Ø	14,6	31,5	35,3	

Abbildung 5.2: Ergebnisse der Demonstration der Methode 2 an Referenz-Maximalstücklisten mit Multipositionen oder Strukturoptionen; alle Angaben in Prozent

Der beschriebene u-förmige Verlauf von d^{MSTL} prägt aus demselben Grund auch die Ergebnisse der Experimente **ohne Multipositionen, aber mit STOs** (Abbildung 5.2, 2 und 5). Wie zu erwarten, ergeben sich für größere Streuungen der Auftretenswahrscheinlichkeiten der STOs größere Abweichungen, da in den Daten unterrepräsentierte

STOs auftreten können. Auch in Fällen mit STOs existieren Fenster für n^{VSTL} und n^{ZKK}, für die eine datenbasierte Erstellung von MSTLs mit hoher Zuverlässigkeit möglich ist. Diese beschränken sich jedoch im Wesentlichen auf kleine MSTLs mit $n^{ZKK} = 10$. Die Experimente an Referenz-MSTLs mit Multipositionen und STOs (Abbildung 5.2, 3 und 6) zeigen, dass solche MSTLs mit den gegebenen **Zeitbeschränkungen** im Allgemeinen nicht zuverlässig datenbasiert erstellt werden können. Da auch hier jedoch ein **u-förmiger Verlauf** von d^{MSTL} erkennbar ist, ist zu vermuten, dass im Falle längerer Rechenzeiten oder höherer Rechenleistung mit ausreichender Datenmenge ebenfalls MSTLs mit geringen Abweichungen erstellt werden können. Dies war jedoch im Rahmen der vorliegenden Arbeit nicht überprüfbar.

Die **Korrelationstabelle** (Abbildung 5.2, 4) bestätigt den großen Einfluss der Zeitbeschränkung auf die Ergebnisse. b^{ZB1} ist stark negativ mit b^{min} korreliert[65]. b^{min} wiederum ist stark positiv mit d^{MSTL} korreliert. Zu dem Zeitpunkt, an dem die Zeitbeschränkung von Schritt 1 erreicht wird, liegt also in vielen Fällen noch keine minimale MSTL vor. Ist die Ergebnis-MSTL nicht minimal, ist ihre Abweichung von der Referenz-MSTL tendenziell groß. Die Auswirkung dessen auf die Ergebnisse ist groß, weil in ca. 52 % der Experimente die Zeitbeschränkung in Schritt 1 erreicht wurde.

5.3 Methode 3: Datenbasierte Erstellung von Maximalarbeitsplänen

Im Folgenden wird die Demonstration der Methode 3 beschrieben. In Anhang A9.3 finden sich darüber hinaus Zeitstudien, die den Effekt der in Kapitel 4.3.1 vorgestellten Funktionen zur Erhöhung der Recheneffizienz von Schritt 1 der Methode 3 quantifizieren. Sie zeigen, dass die im Rahmen der vorliegenden Arbeit entwickelten Funktionen maßgeblich zur Recheneffizienz von Methode 3 beitragen.

5.3.1 Vorgehen

Der Demonstration der Methode 3 liegt dieselbe Idee zugrunde, wie der Demonstration der Methode 2. Mithilfe einer Methode, die im Rahmen der vorliegenden Arbeit entwickelt wurde (siehe Anhang A9.2), werden gleichmäßig zufällig **Referenz-MAPL** erstellt. Aus diesen wird eine Menge S^{VAPL} von VAPLs konfiguriert, die als Eingabe für die

[65] $b^{ZB2} = 1$ ist i. d. R. eine Folge von $b^{min} = 0$, da im Falle nicht minimaler MSTL das in Schritt 2 zu lösende Optimierungsproblem sehr groß werden kann. Dies erklärt, warum auch b^{ZB2} und b^{min} stark negativ korreliert sind.

Methode 3 zur datenbasierten Erstellung eines Ergebnis-MAPL dient. Zur Bewertung der Effektivität von Methode 3 in Abhängigkeit der Kardinalität n^{VAPL} von S^{VAPL} wird die relative Abweichung zwischen der Ergebnis- und der Referenz-MSTL nach der in Anhang A9.1 definierten, normierten **Metrik** $d^{MAPL} \in [0{,}1]$ bestimmt. Diese quantifiziert die Unterschiedlichkeit der Vorrangbeziehungen des Ergebnis- und des Referenz-MAPL.

Es wird analog zur Erstellung von Referenz-MSTLs davon ausgegangen, dass die Anzahl möglicher AVOs mit identischer Bezeichnung in einem VAPL unabhängig von der gültigen Struktur ist, d. h. alle STOs des Referenz-MAPL enthalten dieselbe Anzahl von AVKs. Außerdem werden keine MAPLs mit AVKs, die mehreren, aber nicht allen STOs des MAPL zugeordnet sind, betrachtet. Die **Parameter der Erstellung** sind

- r^{VBZ}, der Anteil der gültigen Vorrangbeziehungen an allen möglichen Vorrangbeziehungen zwischen den AVOs des Referenz-MAPL,
- n^{STO}, die Anzahl der möglichen Strukturen der aus dem MAPL konfigurierbaren VAPL,
- n^{AK}, die Anzahl der AVKs je STO des Referenz-MAPL,
- r^{Mult}, der Anteil mehrfach auftretender AVK-Bezeichnungen je STO
- und r^{Abh}, der Anteil von AVKs eines Referenz-MAPL, die von der gültigen STO abhängen.

Die zufällige Konfiguration von VAPLs aus dem Referenz-MAPL erfolgt analog zur zufälligen Konfiguration von VSTLs aus MSTLs, wie in Kapitel 5.2.1 beschrieben. Dabei werden ebenfalls ungleiche Auftretenswahrscheinlichkeiten der STOs über den Parameter $r^{RadSTO} \in [0; 0{,}5]$ und ungleiche Auftretenswahrscheinlichkeiten der AVKs über den Parameter $r^{RadAK} \in [0; 0{,}5]$ berücksichtigt. Es liegen im VAPL alle Vorrangbeziehungen des MAPL vor, die zwischen instanziierten AVKs bestehen. Die **Parameter eines Experiments** sind n^{VAPL}, r^{VBZ}, n^{STO}, n^{AK}, r^{Mult}, r^{Abh}, r^{RadSTO} und r^{RadAK}.

Analog zur Demonstration der Methode 2 werden zum einen Experimente mit Referenz-MAPL ohne Multipositionen und STOs durchgeführt und zum anderen Experimente für solche mit Multipositionen oder STOs für einen Referenzfall mit $r^{VBZ} = 0{,}5$ und $r^{RadZKK} = 0{,}25$. Jedes Experiment wird in **zehn Durchläufe**n wiederholt. Sowohl für Schritt 1 als auch für Schritt 2 der Methode 3 wird eine **Zeitbeschränkung von 1800 Sekunden** definiert.

5.3.2 Ergebnisse

5.3.2.1 Experimente an Referenz-Maximalarbeitsplänen ohne Multipositionen und Strukturoptionen

Abbildung 5.3 zeigt die jeweils über 10 Durchläufe gemittelten Ergebnisse für d^{MAPL} sowie die Korrelationskoeffizienten ausgewählter Parameter der Experimente in Prozent. Für den **Referenzfall** (Abbildung 5.3, 2) mit $r^{VBZ} = 0{,}5$ und $r^{RadAK} = 0{,}25$ treten bereits ab $n^{VAPL} = 40$ unabhängig von der Größe n^{AK} des Referenz-MAPL keine wesentlichen Abweichungen zwischen Ergebnis- und Referenz-MAPL auf. Lediglich für $n^{VAPL} = 200$ und $n^{AK} = 100$ zeigt sich der in Kapitel 5.2.2.1 beschriebene Effekt der Zeitbeschränkung. Mit der Verringerung (Abbildung 5.3, 1) oder Erhöhung (Abbildung 5.3, 3) der Anzahl von Vorrangbeziehungen fällt d^{MAPL} ab bzw. steigt an, dieser Effekt ist jedoch gering.

(1) Geringere Anzahl an Vorrangbeziehungen

r^{VBZ}: 0,25; r^{RadAK}: 0,25

n^{VAPL}	10	20	50	100	Ø
10	6,5	17,8	6,4	10,7	10,4
20	0,0	0,5	0,6	0,5	0,4
40	0,0	0,0	0,0	0,0	0,0
60	0,0	0,0	0,0	0,0	0,0
80	0,0	0,0	0,0	0,0	0,0
100	0,0	0,0	0,0	0,0	0,0
120	0,0	0,0	0,0	0,0	0,0
140	0,0	0,0	0,0	0,0	0,0
160	0,0	0,0	0,0	0,0	0,0
180	0,0	0,0	0,0	0,0	0,0
200	0,0	0,0	0,0	5,7	1,4
Ø	0,6	1,7	0,6	1,5	

(2) Referenzfall

r^{VBZ}: 0,5; r^{RadAK}: 0,25

n^{VAPL}	10	20	50	100	Ø
10	1,4	3,9	20,4	7,0	8,2
20	2,1	2,8	1,5	1,1	1,9
40	0,5	0,0	0,0	0,0	0,1
60	0,0	0,0	0,0	0,0	0,0
80	0,0	0,0	0,0	0,0	0,0
100	0,0	0,0	0,0	0,0	0,0
120	0,0	0,0	0,0	0,0	0,0
140	0,0	0,0	0,0	0,0	0,0
160	0,0	0,0	0,0	0,0	0,0
180	0,0	0,0	0,0	0,0	0,0
200	0,0	0,0	0,0	9,0	2,3
Ø	0,4	0,6	2,0	1,6	

(3) Höhere Anzahl an Vorrangbeziehungen

r^{VBZ}: 0,75; r^{RadAK}: 0,25

n^{VAPL}	10	20	50	100	Ø
10	4,9	7,1	5,8	8,7	6,6
20	1,8	1,2	1,1	1,7	1,5
40	0,7	0,0	0,0	0,1	0,2
60	0,0	0,0	0,0	0,0	0,0
80	0,0	0,0	0,0	0,0	0,0
100	0,0	0,0	0,0	0,0	0,0
120	0,0	0,0	0,1	0,0	0,0
140	0,0	0,0	0,0	0,0	0,0
160	0,0	0,0	0,0	3,1	0,8
180	0,0	0,0	0,0	9,4	2,4
200	0,0	0,0	0,0	18,5	4,6
Ø	0,7	0,8	0,6	3,8	

(4) Ohne Streuung der Auftretenswahrscheinlichkeiten

r^{VBZ}: 0,5; r^{RadAK}: 0

n^{VAPL}	10	20	50	100	Ø
10	3,8	3,6	1,7	5,3	3,6
20	0,5	0,0	0,1	0,2	0,2
40	0,0	0,0	0,0	0,0	0,0
60	0,0	0,0	0,0	0,0	0,0
80	0,0	0,0	0,0	0,0	0,0
100	0,0	0,0	0,0	0,0	0,0
120	0,0	0,0	0,0	0,0	0,0
140	0,0	0,0	0,0	0,0	0,0
160	0,0	0,0	0,0	0,0	0,0
180	0,0	0,0	0,0	0,0	0,0
200	0,0	0,0	0,0	18,2	4,6
Ø	0,4	0,3	0,2	2,2	

(5) Korrelationen

	d^{MAPL}
n^{VAPL}	-14,8
n^{AK}	7,8
r^{VBZ}	3,1
r^{RadAK}	15,5
b^{ZB1}	59,1
b^{ZBZ}	46,6
b^{min}	-62,2

	b^{min}
b^{ZB1}	46,6

(6) Höhere Streuung der Auftretenswahrscheinlichkeiten

r^{VBZ}: 0,5; r^{RadAK}: 0,45

n^{VAPL}	10	20	50	100	Ø
10	23,6	19,5	26,6	34,4	26,0
20	25,5	14,2	10,5	12,2	15,6
40	8,0	8,6	8,0	4,2	7,2
60	1,6	1,6	1,9	1,1	1,5
80	1,2	0,6	1,0	0,7	0,9
100	1,2	0,2	0,2	0,8	0,6
120	1,8	0,1	0,5	0,3	0,7
140	0,0	0,3	0,4	0,2	0,2
160	1,1	0,0	0,1	0,4	0,4
180	0,0	0,2	0,3	0,2	0,1
200	0,0	0,1	0,0	18,4	4,6
Ø	5,8	4,1	4,5	6,6	

Abbildung 5.3: Ergebnisse der Demonstration der Methode 3 an Referenz-Maximalarbeitsplänen ohne Multipositionen und Strukturoptionen; alle Angaben in Prozent

Analog zu den in Kapitel 5.2.2.1 beschriebenen Ergebnissen für die Demonstration der Methode 2 hat die Streuung der Auftretenswahrscheinlichkeiten der AVKs eine nennenswerte Auswirkung auf d^{MAPL} (Abbildung 5.3, 4 und 6), was durch die **Korrelationskoeffizienten** (Abbildung 5.3, 5) bestätigt wird. Ebenfalls analog zeigt sich darüber hinaus eine hohe positive Korrelation zwischen b^{ZB1} und d^{MAPL} sowie b^{ZB2} und d^{MAPL} und eine hohe negative Korrelation zwischen b^{min} und d^{MAPL}.

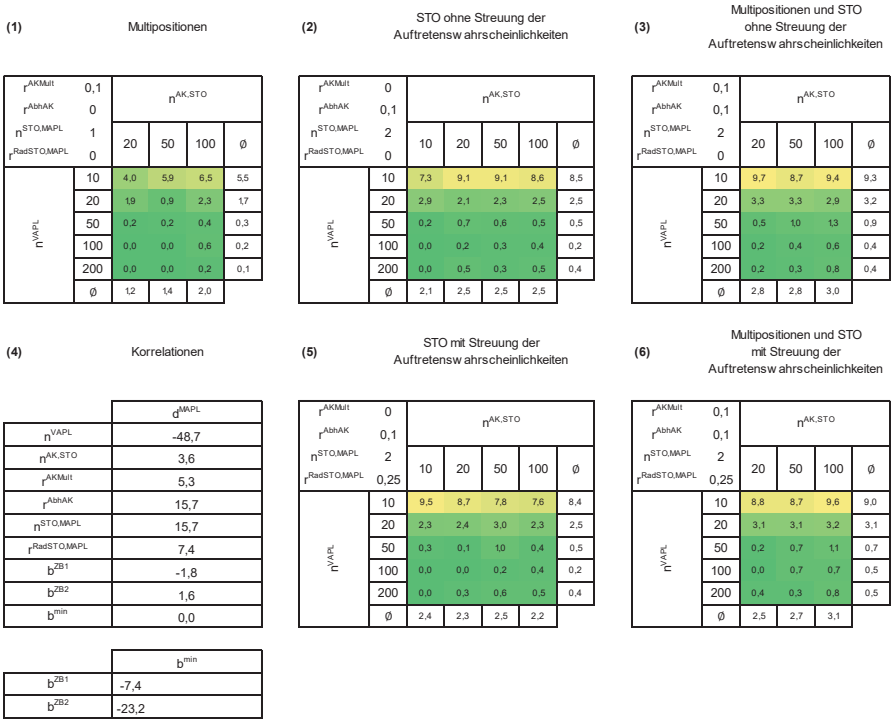
5.3.2.2 Experimente an Referenz-Maximalarbeitsplänen mit Multipositionen oder Strukturoptionen

Abbildung 5.4 zeigt die jeweils über 10 Durchläufe gemittelten Ergebnisse für d^{MAPL} sowie die Korrelationskoeffizienten ausgewählter Parameter der Experimente in Prozent.

Liegen ausschließlich **Multipositionen und keine STOs** vor (Abbildung 5.4, 1), können MSTLs unabhängig von ihrer Größe ab $n^{VSTL} = 50$ mit Abweichungen von weniger als 1 % reproduziert werden. Dasselbe gilt im Wesentlichen auch für den Fall, dass **STOs aber keine Multipositionen** vorliegen (Abbildung 5.4, 2 und 5). Hierbei hat die Streuung der Auftretenswahrscheinlichkeit nur einen geringfügigen Einfluss.

Insgesamt zeigt sich, dass unabhängig von Multipositionen und STOs gute Ergebnisse erzielt werden können. Dies steht im Gegensatz zu den entsprechenden in 5.2.2.2 dargestellten Ergebnissen für Methode 2. Dieser Gegensatz lässt sich dadurch erklären, dass gegenüber Methode 2 die **Zeitbeschränkungen für Methode 3 einen geringeren Einfluss** auf die Ergebnisse haben. Insgesamt wurde Schritt 1 in ca. 51 % der durchgeführten Experimente vorzeitig abgebrochen, was in etwa der entsprechenden Abbruchrate für die Demonstration der Methode 2 entspricht. Lediglich in ca. 0,6 % der durchgeführten Experimente wurde allerdings kein komplexitätsminimaler MAPL ermittelt. Dies zeigt sich auch in der geringen Korrelation zwischen b^{ZB1} und b^{min} (Abbildung 5.4, 4)[66]. Wird Schritt 1 vorzeitig abgebrochen, liegt also i. d. R. bereits eine optimale Lösung vor. Dies steht in einem Gegensatz zu den in Kapitel 5.2.2.2 gezeigten Ergebnissen für die Demonstration der Methode 2; für Methode 2 geht ein Abbruch tendenziell mit suboptimalen MSTLs einher. Schritt 1 der Methode 3 ist also wesentlich

[66] Da vorzeitige Abbrüche von Schritt 2 i. d. R. das Resultat sehr komplexer MAPL aus Schritt 1 sind und hier i. d. R. komplexitätsminimale MAPL gefunden werden, traten auch nur in ca. 0,3 % der Experimente vorzeitige Abbrüche von Schritt 2 auf. Die zu b^{ZB2} gehörigen Korrelationskoeffizienten haben deshalb nur eine geringe Aussagekraft, weshalb hierauf nicht eingegangen wird.



r^{AKMult}	0,1
r^{AbhAK}	0
$n^{STO,MAPL}$	1
$r^{RadSTO,MAPL}$	0

n^{VAPL}	$n^{AK,STO}$ 20	50	100	Ø
10	4,0	5,9	6,5	5,5
20	1,9	0,9	2,3	1,7
50	0,2	0,2	0,4	0,3
100	0,0	0,0	0,6	0,2
200	0,0	0,0	0,2	0,1
Ø	1,2	1,4	2,0	

r^{AKMult}	0
r^{AbhAK}	0,1
$n^{STO,MAPL}$	2
$r^{RadSTO,MAPL}$	0

n^{VAPL}	$n^{AK,STO}$ 10	20	50	100	Ø
10	7,3	9,1	9,1	8,6	8,5
20	2,9	2,1	2,3	2,5	2,5
50	0,2	0,7	0,6	0,5	0,5
100	0,0	0,2	0,3	0,4	0,2
200	0,0	0,5	0,3	0,5	0,4
Ø	2,1	2,5	2,5	2,5	

r^{AKMult}	0,1
r^{AbhAK}	0,1
$n^{STO,MAPL}$	2
$r^{RadSTO,MAPL}$	0

n^{VAPL}	$n^{AK,STO}$ 20	50	100	Ø
10	9,7	8,7	9,4	9,3
20	3,3	3,3	2,9	3,2
50	0,5	1,0	1,3	0,9
100	0,2	0,4	0,6	0,4
200	0,2	0,3	0,8	0,4
Ø	2,8	2,8	3,0	

	d^{MAPL}
n^{VAPL}	-48,7
$n^{AK,STO}$	3,6
r^{AKMult}	5,3
r^{AbhAK}	15,7
$n^{STO,MAPL}$	15,7
$r^{RadSTO,MAPL}$	7,4
b^{ZB1}	-1,8
b^{ZB2}	1,6
b^{min}	0,0

	b^{min}
b^{ZB1}	-7,4
b^{ZB2}	-23,2

r^{AKMult}	0
r^{AbhAK}	0,1
$n^{STO,MAPL}$	2
$r^{RadSTO,MAPL}$	0,25

n^{VAPL}	$n^{AK,STO}$ 10	20	50	100	Ø
10	9,5	8,7	7,8	7,6	8,4
20	2,3	2,4	3,0	2,3	2,5
50	0,3	0,1	1,0	0,4	0,5
100	0,0	0,0	0,2	0,4	0,2
200	0,0	0,3	0,6	0,5	0,4
Ø	2,4	2,3	2,5	2,2	

r^{AKMult}	0,1
r^{AbhAK}	0,1
$n^{STO,MAPL}$	2
$r^{RadSTO,MAPL}$	0,25

n^{VAPL}	$n^{AK,STO}$ 20	50	100	Ø
10	8,8	8,7	9,6	9,0
20	3,1	3,1	3,2	3,1
50	0,2	0,7	1,1	0,7
100	0,0	0,7	0,7	0,5
200	0,4	0,3	0,8	0,5
Ø	2,5	2,7	3,1	

Abbildung 5.4: Ergebnisse der Demonstration der Methode 3 an Referenz-Maximalarbeitsplänen mit Multipositionen oder Strukturoptionen; alle Angaben in Prozent.

effizienter darin, komplexitätsminimale MAPLs zu finden als Schritt 1 der Methode 2 komplexitätsminimale MSTLs zu finden.

5.4 Methode 4: Datenbasierte Erstellung von Regeln

In diesem Kapitel wird die Demonstration der Methode 4 beschrieben. Über die Demonstration hinaus wurden zwei Benchmarkstudien durchgeführt, auf die an dieser Stelle kurz eingegangen wird. In der **ersten Benchmarkstudie**, die in Anhang A10.3 ausführlich beschrieben wird, wurde Methode 4 mit der Methode Two Stage von Ignatiev et al. (2021, siehe Kapitel 3.4.2) hinsichtlich **Recheneffizienz** verglichen. Wie in Kapitel 3.4.2 erläutert, entspricht Two Stage dem Stand der Forschung für das Lernen komplexitätsminimaler boolescher Ausdrücke mit perfekter Trainingsgenauigkeit. Der Vergleich bestätigt, dass Two Stage im Gegensatz zu Methode 4 nicht ausreichend

effizient ist um die in der vorliegenden Arbeit betrachteten Probleme zu lösen (siehe Kapitel 3.4.2). In der **zweiten Benchmarkstudie** wurde Methode 4 mit dem Algorithmus DK-XTSD von Costamagna & Micheli (2023, siehe Kapitel 3.4.2) hinsichtlich **Generalisierungsfähigkeit** verglichen. Wie in Kapitel 3.4.2 erläutert, entspricht DK-XTSD dem Stand der Forschung für das Lernen boolescher Ausdrücke mit perfekter Trainingsgenauigkeit, er garantiert jedoch keine minimale Komplexität. Der Vergleich zeigt, dass Methode 4 eine höhere Generalisierungsfähigkeit aufweist als DK-XTSD und damit für die in der vorliegenden Arbeit betrachteten Probleme besser geeignet ist. Die Benchmarkstudie ist in Anhang A10.4 ausführlich dargestellt. Im Folgenden wird die Demonstration der Methode 4 an den vom Industriepartner bereitgestellten Daten beschrieben.

5.4.1 Vorgehen

Zunächst wird ein Trainingsdatensatz für die Anwendung der Methode 4 erstellt. Ein Datenpunkt entspricht dabei einer zulässigen Variante des HLKM. Die **Features** des Datenpunkts entsprechen den Produktmerkmalen, wobei kategorische Merkmale in One-Hot-Codierung überführt werden. Die **Labels** des Datenpunkts entsprechen jeweils einem booleschen Wert, der angibt, ob eine bestimmte Position der MSTL oder des MAPL für diese Variante aktiv ist. Um zu vermeiden, dass die Ergebnisse der Demonstration durch die Wahl der Varianten verzerrt werden, werden gleichmäßig zufällig Varianten aus dem HLKM des betrachteten Produkts ausgewählt. Der Anteil zulässiger Varianten unter Berücksichtigung von Beschränkungen an allen kombinatorisch möglichen Varianten ist bei den betrachteten Produkten mit ca. 1 zu 1,43 * 10^{17} gering. Eine naheliegende Auswahl einer Variante durch gleichmäßig zufällige Festlegung der Merkmalausprägungen ohne Berücksichtigung der Beschränkungen führt deshalb in fast allen Fällen zu einer unzulässigen Variante. Deshalb wird **Uniform Model Sampling** nach Soos et al. (2020) unter Verwendung der Python-Bibliothek pyunigen[67] eingesetzt, um gleichmäßig zufällige Lösungen der High-Level-Formel (HLF) zu bestimmen[68]. Auf diese Weise werden je Durchlauf $\hat{n}^{Training} = 200$ Varianten ausgewählt. Für die ausgewählten Varianten werden mithilfe der vorhandenen LLKMs Labels generiert, d. h. es wird ermittelt, welche Positionen der MSTL und des MAPL aktiv sind. Mit dem Binary-

[67] Siehe https://pypi.org/project/pyunigen/ (zuletzt überprüft am 07.06.2025). Die Einstellparameter des Algorithmus werden mit $\delta = 0{,}5$ und $\epsilon = 0{,}5$ gewählt.
[68] Um vertretbare Rechenzeiten zu erreichen, müssen die Produktmerkmale für die Erstellung der HLF statt per One-Hot-Codierung per Dualcodierung codiert werden, da ansonsten zu viele Binärvariablen und Klauseln vorliegen.

Relevance-Ansatz wird daraus ein Trainingsdatensatz je Label erstellt[69]. Zunächst werden je Trainingsdatensatz die ersten $n^{Training} = 10$ Datenpunkte ausgewählt und Methode 4 jeweils auf diese reduzierten Datensätze angewandt. Anschließend wird das Experiment sukzessive für $n^{Training} = 20, 50, 100$ und 200 wiederholt, um den Einfluss der Anzahl von Datenpunkten zu ermitteln. Aus jedem Experiment ergibt sich eine Regel, d. h. ein boolescher Ausdruck, je Label. Für die Bewertung der datenbasiert erstellten Regeln werden zum einen die **Metriken** r^{GenIn} und r^{GenEx} verwendet, welche der **Genauigkeit** der Regeln auf einem Testdatensatz mit oder ohne Berücksichtigung trivialer Regeln (siehe Kapitel 5.1) entsprechen. Zum anderen werden die Metriken r^{ModIn} und r^{ModEx} verwendet, die den Anteil vollständig **korrekte**r **Modelle** mit bzw. ohne Berücksichtigung trivialer Regeln angeben. Die Metriken werden in Anhang A10.1 ausführlich erläutert.

Da die Methode 4 rechenintensiv ist und im Rahmen der Demonstration vielfach angewandt wird, wird je zu erstellender Regel eine **Zeitbeschränkung von 100 Sekunden** festgelegt. Es werden **10 Durchläufe** durchgeführt und die Ergebnisse über die Durchläufe gemittelt.

5.4.2 Ergebnisse

Im Folgenden werden die Ergebnisse für Produkt B vorgestellt, dessen KM eine höhere Komplexität als das von Produkt A aufweist. Die Ergebnisse für Produkt A finden sich in Anhang A10.2. Soweit die Betrachtung der Ergebnisse von Produkt A zusätzliche Erkenntnisse ermöglicht, wird darauf im Folgenden eingegangen.

Abbildung 5.5 und Abbildung 5.6 zeigen r^{GenIn} und r^{GenEx} bzw. r^{ModIn} und r^{ModEx} in Abhängigkeit von $n^{Training}$ für das **Produkt B**. Alle Kurven verlaufen monoton steigend, d. h. mehr Trainingsdaten führen zu genaueren Modellen und zu höheren Wahrscheinlichkeiten die tatsächlichen Regeln exakt abzubilden. Wie aufgrund der Definition der beiden Metriken zu erwarten, verläuft die Kurve von r^{GenEx} unterhalb der Kurven von r^{GenIn} und die Kurve von r^{ModEx} unterhalb der Kurve von r^{ModIn}. Im Vergleich von r^{GenEx} mit r^{ModEx} zeigt sich, dass bereits für $n^{Training} = 10$ Regeln mit einer Genauigkeit von

[69] Für Produkt A bzw. Produkt B liegen insgesamt 852 bzw. 1.982 Positionen in der MSTL und im MAPL vor. Da sich die Labels der Trainingsdatensätze darüber hinaus aus Regeln sehr unterschiedlicher Komplexität ergeben, decken die Trainingsdatensätze ein breites Spektrum an möglichen Anwendungsfällen ab. Deshalb ermöglichen die KM des Industriepartners eine umfassende Demonstration, so dass auf die Verwendung synthetischer Daten verzichtet wird.

über 94 % erstellt werden können, auch wenn nur weniger als 7 % der nicht trivialen Regeln korrekt abgebildet werden. Entsprechend des Vorgehens der Methode 4 werden für $n^{Training} = 10$ Regeln mit geringer Komplexität erstellt. Der niedrige Wert für r^{ModEx} zeigt, dass deren Komplexität zu gering ist, um die tatsächlichen Regeln der KMs korrekt wiederzugeben. Der hohe Wert für r^{GenEx} zeigt hingegen, dass tatsächliche Regeln des KM, auch wenn sie komplex sind, durch einfache Regeln gut angenähert werden können. Dabei bleiben allerdings zwangsläufig seltene Fälle, die durch lange Monome in der Regel ausgedrückt werden, unberücksichtigt. Für eine zunehmende Datenmenge zeigen sowohl r^{GenEx} als auch r^{ModEx} ein **asymptotisches Verhalten**. Es sind somit immer größere Datenmengen erforderlich, um Testgenauigkeit und Modellübereinstimmung weiter zu steigern. Auch für $n^{Training} = 200$ verbleiben zahlreiche Regeln, die **nicht vollständig korrekt** erkannt werden. Die Auswirkung dessen auf die Testgenauigkeit, d. h. die Genauigkeit der Vorhersage von VSTLs und VAPLs, die selbst für nichttriviale Regeln **über 99 %** beträgt, ist jedoch gering.

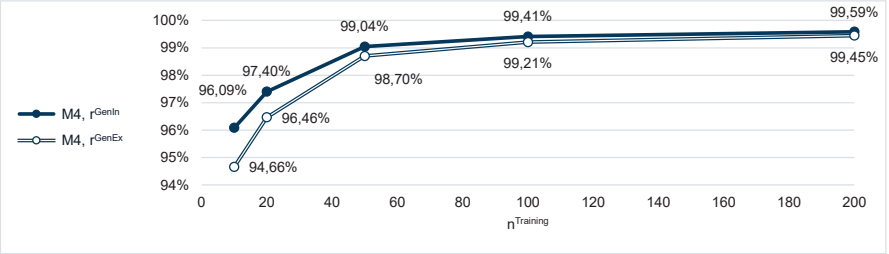


Abbildung 5.5: Ergebnisse der Demonstration der Methode 4 an Produkt B hinsichtlich Testgenauigkeit

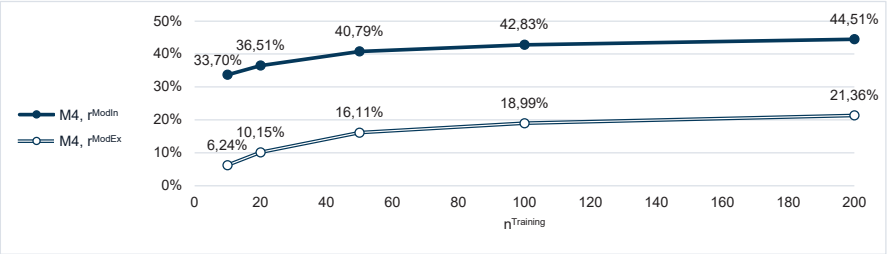


Abbildung 5.6: Ergebnisse der Demonstration der Methode 4 an Produkt B hinsichtlich Modellübereinstimmung

Der Verlauf der Kurven für **Produkt A** ist grundsätzlich mit denen von Produkt B vergleichbar, wobei jedoch die Kurve für r^{ModEx} deutlich oberhalb der entsprechenden Kurve für Produkt B verläuft. Für $n^{Training} = 200$ können für Produkt A ca. 53 % der Regeln korrekt abgebildet werden ggü. ca. 21 % für Produkt B. Dies ist zu erwarten, weil die Regeln für Produkt B komplexer sind und damit von Methode 4 erst später gefunden werden. Wie der Vergleich der Ergebnisse für Produkt A und B zeigt, gilt dies jedoch nicht zwangsläufig für die Generalisierungsfähigkeit, die für Produkt A geringer ist als für Produkt B. Die Komplexität einer Regel legt damit nicht zwingend fest, wie gut diese durch Regeln geringerer Komplexität angenähert werden kann.

5.5 Methode 5: Auswahl von repräsentativen Varianten zur Erweiterung der Datenbasis

In diesem Kapitel wird die Demonstration der Methode 5 beschrieben.

5.5.1 Vorgehen

Zunächst wird ein **Trainingsdatensatz** mit $n^{Training} = 10$ Datenpunkten zu zufällig ausgewählten Varianten und einem **Label** je Position der MSTL und des MAPL erstellt, wie in Kapitel 5.4.1 beschrieben. Damit wird berücksichtigt, dass im Unternehmen i. d. R. bereits VSTLs und VAPLs für Varianten existieren, die nicht systematisch ausgewählt wurden. Dieser initiale Trainingsdatensatz wird zunächst nicht nach dem Binary-Relevance-Ansatz geteilt. Ausgehend vom initialen Trainingsdatensatz erfolgt eine **Simulation des iterativen Prozesseses** in den Methode 5 eingebettet ist (siehe Kapitel 4.1.3). In jeder Iteration wird Methode 5 angewandt, um eine Variante aus dem HLKM auszuwählen. Zu Beginn der Iteration wird, wie in Kapitel 4.5.1 beschrieben, ein Versionenraum (VR) pro Label erstellt. Das Komplexitätsminimale Modell je VR entspricht nach Annahme der genauesten, zu diesem Zeitpunkt bekannten Regel. Es wird mit den in Kapitel 5.4.1 genannten Metriken bewertet[70].

Für die ausgewählte Variante werden mit Hilfe des LLKM Labels erstellt, d. h. bestimmt, welche Positionen der MSTL und des MAPL aktiv sind. Dieser Schritt simuliert die Erstellung von VSTLs und VAPLs durch einen Experten. Zum Abschluss der Iteration liegt ein zusätzlicher **annotierter Datenpunkt** vor, der dem Trainingsdatensatz hinzugefügt

[70] Liegen mehrere komplexitätsminimale Modelle im VR vor, wird willkürlich dasjenige verwendet, das zuerst in den VR aufgenommen wurde.

wird. Anschließend wird Methode 5 erneut angewandt, indem die VRs aktualisiert werden, wie in Kapitel 4.5.1 beschrieben, und eine weitere Variante ausgewählt wird. Der iterative Prozess wird fortgesetzt, bis $\hat{n}^{Training}$ Datenpunkte im Trainingsdatensatz vorliegen. Da Methode 5 rechenintensiv ist, wird für die Demonstration $\hat{n}^{Training} = 100$ gewählt, um die Experimente in vertretbarer Zeit durchführen zu können. Darüber hinaus wird zugunsten einer vertretbaren Rechenzeit der Parameter n^{VR} (siehe Kapitel 4.5.1) auf 3 gesetzt, d. h. es liegen immer **3 Modelle je VR** vor. Der Parameter w^{MS} (siehe Kapitel 4.5.2.5) wird entsprechend der Ergebnisse der Parameterstudie in Anhang A11.1 mit 0,5 gewählt. Für die Demonstration werden kategorische Merkmale in One-Hot-Codierung codiert (siehe Anhang A6.2)[71].

5.5.2 Ergebnisse

Im Folgenden werden die Ergebnisse für Produkt B vorgestellt, dessen KM eine höhere Komplexität als das von Produkt A aufweist. Die Ergebnisse für Produkt A finden sich in Anhang A12.3. Soweit die Betrachtung der Ergebnisse von Produkt A zusätzliche Erkenntnisse ermöglicht, wird darauf im Folgenden eingegangen.

Abbildung 5.7 und Abbildung 5.8 zeigen r^{GenIn} und r^{GenEx} bzw. r^{ModIn} und r^{ModEx} in Abhängigkeit von $n^{Training}$ für das Produkt B. Zum Vergleich sind die entsprechenden Ergebnisse aus der Demonstration der Methode 4 hinterlegt. Ebenso wie die Kurven für Methode 4 verlaufen die Kurven für Methode 5 monoton steigend, d. h. mehr

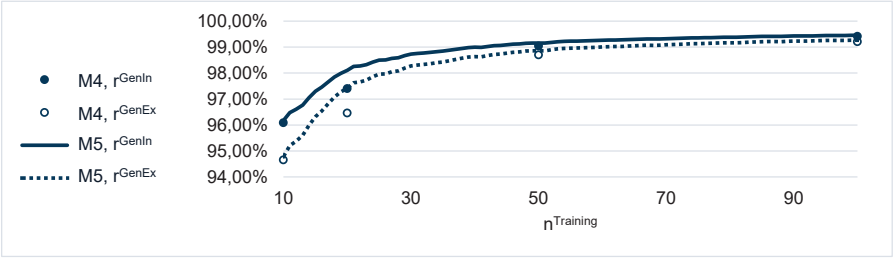


Abbildung 5.7: Ergebnisse der Demonstration der Methode 5 an Produkt B hinsichtlich Testgenauigkeit

[71] Die gleichzeitige Auswahl verschiedener boolescher Variablen die zum selben kategorischen Merkmal gehören wird verhindert, indem Nebenbedingungen in das in Kapitel 4.5.2.5 beschriebene Optimierungsproblem eingeführt werden, die die Summe der Variablen auf kleiner gleich 1 beschränken.

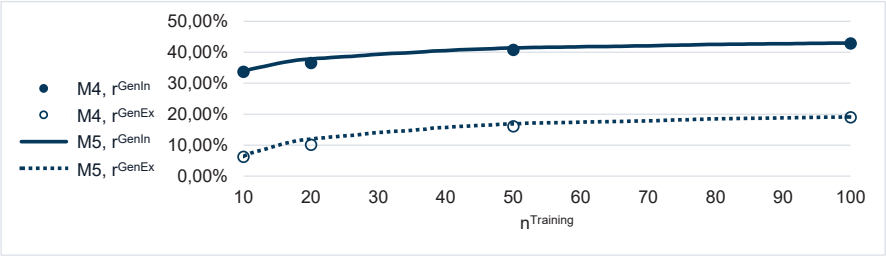


Abbildung 5.8: Ergebnisse der Demonstration der Methode 5 an Produkt B hinsichtlich Modellübereinstimmung

Trainingsdaten führen zu höheren Testgenauigkeiten und höheren Modellübereinstimmungen. Die Kurven für Methode 5 verlaufen stets **oberhalb der Referenzpunkte** für Methode 4.

Tabelle 5.2 zeigt den Vergleich im Detail sowie den p-Wert eines **zweiseitigen Welch-Tests** für die Nullhypothese einer identischen Verteilung.

Tabelle 5.2: Ergebnisse der Demonstrationen der Methode 4 und Methode 5 an Produkt B im Detailvergleich

r^{GenEx}	$n^{Training}$				r^{ModEx}	$n^{Training}$			
	10	**20**	**50**	**100**		**10**	**20**	**50**	**100**
Methode 4	94,66 %	96,46 %	98,70 %	99,21 %	**Methode 4**	6,24 %	10,15 %	16,11 %	18,99 %
Methode 5	94,79 %	97,42 %	98,86 %	99,27 %	**Methode 5**	6,40 %	11,99 %	16,90 %	19,13 %
p-Wert	38,36 %	0,00 %	0,15 %	5,25 %	**p-Wert**	36,60 %	0,00 %	0,17 %	59,02 %

Nur für $n^{Training} = 20$ und $n^{Training} = 50$ kann die Nullhypothese zum Signifikanzniveau 5 % verworfen werden, d. h. nur für diese Fälle ist die Differenz der Ergebnisse für Methode 4 und Methode 5 statistisch signifikant. Dass die Differenz für $n^{Training} = 10$ nicht statistisch signifikant ist, ist dadurch bedingt, dass die ersten 10 Datenpunkte sowohl für die Demonstration von Methode 4 als auch für die Demonstration von Methode 5 zufällig gewählt wurden. Dass der Vorteil der Methode 5 gegenüber einer zufälligen Auswahl von Varianten mit größeren Datenmengen abnimmt, kann verschiedene Ursachen haben. Erstens können für kleine Datenmengen stochastisch bedingte unsystematische Verzerrungen auftreten, d. h. bestimmte Bereiche des Variantenraums können überproportional repräsentiert sein. Für größere Datenmengen löst sich diese Verzerrung nach dem Gesetz der großen Zahl auf. Zweitens profitiert Methode 5 mit zunehmender Datenmenge immer weniger vom Diversitätskriterium bei der Auswahl von

Varianten, da die möglichen Abstände zu bereits betrachteten Varianten immer geringer werden. Zuletzt ist davon auszugehen, dass es Regeln gibt, die grundsätzlich mit datenbasierten Methoden auf Basis praxisrelevanter Datenmengen nicht zuverlässig vorhergesagt werden können. Dies ist besonders vor dem Hintergrund der in Kapitel 5.4.2 vorgestellten Ergebnisse naheliegend. Verbleiben nur noch diese Regeln, wird es u. U. irrelevant, welche Datenpunkte im Trainingsdatensatz vorliegen.

Die Ergebnisse für **Produkt A** unterscheiden sich nicht prinzipiell von denen von Produkt B. Ebenso wie bei der Demonstration der Methode 4 können allerdings insbesondere höhere Modellübereinstimmungen erreicht werden, wie z. B. $r^{ModEx} = 54{,}92\ \%$ für $n^{Training} = 100$. Der Vorteil von Methode 5 gegenüber einer zufälligen Auswahl von Varianten ist insbesondere hinsichtlich der Modellübereinstimmung für Produkt A deutlich stärker ausgeprägt als für Produkt B. Dieser Vorteil besteht auch für $n^{Training} = 100$. Der Vergleich der Ergebnisse von Produkt A und B legt nahe, dass vor allem komplexe Regeln – wie sie in Produkt B vorliegen – den Nutzen von Methode 5 begrenzen.

5.6 Methode 6: Datenbasierte Überprüfung von Regeln

In diesem Kapitel wird die Demonstration der Methode 6 beschrieben.

5.6.1 Vorgehen

In den KMs des Industriepartners sind die Positionen der MSTL und des MAPL unmittelbar von den Produktmerkmalen abhängig. Diese Produktmerkmale können kategorisch oder mehrwertig sein, wie in Kapitel 5.1 erläutert. Eine Regel besteht aus einem oder mehreren Termen, die disjunktiv verknüpft sind. Ist im Falle mehrerer Terme einer der Terme wahr, ist die Position aktiv. Jeder **Term** besteht aus einer Verknüpfung von Aussagen hinsichtlich eines Produktmerkmals. Wäre z. B. Merkmal 1 mehrwertig und wären z. B. Merkmale 2 und 3 kategorisch, könnte ein Term wie folgt aussehen:

$$x_1^{init} = \left(v_{1,1}, \neg v_{1,2}\right) \wedge x_2^{init} \notin \left\{v_{2,1}, v_{2,2}\right\} \wedge x_3^{init} \in \{v_{3,1}, v_{3,2}\}, \tag{5.1}$$

wobei x_i^{init} Produktmerkmale und $v_{i,j}$ Merkmalausprägungen des Merkmals i darstellen. Werden kategorische Merkmale per One-Hot-Codierung codiert ergibt sich daraus z. B. der boolesche Ausdruck

$$\left(x_{11} \wedge \neg x_{1,2}\right) \wedge \left(\neg x_{2,1} \wedge \neg x_{2,2}\right) \wedge \left(x_{3,1} \vee x_{3,2}\right). \tag{5.2}$$

Jede boolesche Variable des so transformierten Terms lässt sich somit entweder einem Merkmal oder einer Merkmalausprägung des eigentlichen Terms zuordnen. Im

Allgemeinen sind die Terme in den KMs des Industriepartners nach Transformation in boolesche Ausdrücke keine Monome – ebenso wie der Term in Formel 5.2. Deshalb ergibt sich durch die disjunktive Verknüpfung der Terme im Allgemeinen keine DNF. Anstatt der im Beispiel in Kapitel 4.6.1 verwendeten Literal- und Monomtabelle wurde deshalb zur tabellarischen Codierung der Regeln eine **fallspezifische tabellarische Darstellung** gewählt. Anhang A12.1 geht auf diese Darstellung und ihre Vorteile im Anwendungsfall ein. In dieser Darstellung sind die Ausdrücke durch ihre Terme in einer Termtabelle codiert und die Terme durch ihre Literale in einer Literaltabelle. Nachdem die **Term- und** die **Literaltabelle** vorliegen, werden in diese Tabellen wie in Anhang A12.1 beschrieben, Fehler eingebracht.

Fehler, die in die Literaltabelle eingebracht werden, betreffen einzelne boolesche Variablen des transformierten Terms und damit einzelne Merkmale oder Merkmalausprägungen im eigentlichen Term. Betrachtet werden die folgenden **Fehlerarten** (siehe Beispiel in Anhang A12.1):

- **Negationsfehler**: Gegenüber der tatsächlichen Regel liegt eine boolesche Variable im transformierten Term negiert anstatt positiv vor oder umgekehrt.
- **Zusätzliche Variable**: Gegenüber der tatsächlichen Regel liegt eine zusätzliche boolesche Variable im transformierten Term vor. Berücksichtigt werden dabei nur solche Variablen, die an der entsprechenden Stelle tatsächlich auftreten können.
- **Fehlende Variable**: Gegenüber der tatsächlichen Regel fehlt eine boolesche Variable im transformierten Term.

Alle booleschen Variablen im transformierten Term, die demselben kategorischen Merkmal zugeordnet sind, treten entweder alle negiert oder alle positiv auf. Deshalb ist davon auszugehen, dass Negationsfehler in einzelnen Variablen bereits durch eine syntaktische Prüfung der Regeln durch das KS gefunden würden. Daher wird auf Negationsfehler in solchen Variablen verzichtet, um die Ergebnisse der Demonstration nicht positiv zu verfälschen. Damit können Negationsfehler ausschließlich boolesche Variablen im transformierten Term betreffen, die aus mehrwertigen oder booleschen Merkmalen hervorgegangen sind. Dadurch sind diese Variablen beim Einbringen zufälliger Fehler tendenziell öfter betroffen. Dies wird, wie unten beschrieben, bei der Bewertung berücksichtigt. Fehler, die in die Termtabelle eingebracht werden, entsprechen Fehlern, die einen gesamten Term einer Regel betreffen. Es werden die Fehler zusätzlicher Term und fehlender Term betrachtet, die einem zusätzlichen bzw. fehlenden Term in einer Regel entsprechen.

In jedem Durchlauf der Demonstration werden entweder Fehler in die Literaltabelle oder in die Termtabelle eingebracht. Jede durch Methode 6 vorgeschlagene **Überprüfung** betrifft genau ein Feld der Literal- oder der Termtabelle. Die Hinweise lassen sich unmittelbar auf die Terme in ihrer initialen Form übertragen. Für die Überprüfung eines Hinweises in der Literaltabelle ist ein Produktmerkmal oder eine Produktmerkmalausprägung für einen bestimmten Term in den Regeln zu überprüfen. Tritt dieser Term mehrfach in Regeln auf, ist jeweils eine Überprüfung durchzuführen. Für die Überprüfung eines Hinweises in der Termtabelle ist genau ein Term in den Regeln auf korrektes Vorhandensein oder Nichtvorhandensein zu überprüfen. Je Hinweis liegt ein Korrekturvorschlag vor (siehe Kapitel 4.6.5). Damit sind die Hinweise sehr spezifisch, weshalb davon auszugehen ist, dass diese von einem Experten schnell überprüft werden können.

Es werden je Durchlauf n^{Fehler} wie in Anhang A12.1 beschrieben gleichmäßig zufällig eingebracht. Variiert werden für die Experimente

- das **Konfigurationsmodell** ($k^{KM} = A$ oder $k^{KM} = B$), um eine gewisse Generalisierbarkeit der Erkenntnisse zu gewährleisten,
- die Verteilung $k^{Fehlerart}$ der **Fehlerarten** – entweder ausschließlich eine Fehlerart oder gleichverteilt über alle Fehlerarten – um zu ermitteln, welche Art von Fehlern besonders gut oder besonders schlecht gefunden werden
- und die Anzahl n^{Fehler}, der **eingebrachte**n **Fehler**, um zu untersuchen, inwiefern Fehler in Modellen mit vielen Fehlern effizienter oder weniger effizient gefunden werden können.

Zunächst wird von $n^{Fehler} = 100$ ausgegangen. Um die Ergebnisse abzusichern, werden jedoch auch Experimente mit deutlich weniger Fehlern ($n^{Fehler} = 10$) und deutlich mehr Fehlern ($n^{Fehler} = 1000$) durchgeführt.

Die Einstellparameter der Methode 6 entsprechen den Einstellparametern des Random-Forest-Algorithmus den sie nutzt. Bei der informationstechnischen Implementierung der Methode 6 im Rahmen der vorliegenden Arbeit wird der Random-Forest-Algorithmus der Software-Bibliothek scikit-learn[72] verwendet. Grundsätzlich werden die Standardeinstellungen der Bibliothek genutzt. Für vier Parameter, bei denen von einer

[72] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (zuletzt überprüft am 02.09.2024)

großen Auswirkung auf die Ergebnisse ausgegangen wird, wird jedoch eine **Parameterstudie** durchgeführt. Anhang A12.2 zeigt die Parameterstudie und die für die Demonstration gewählten Parameterausprägungen, die sich daraus ergeben.

Um zu überprüfen, ob Methode 6 grundsätzlich geeignet ist, um Problem 6 zu lösen, wird eine **Benchmark-Methode** betrachtet, die **zufällige Einträge** der Literal- bzw. Termtabelle auswählt und zur Überprüfung vorschlägt. Durch das zufällige Einbringen von Fehlern sind im Falle gleichverteilter Fehlerarten, wie oben beschrieben, nicht alle Produktmerkmale gleich oft von Fehlern betroffen. Damit könnte Methode 6 implizit zu einer Strategie führen, die schlicht diejenigen Einträge häufiger auswählt, die mit einer höheren Wahrscheinlichkeit Fehler enthalten. Deshalb wird eine **weitere Benchmark-Methode** betrachtet, die **zusätzliche Informationen** darüber nutzt, wie viele Fehler jeweils pro Merkmaltyp vorliegen. Die Methode wählt zufällig Einträge der Literal- bzw. Termtabelle aus, die zu Merkmaltypen mit einem höherem Fehleranteil gehören. Sie nutzt damit mehr Informationen, als in der praktischen Anwendung verfügbar wären. Falls Methode 6 ausschließlich Hinweise auf Basis von Merkmaltypen treffen würde, könnte sie keine besseren Ergebnisse als diese Benchmark-Methode erzielen. Entscheidend für den wirtschaftlichen Einsatz der Methode 6 ist die Anzahl generierter Hinweise, die überprüft werden müssen, um einen gewissen Anteil von Fehlern in den Regeln eines LLKM zu finden. Dies entspricht der Metrik der Demonstration.

5.6.2 Ergebnisse

Im Folgenden werden die Ergebnisse für Produkt B vorgestellt, dessen KM eine höhere Komplexität als das von Produkt A aufweist. Die Ergebnisse für Produkt A finden sich in Anhang A12.3. Soweit die Betrachtung der Ergebnisse von Produkt A zusätzliche Erkenntnisse ermöglicht, wird im Folgenden darauf eingegangen.

Abbildung 5.9 zeigt die Ergebnisse der Experimente für den Referenzfall mit $n^{Fehler} = 100$ in der **Literaltabelle** und einer Gleichverteilung der Fehler über alle drei Fehlerarten[73] in einer Gesamt- und einer Detailansicht für **Produkt B**. Für eine zufällige Auswahl werden Fehler immer mit in etwa gleicher Wahrscheinlichkeit gefunden. Deshalb verläuft die entsprechende Kurve in etwa linear. Die Kurve für die zufällige Auswahl unter zusätzlicher Information (z. I.) verläuft in zwei Abschnitten linear. Zunächst werden alle

[73] Um exakt 100 Fehler zu betrachten existiert ein zusätzlicher Negationsfehler, d. h. es liegen 34 Negationsfehler, 33 zusätzliche Variablen und 33 fehlende Variablen vor.

Felder der Literaltabelle mit hoher Fehlerwahrscheinlichkeit durchsucht und anschließend alle Felder mit niedriger Fehlerwahrscheinlichkeit. Deshalb verläuft die Kurve im ersten Abschnitt flacher als im zweiten. Die Kurve der Methode 6 verläuft bis ca. 84 % der gefundenen Fehler in etwa linear und zeigt anschließend ein überlineares Verhalten[74] (siehe Abbildung 5.9, rechts). Im Datensatz liegen also zum einen Fehler vor, die durch Anomalieerkennung schnell identifiziert werden können und zum anderen Fehler, deren Ausreißerwerte weniger stark ausgeprägt sind. Die Kurve für Methode 6 verläuft durchgehend unterhalb der beiden anderen Kurven. Damit zeigt sich, dass es mit Methode 6 möglich ist, Negationsfehler, zusätzliche Variablen und fehlende Variablen durch Anomalieerkennung effizient zu ermitteln. Um z. B. 84 % der Fehler zu finden, sind im Mittel selbst bei zusätzlicher Information über die Verteilung der Fehlerhäufigkeiten ca. 220-mal so viele Überprüfungen[75] notwendig, wenn eine zufällige Überprüfung der Regeln durchgeführt wird. Sobald die Kurve für Methode 6 nach der Ermittlung von 84 % der Fehler mit ca. 1.255 Überprüfungen den linearen Abschnitt verlässt, nimmt die Effizienz der Methode ab. Wie in Kapitel 4.6.5 beschrieben, ist dies ein geeignetes **Abbruchkriterium**. Anschließend kann bei Bedarf mit empirischem Testen fortgefahren werden. Da die Hinweise zur Überprüfung wie zuvor beschrieben, sehr spezifisch sind und damit schnell überprüft werden können, erscheint die Durchführung von 1.255 Überprüfungen praxisrelevant. Damit können mit Methode 6 im betrachteten Fall **mindestens 84 % der Fehler effizient gefunden** werden. Der Abstand zu den Kurven der zufälligen Methoden vergrößert sich jedoch auch darüber hinaus noch

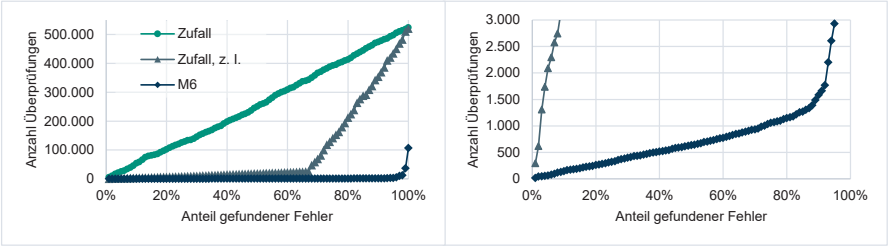


Abbildung 5.9: Ergebnisse der Demonstration der Methode 6 für die Literaltabelle von Produkt B mit $n^{Fehler} = 100$ und $k^{Fehlerart} = Gleichverteilt$ in Gesamtansicht (links) und Detailansicht (rechts)

[74] Zur Ermittlung des Übergangspunkts wird die in Anhang AA12.3 beschriebene Methode verwendet.
[75] 276.532,1 Überprüfungen ggü. 1.254,8 Überprüfungen

weiter. Somit können auch weitere Fehler systematisch gefunden werden, wenn auch weniger effizient. Wie die Ergebnisse für **Produkt A** (siehe Anhang A12.3) zeigen, reduziert sich für weniger komplexe KMs der Überprüfungsaufwand. So sind z. B. lediglich 889,9 Überprüfungen notwendig, um 84 % der Fehler zu finden. Der prinzipielle Verlauf der Kurve ist für Produkt A jedoch identisch.

Wie Abbildung 5.10 (links) zeigt, ist der Aufwand zum Finden von Fehlern je nach **Fehlerart** unterschiedlich groß. Zusätzliche Variablen können mit weniger Überprüfungen gefunden werden als fehlende Variablen und Negationsfehler. Dass es weniger aufwändig ist, zusätzliche Variablen zu finden als fehlende Variablen oder Negationsfehler, ist naheliegend. Es gibt i. d. R. sehr viele Möglichkeiten, zusätzliche Variablen in einen Term einzufügen und dies führt in vielen Fällen zu Kombinationen von Variablen, die im Datensatz ansonsten nicht auftreten. Das Finden von fehlenden Variablen ist zunächst in etwa ebenso effizient möglich wie das Finden von Negationsfehlern. Ab ca. 84 % der gefundenen Fehler steigt jedoch der Überprüfungsaufwand für das Finden weiterer Fehler für fehlende Variablen steiler an als für Negationsfehler. Im Allgemeinen können also fehlende Variablen und Negationsfehler mit identischer Effizienz gefunden werden, jedoch existieren bestimmte fehlende Variablen, die nur mit großem Aufwand gefunden werden können. Dies ist begründbar, da es Fälle geben kann in denen bestimmte Merkmale oder Merkmalausprägungen nur in bestimmten Fällen relevant sind. Es existieren somit ähnliche Terme mit und ohne ein bestimmtes Merkmal oder eine bestimmte Merkmalausprägung. Daraus ergeben sich Fehler, die schwer zu finden sind. Die Kurven für alle drei Fehlerarten weisen einen Übergang von einem linearen in einen überlinearen Abschnitt auf. Für alle drei Fehlerarten existieren also Fehler, die mit Methode 6 effizient erkannt werden können und Fehler, die mit Methode 6 nicht

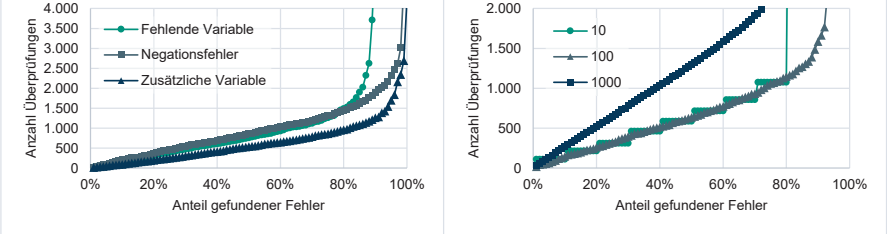


Abbildung 5.10: Ergebnisse der Demonstration der Methode 6 für die Literaltabelle von Produkt B mit variierten Fehlerarten (links) und variierten Fehleranzahlen (rechts)

effizient erkannt werden können. Der Verlauf der Kurve aus Abbildung 5.9 lässt sich als Konsequenz daraus verstehen. Die Ergebnisse für Produkt A sind prinzipiell mit denen für Produkt B vergleichbar, nur dass die Differenzierung zwischen fehlenden Variablen und Negationsfehlern früher einsetzt.

Abbildung 5.10 (rechts) zeigt außerdem, dass die Anzahl benötigter Überprüfungen, um einen bestimmten Anteil aller Fehler zu finden von der **Anzahl bestehender Fehler** abhängt. Wie zu erwarten, sind mehr Überprüfungen notwendig, um z. B. 50 % der Fehler zu finden, wenn dieser Anteil 500 Fehlern entspricht, als wenn dieser Anteil 5 Fehlern entspricht. Dies ist in der Abbildung erkennbar. Ebenso ist allerdings zu erwarten, dass die Anzahl benötigter Überprüfungen nicht linear mit der Anzahl vorliegender Fehler steigt. Mit einer höheren Fehleranzahl nimmt auch die Fehlerwahrscheinlichkeit je Eintrag der Literaltabelle zu. Damit werden mit einer höheren Wahrscheinlichkeit Fehler gefunden. Die Kurve für $n^{Fehler} = 10$ weist Stufen auf, weil jeder Schritt von 10 Prozentpunkten genau einem gefundenen Fehler entspricht. Abgesehen davon ist der grundsätzliche Verlauf der Kurven vergleichbar.

Abbildung 5.11 zeigt die Ergebnisse der Experimente für den Referenzfall für $n^{Fehler} = 100$ in der **Termtabelle** und einer Gleichverteilung der Fehler über die beiden Fehlerarten zusätzlicher Term und fehlender Term für **Produkt B**. Da die Fehlerhäufigkeit für alle Einträge der Termtabelle gleich ist, wird keine zufällige Auswahl mit zusätzlicher Information betrachtet. Die Kurve der zufälligen Auswahl verläuft erneut linear. Die Kurve für Methode 6 weist vier Abschnitte auf. Zunächst steigt sie bis 2 % Anteil der Fehler steil an, verläuft anschließend in etwa linear bis ca. 50 % der Fehler, steigt

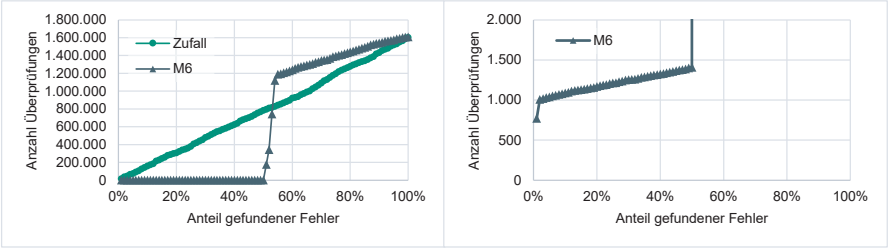


Abbildung 5.11: Ergebnisse der Demonstration der Methode 6 für die Termtabelle von Produkt B mit $n^{Fehler} = 100$ und $k^{Fehlerart} = Gleichverteilt$ in Gesamtansicht (links) und Detailansicht (rechts)

anschließend erneut steil an bis ca. 54 % der Fehler und verläuft abschließend in etwa linear oberhalb der Kurve der zufälligen Auswahl.

Dieser Kurvenverlauf lässt sich durch die Betrachtung der Kurven in Abhängigkeit der **Fehlerart**, wie in Abbildung 5.12 (links) dargestellt, erklären. Liegen ausschließlich zusätzliche Terme vor, zeigt die Kurve zunächst den aus der Betrachtung der Literaltabelle bekannten Verlauf. Liegen hingegen ausschließlich fehlende Terme vor, steigt die Anzahl notwendiger Überprüfungen bereits für das Finden von 3 Fehlern sprunghaft an. Mit Methode 6 können somit **fehlende Terme grundsätzlich nicht effizient identifiziert** werden. Abbildung 5.12 legt nahe, dass in dem in Abbildung 5.11 betrachteten Fall zunächst Fehlerhinweise generiert werden, die auf fehlende Terme schließen lassen und nur eine geringe Spezifizität aufweisen. Anschließend werden effizient Fehlerhinweise generiert, die auf zusätzliche Terme hinweisen. Sind nahezu alle zusätzlichen Terme gefunden, verbleiben lediglich fehlende Terme sowie einige wenige, schwer zu identifizierende zusätzliche Terme, womit keine effiziente Überprüfung mehr möglich ist. Alle weiteren korrekten Hinweise sind im Wesentlichen Zufall.

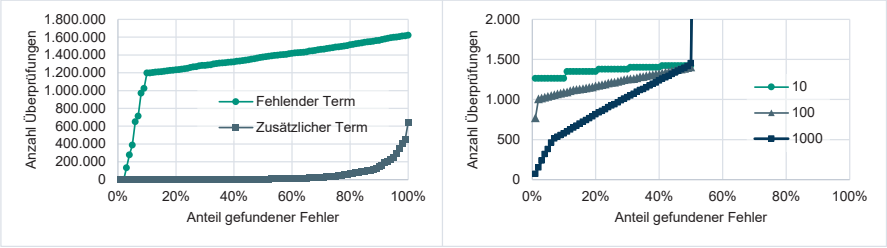


Abbildung 5.12: Ergebnisse der Demonstration der Methode 6 für die Termtabelle von Produkt B mit variierten Fehlerarten (links) und variierten Fehleranzahlen (rechts)

Abbildung 5.12 (rechts) zeigt außerdem, dass sich auch für die Termtabelle die **Anzahl von Fehlern** im Datensatz auf den Überprüfungsaufwand auswirkt. Im Falle von 10 Fehlern in der Termtabelle sind allerdings mehr Überprüfungen notwendig, um einen bestimmten Anteil von Fehlern zu finden, als im Falle von 100 Fehlern. Dies steht im Gegensatz zu den Ergebnissen für die Literaltabelle. Dies kann daran liegen, dass die Hälfte der Fehler fehlenden Termen entspricht und nicht systematisch gefunden werden kann. Es verbleiben also für 10 Fehler nur 5 Fehler, die überhaupt systematisch gefunden werden können, womit die Wahrscheinlichkeit, einen Fehler zu finden zu Ungunsten der geringeren Fehleranzahl ausfällt. Je weniger Fehler im Fall $n^{Fehler} = 100$

verbleiben, desto näher liegt deren Auffindwahrscheinlichkeit bei der des Falls $n^{Fehler} = 10$.

Die Ergebnisse für **Produkt A** sind auch hinsichtlich der Termtabelle mit denen von Produkt B vergleichbar. Auch hier sind jedoch tendenziell weniger Überprüfungen notwendig um einen gewissen Anteil von Fehlern zu finden, als für Produkt B.

6 Diskussion, Fazit und Ausblick

Im Folgenden werden die entwickelten Methoden 1 bis 6 diskutiert und bewertet. Dies entspricht dem fünften Schritt des Design Science Research Process (DSRP, siehe Kapitel 1.3). Auf Basis dessen werden die Forschungsfragen 1 bis 6 beantwortet und ein Ausblick auf den zukünftigen Forschungsbedarf gegeben. Die folgenden Unterkapitel sind den Forschungsfragen und zugehörigen Methoden 1 bis 6 zugeordnet.

6.1 Forschungsfrage 1: Datenbasierte Erstellung von Konfigurationsmodellen

6.1.1 Diskussion

Die Demonstrationen der Methoden 2 bis 5 zeigen, dass diese insgesamt grundsätzlich in der Lage sind, Maximalstücklisten (MSTLs), Maximalarbeitspläne (MAPLs) und Regeln datenbasiert zu erstellen. Damit ist Methode 1 grundsätzlich in der Lage Low-Level-Konfigurationsmodelle (LLKMs) nach Stand der Technik datenbasiert zu erstellen. Aus den Einschränkungen der Methoden 2 bis 5 ergeben sich jedoch Einschränkungen für Methode 1. Bei der Erstellung von MSTLs besteht gegenwärtig noch ein Problem mit der Recheneffizienz für komplexe MSTLs, was in Kapitel 6.2.1 genauer beleuchtet wird. Methode 4 ist in der Lage für relevante Anwendungsfälle Regeln mit hoher Genauigkeit zu erstellen, was in Kapitel 6.4.1 ausgeführt wird. Jedoch können, auch unter Anwendung von Methode 5, i. d. R. keine vollständig korrekten Regeln erstellt werden. Damit kann Methode 1 für die in Kapitel 4.1.2 herausgegriffenen Szenarien DMM, DDD, D0M, D0D und 00D eingesetzt werden, sofern MSTLs mit geringer Komplexität vorliegen. Für die Szenarien D00, 0D0 oder DD0, bei denen nach einer datenbasierten Erstellung der LLKMs auf eine Überwachung und Korrektur im Betrieb verzichtet wird, ist sie hingegen nicht geeignet. Regeln, die nicht vollständig korrekt sind, könnten sich in diesen Szenarien in vereinzelten falschen Positionen der variantenbezogenen Stücklisten (VSTLs) und variantenbezogenen Arbeitsplänen (VAPLs) niederschlagen.

6.1.2 Fazit

Forschungsfrage 1 lässt sich auf Basis der vorliegenden Arbeit wie folgt beantworten. LLKMs lassen sich, mit der im Rahmen der vorliegenden Arbeit entwickelten Methode 1, datenbasiert erstellen. Die Zuverlässigkeit, mit der diese erstellt werden, ist im Rahmen des Betrachtungsumfangs hoch, sodass ein wirtschaftlicher Nutzen zu erwarten

ist. Sie ist jedoch nicht ausreichend hoch für einen unüberwachten Betrieb des Low-Level-Konfigurationssystems (LLKS).

6.1.3 Ausblick

Methode 1 würde von der Weiterentwicklung ihrer untergeordneten Methoden, wie in den folgenden Kapiteln beschrieben, profitieren. Außerdem wäre eine weitergehende Erprobung sinnvoll. Im Rahmen der vorliegenden Arbeit wurde Methode 1 entwickelt und kann durch die Demonstration ihrer untergeordneten Methoden aus technischer Sicht beurteilt werden. Die Ergebnisse der Demonstration lassen darüber hinaus eine erste Einschätzung über den wirtschaftlichen Nutzen der Methode zu. Um die datenbasierte Erstellung von LLKMs abschließend beurteilen zu können, wäre jedoch darüber hinaus eine Studie der Wirtschaftlichkeit und der Akzeptanz in Industrieunternehmen auf Basis verschiedener Anwendungsszenarien notwendig.

6.2 Forschungsfrage 2: Datenbasierte Erstellung von Maximalstücklisten

6.2.1 Diskussion

Die Demonstration der Methode 2 zeigt, dass MSTLs mit bis zu 100 Zukaufkomponentenklassen (ZKKs) ohne Multipositionen und ohne Strukturoptionen (STOs) mit einer zweistelligen Anzahl von Datenpunkten zuverlässig datenbasiert erstellt werden können. Ob dies auch für MSTLs mit Multipositionen oder mit STOs gilt, konnte auf Basis der durchgeführten Experimente nicht abschließend beurteilt werden, da in der durchgeführten Experimentreihe der Rechenaufwand den begrenzenden Faktor darstellte. Dieser Rechenaufwand kann ein Hindernis für einen Einsatz der Methode 2 in der Industrie darstellen. Dabei ist jedoch zu berücksichtigen, dass für die einmalige Erstellung einer MSTL längere Rechenzeiten akzeptabel sind als für eine umfassende Experimentreihe. Der Rechenaufwand ist damit evtl. nur für sehr große MSTLs mit STOs oder Multipositionen tatsächlich kritisch.

6.2.2 Fazit

Forschungsfrage 2 lässt sich auf Basis der vorliegenden Arbeit wie folgt beantworten. MSTLs lassen sich, mit der im Rahmen der vorliegenden Arbeit entwickelten Methode 2, datenbasiert erstellen. Die Zuverlässigkeit, mit der diese erstellt werden, ist im Rahmen des Betrachtungsumfangs für MSTLs ohne STOs und Multipositionen hoch. Für

MSTLs mit Multipositionen oder STOs kann keine abschließende Aussage getroffen werden.

6.2.3 Ausblick

Die Demonstration der Methode 3, welche demselben Prinzip wie Methode 2 folgt, zeigt, dass die grundsätzliche Vorgehensweise der Methode nicht die Ursache für die geringe Recheneffizienz von Methode 2 ist. Die Gründe liegen viel mehr in den Unterschieden der beiden Methoden: Alg^{MSTL} benötigt mehr Zeit zur Erstellung oder Überprüfung einer MSTL als Alg^{MAPL} zur Erstellung oder Überprüfung eines MAPL. Darüber hinaus nutzen beide Methoden jeweils ein eigenes Distanzmaß für die Priorisierung der Verzweigung im Suchbaum. Inwieweit das für Methode 2 verwendete Maß weniger aussagekräftig ist als das für Methode 3 verwendete Maß wurde in der vorliegenden Arbeit nicht systematisch untersucht. Durch Optimierung von Alg^{MSTL} sowie eine genauere Betrachtung des Distanzmaßes kann u. U. die Recheneffizienz von Methode 2 erhöht werden, wodurch u. U. Rechenzeit als kritischer Faktor eliminiert werden kann.

6.3 Forschungsfrage 3: Datenbasierte Erstellung von Maximalarbeitsplänen

6.3.1 Diskussion

Die Demonstration der Methode 3 zeigt, dass MAPLs mit bis zu 100 Arbeitsvorgangsklassen (AVKs) auch mit Multipositionen und STOs mit einer zweistelligen Anzahl von Datenpunkten zuverlässig datenbasiert erstellt werden können. Für industrielle Anwendungsfälle, die die Prämisse erfüllen, dass identische Arbeitsvorgänge (AVOs) identifizierbar sind (siehe Kapitel 4.3) ist die Methode damit nutzbringend anwendbar. Neben der Erstellung von MAPLs für LLKMs kann die Methode auch zur Ermittlung von Prozessvorranggraphen unter Berücksichtigung von Multipositionen und STOs eingesetzt werden und kann damit z. B. auch einen Beitrag zur Austaktung und Steuerung variantenreicher Montagesysteme leisten.

6.3.2 Fazit

Forschungsfrage 3 lässt sich auf Basis der vorliegenden Arbeit wie folgt beantworten. MAPLs lassen sich, mit der im Rahmen der vorliegenden Arbeit entwickelten Methode 3, datenbasiert erstellen. Die Zuverlässigkeit, mit der diese erstellt werden, ist im

Rahmen des Betrachtungsumfangs sowohl für MAPLs mit STOs und Multipositionen als auch für MAPLs ohne STOs und Multipositionen hoch.

6.3.3 Ausblick

Aufgrund der hohen Reife von Methode 3 bestehen Weiterentwicklungsmöglichkeiten weniger in der Methode selbst als viel mehr in deren Schnittstellen. Zum einen könnten die Voraussetzungen für die Anwendbarkeit reduziert werden, indem geeignete Methoden zur Generalisierung von AVOs entwickelt würden. Diese müssten dergestalt mit Methode 3 integriert sein, dass bei der Generalisierung von AVOs die Anzahl von STOs im MAPL minimiert wird. Zum anderen könnte der Nutzen von Methode 3 für die Arbeitsplanung erhöht werden, indem erforscht würde, wie Multipositionen und STOs für die Austaktung und Steuerung von Montagesystemen verwendet werden können.

6.4 Forschungsfrage 4: Datenbasierte Erstellung von Regeln

6.4.1 Diskussion

Die Demonstration der Methode 4 zeigt, dass sich bereits auf Basis von 10 Datenpunkten, d. h. 10 Varianten mit zugehörigen VSTLs und VAPLs, Regeln mit einer Testgenauigkeit von über 90 % und für 100 Datenpunkte von über 99 % erstellen lassen. Die so erstellten Regeln sind somit geeignet, um Positionen der VSTL und der VAPLs mit hoher Genauigkeit korrekt zu bestimmen. Dabei werden tatsächlich gültige Regeln in vielen Fällen gut angenähert, jedoch nicht zwangsläufig vollständig korrekt erkannt. Methode 4 ist somit geeignet um auf Basis von unsystematisch ausgewählten Varianten automatisch gute, wenn auch nicht vollständig korrekte, Regeln zu erstellen. Auch wenn diese manuell finalisiert werden müssen, um eine Genauigkeit von nahezu 100 % zu erreichen, ist somit eine Verringerung des manuellen Aufwands möglich. Darüber hinaus ist eine Reduktion von Fehlern in Regeln zu erwarten, da die vollständig korrekt ermittelten Regeln – bis zu ca. 53 % in den betrachteten Fällen – manuelle Fehler ausschließen.

6.4.2 Fazit

Forschungsfrage 4 lässt sich auf Basis der vorliegenden Arbeit wie folgt beantworten. Regeln in LLKMs lassen sich, mit der im Rahmen der vorliegenden Arbeit entwickelten Methode 4, datenbasiert erstellen. Die Zuverlässigkeit, mit der diese erstellt werden ist im Rahmen des Betrachtungsumfangs hoch, sodass ein wirtschaftlicher Nutzen zu

erwarten ist. Sie ist jedoch nicht ausreichend hoch, um das zugehörige LLKS unüberwacht zu betreiben.

6.4.3 Ausblick

Methode 4 verfolgt konsequent das Paradigma komplexitätsminimaler Modelle unter Gewährleistung einer perfekten Trainingsgenauigkeit. Dadurch erzielt sie im Anwendungsfall bessere Ergebnisse als vergleichbare Methoden nach Stand der Forschung. Es ist deshalb nicht davon auszugehen, dass auf Basis derselben Datenmengen Modelle mit höherer Genauigkeit erstellt werden können. Ein Nachteil der Methode ist jedoch ihr hoher Rechenaufwand. Dieser könnte voraussichtlich reduziert werden, indem der Stand der Forschung hinsichtlich Spaltengenerierung ausgeschöpft würde, wie in Kapitel 4.4.2.2.1 erwähnt. Dadurch könnten Effizienzprobleme, die für große Konfigurationsmodelle (KMs) auftreten können, gelöst werden. Auch für andere Anwendungsfälle des maschinellen Lernens (ML), die Modelle in Form boolescher Ausdrücke mit perfekter Trainingsgenauigkeit erfordern, wäre eine solche Weiterentwicklung u. U. gewinnbringend.

6.5 Forschungsfrage 5: Auswahl von repräsentativen Varianten zur Erweiterung der Datenbasis

6.5.1 Diskussion

Die Demonstration der Methode 5 zeigt, dass eine datenbasierte Erstellung von Regeln auf Basis systematisch ausgewählter Datenpunkte einer datenbasierten Erstellung von Regeln auf Basis zufällig ausgewählter Datenpunkte überlegen ist. Damit ist die Anwendung von Methode 5 zur Erweiterung der Datenbasis für eine datenbasierte Erstellung von Regeln effizient. Die Effektivität der Methode 5 ist hingegen dadurch begrenzt, dass es Regeln geben kann, die grundsätzlich mit einer relevanten Menge an Daten nicht vollständig korrekt datenbasiert abgebildet werden können. Eine rein datenbasierte Erstellung von vollständig korrekten Regeln ist somit mit wirtschaftlich vertretbarem Aufwand nicht möglich. Dennoch kann Methode 5 genutzt werden, um Regeln mit einer hohen Genauigkeit zu erstellen und somit den Korrekturaufwand im Rahmen des Auftragsabwicklungsprozesses gering zu halten. Dies kann z. B. in Fällen, in denen die Lieferzeit kritisch und nur wenig Wissen über KMs im Unternehmen vorhanden ist, relevant sein. Ein grundsätzliches Problem der Methode 5 ist gegenwärtig noch ihr hoher Rechenaufwand. Die Rechenzeit steigt mit der Anzahl vorliegender Varianten und

betrug in den Experimenten z. B. für 100 Datenpunkte bereits weit über eine Stunde. Das schränkt den in Kapitel 4.1.3 beschriebenen iterativen Prozess insofern ein, als der Nutzer u. U. auf die Ausgabe des Programms warten muss.

6.5.2 Fazit

Forschungsfrage 5 lässt sich auf Basis der vorliegenden Arbeit wie folgt beantworten. Die Datenbasis für die datenbasierte Erstellung von LLKMs lässt sich durch die Auswahl von repräsentativen Varianten mittels der im Rahmen der vorliegenden Arbeit entwickelten Methode 5 effizient erweitern. Durch die systematische Auswahl wird die Zuverlässigkeit der datenbasiert erstellten Regeln in vielen Fällen signifikant erhöht. Auch wenn eine vollständig datenbasierte Erstellung von Regeln nicht wirtschaftlich ist, ist dennoch ein wirtschaftlicher Nutzen für die Erstellung ausreichend genauer Regeln in bestimmten Anwendungsfällen zu erwarten.

6.5.3 Ausblick

Zunächst würde die praktische Anwendung von Methode 5 durch eine Verringerung des Rechenaufwands profitieren, sodass eine Interaktion mit dem entsprechenden Programm in Echtzeit möglich wird. Hierzu könnte z. B. der Einsatz von Metaheuristiken zur Lösung des in Kapitel 4.5.2.5 beschriebenen Optimierungsproblems untersucht werden. Des Weiteren könnten grundsätzliche Alternativen zur Erweiterung der Datenbasis betrachtet werden, wie z. B. die Übertragung von VSTLs und VAPLs verwandter Produkte durch eine Zuordnung auf Basis von Syntaktik und Semantik von Produktmerkmalen und Positionen. Hierbei ist zwar ein manueller Korrekturaufwand zu erwarten, u. U. könnte dies jedoch in Fällen, in denen geeignete Daten verfügbar sind eine sinnvolle Ergänzung zu Methode 5 darstellen.

6.6 Forschungsfrage 6: Datenbasierte Überprüfung von Regeln

6.6.1 Diskussion

Der Vergleich der Ergebnisse für Methode 6 mit zufallsbasierten Benchmarks zeigt, dass Methode 6 in der Lage ist, systematisch Fehler in Regeln in LLKMs zu ermitteln, mit Ausnahme fehlender Terme. Ein Term entspricht hier einem Fall innerhalb einer Regel, der zur Aktivierung einer bestimmten Position der MSTL oder des MAPL führt. Ein fehlender Term entspricht somit einem nichtberücksichtigten Fall. Es ist davon auszugehen, dass fehlende Fälle in Regeln großen Einfluss auf die resultierenden VSTLs

und VAPLs haben, sodass sie effizient mit empirischem Testen gefunden werden können. Im Anwendungsfall konnten jeweils mehr als die Hälfte der in den Regeln vorhandenen Fehler bzgl. Variablen bzw. Monomen mit weniger als 400 zu überprüfenden Tabelleneinträgen gefunden werden. Damit besteht ein klarer Vorteil der Methode 6 gegenüber einer unsystematischen Inspektion.

6.6.2 Fazit

Forschungsfrage 6 lässt sich auf Basis der vorliegenden Arbeit wie folgt beantworten. Regeln in LLKMs lassen sich, durch die im Rahmen der vorliegenden Arbeit entwickelte Methode 6, datenbasiert überprüfen. Die Methode ist nicht effektiv beim Finden fehlender Terme in Regeln. Für alle anderen untersuchten Fehlerarten kann sie effektiv und effizient eingesetzt werden, um einen gewissen Anteil der Fehler zu identifizieren. Für eine wirtschaftliche Identifikation aller oder zumindest nahezu aller Fehler in den Regeln eines LLKM muss sie mit anderen Methoden zur Überprüfung von Regeln kombiniert werden.

6.6.3 Ausblick

Die Einbindung der Methode 6 in betriebliche Abläufe bringt Anforderungen mit sich, die im Rahmen der vorliegenden Arbeit nicht berücksichtigt wurden. I. d. R. erfolgt eine Überprüfung eines LLKM nicht einmalig, sondern regelmäßig oder anlässlich großer Änderungen. Bei jeder Überprüfung über die erste hinaus ist bekannt, welche Bestandteile der Regeln zuvor bereits überprüft, für korrekt befunden und seither nicht geändert wurden. Diese Information kann zum einen trivial in Methode 6 einfließen, indem die entsprechenden Bestandteile nicht mehr als Hinweise vorgeschlagen werden. Zum anderen kann sie jedoch auch für die Ermittlung der Ausreißerwerte genutzt werden, indem korrekte Felder der Literal- und Monomtabelle höher gewichtet werden. Eine solche Weiterentwicklung der Methode 6 kann auf lange Sicht zu ihrem wirtschaftlichen Einsatz im Unternehmen beitragen.

7 Zusammenfassung

Konfigurationssysteme sind ein wichtiges Werkzeug für Industrieunternehmen zur Automatisierung und damit Rationalisierung der Arbeitsablaufplanung im Rahmen des Auftragsabwicklungsprozesses. Großes Potenzial besteht insbesondere in der automatischen Konfiguration von variantenbezogenen Stücklisten und Arbeitsplänen, d. h. in der Produkt- und Prozesskonfiguration. Bisher werden Konfigurationssysteme in diesem Bereich dennoch nicht umfassend eingesetzt. Wesentliche Hinderungsgründe sind der Aufwand für die Erstellung sowie die hohe Fehleranfälligkeit der hinterlegten Konfigurationsmodelle, welche die Rahmenbedingungen und Regeln der Konfiguration festlegen. Diese Herausforderungen können durch datenbasierte Methoden, wie z. B. Verfahren des maschinellen Lernens, adressiert werden. Hierdurch können Modelle für die Produkt- und Prozesskonfiguration zum einen effizient erstellt und zum anderen effizient überprüft werden. Nach Stand der Forschung sind jedoch datenbasierte Methoden im Zusammenhang mit Konfigurationsmodellen nur rudimentär erforscht.

Ziel der vorliegenden Arbeit war es deshalb, die wissenschaftlichen Grundlagen für den Einsatz datenbasierter Methoden zur Erstellung und Überprüfung von Modellen für die Produkt- und Prozesskonfiguration zu erarbeiten. Hierfür wurden sechs Methoden entwickelt. Methode 1 dient der datenbasierten Erstellung von Produkt- und Prozesskonfigurationsmodellen. Es wurde ein Schema für derartige Konfigurationsmodelle entwickelt. Dieses ermöglicht über den Stand der Forschung hinaus alternative Strukturen für Stücklisten und Arbeitspläne, z. B. in Abhängigkeit der vom Kunden gewählten Variante, zu berücksichtigen. Die Anwendungsfälle für eine datenbasierte Erstellung von Produkt- und Prozesskonfigurationsmodellen in der Industrie wurden in systematisch hergeleiteten Anwendungsszenarien zusammengefasst. Die herausgearbeiteten Anwendungsszenarien werden durch Methode 1 adressiert, indem die untergeordneten Methoden 2 bis 5 integriert werden.

Die Methoden 2 und 3 dienen der datenbasierten Erstellung von Maximalstücklisten bzw. Maximalarbeitsplänen. Kern der beiden Methoden ist jeweils die Ermittlung minimaler Maximalstücklisten bzw. Maximalarbeitsplänen sowie einer minimalen Anzahl von Strukturoptionen um die zugrundeliegenden variantenbezogenen Stücklisten bzw. Arbeitspläne korrekt daraus konfigurieren zu können. Hierfür werden jeweils eine heuristische Tiefensuche und eine ganzzahlige lineare Optimierung eingesetzt. Die entwickelte Methode 4 dient der datenbasierten Erstellung von Regeln. Sie fokussiert Regeln

von binären Parametern. Entsprechend ist die Methode ein Verfahren des maschinellen Lernens für Datensätze mit binären Labels. Die Methode nutzt ganzzahlig lineare Optimierung mittels Spaltengenerierung, um Modelle in Form minimaler boolescher Ausdrücke zu lernen. Die entwickelte Methode 5 dient der Erweiterung der Datenbasis durch gezielte Auswahl von repräsentativen Varianten aus dem Konfigurationsraum. Varianten werden so ausgewählt, dass mit wenigen zusätzlichen Daten möglichst genaue Konfigurationsmodelle erstellt werden können. Die Methode nutzt multikriterielle ganzzahlige Optimierung, um zulässige und optimale Varianten auszuwählen, wobei Diversität und die Verkleinerung des Versionenraums als Kriterien berücksichtigt werden. Methode 6 komplementiert Methode 1 durch die datenbasierte Überprüfung von Regeln in Konfigurationsmodellen. Hierfür werden die Regeln zunächst in eine tabellarische Darstellung transformiert. Anschließend wird ein Verfahren des unüberwachten maschinellen Lernens mittels eines Random-Forest-Algorithmus angewandt, das im Rahmen der vorliegenden Arbeit entwickelt wurde. Hiermit können Hinweise auf Anomalien in den tabellarischen Daten und damit den Regeln generiert werden.

Die Methoden 1 bis 6 stellen jeweils Fortschritte gegenüber dem Stand der Forschung dar und ermöglichen insgesamt eine ganzheitliche datenbasierte Erstellung und Überprüfung von Modellen zur Produkt- und Prozesskonfiguration. Um die Anwendbarkeit der Methoden zu überprüfen und deren Effektivität zu beurteilen, wurden diese jeweils als Computerprogramme implementiert und für reale oder synthetische Daten demonstriert. Für die Demonstration an synthetischen Daten wurde im Rahmen der vorliegenden Arbeit je eine Methode zur Erzeugung gleichmäßig zufälliger Maximalstücklisten und Maximalarbeitsplänen entwickelt. Dadurch wurde eine Demonstration der Methoden 2 und 3 für allgemeine Fälle ermöglicht. Die Demonstrationen haben gezeigt, dass alle entwickelten Methoden grundsätzlich funktionsfähig sind und geeignet sind, die entsprechenden Probleme der datenbasierten Erstellung und Überprüfung von Konfigurationsmodellen zu lösen. Mittels Methode 2 können Maximalstücklisten ohne mehrfach auftretende Zukaufkomponentenklassen und ohne Strukturoptionen mit einer Größe von bis zu 100 Zukaufkomponentenklassen bereits mit einer mittleren zweistelligen Anzahl von Datenpunkten zuverlässig erstellt werden. Damit kann Methode 2 für entsprechende Anwendungsfälle in der Praxis eingesetzt werden. Für Methode 3 konnten sogar für alle Ausprägungen von Maximalarbeitsplänen mit bis zu 100 Arbeitsvorgangsklassen gute Ergebnisse mit einer mittleren zweistelligen Anzahl von Datenpunkten erzielt werden. Damit ist auch diese Methode für entsprechende Anwendungsfälle

in der Praxis geeignet. Es konnte gezeigt werden, dass Methode 4 für die datenbasierte Erstellung von Regeln den existierenden Methoden nach Stand der Forschung überlegen ist. Für die Produkte eines Industriepartners konnten Regeln auf Basis von ca. 100 Datenpunkten mit einer Genauigkeit von über 99 % erstellt werden. Die Ergebnisse implizieren, dass die datenbasierte Erstellung vollständig korrekter Regeln nicht mit praxisrelevanten Datenmengen möglich ist. Dennoch kann diese Methode in einigen der herausgearbeiteten Anwendungsszenarien gewinnbringend eingesetzt werden. Werden keine Daten zufälliger Varianten betrachtet, sondern Varianten mit der hierfür entwickelten Methode 5 ausgewählt, bleibt diese Aussage gültig, es kann jedoch eine deutliche Effizienzsteigerung erreicht werden. Die Demonstrationen der Methoden 2 bis 5 zeigen, dass Methode 1 zur datenbasierten Erstellung von Low-Level-Konfigurationsmodellen in vielen praktischen Anwendungsszenarien sinnvoll eingesetzt werden kann. Für die datenbasierte Überprüfung von Regeln mit Methode 6 hat sich ergeben, dass ein fallabhängiger Anteil von mehr als 50 % der eingebrachten Fehler effizient gefunden werden konnte. Damit stellt die Methode eine sinnvolle Ergänzung bestehender Methoden zur Überprüfung von Konfigurationsmodellen dar.

Alles in allem kann festgehalten werden, dass die im Rahmen der vorliegenden Arbeit entwickelten Methoden die datenbasierte Erstellung und Überprüfung von Modellen zur Produkt- und Prozesskonfiguration ermöglichen. Zum einen leistet die Arbeit damit einen Beitrag zum weitergehenden Einsatz von Konfigurationssystemen in der Arbeitsablaufplanung und damit zur Effizienzsteigerung in Industrieunternehmen. Zum anderen legt sie einen Grundstein für eine weitergehende Forschung an datenbasierten Methoden für die Erstellung und Überprüfung von Modellen zur Produkt- und Prozesskonfiguration.

Literaturverzeichnis

Verweise nach Schema A_<Name><Jahr> beziehen sich auf Abschlussarbeiten am wbk Institut für Produktionstechnik des Karlsruher Instituts für Technologie (KIT), die vom Verfasser der vorliegenden Arbeit angeleitet wurden.

A_Kahn 2023

Kahn, J. (2023), *Methode zur Validierungsunterstützung von Konfigurationssystemen mit Ansätzen des maschinellen Lernens. Bachelorthesis. Karlsruher Institut für Technologie (KIT). wbk - Institut für Produktionstechnik*.

A_Zacateco Herrera 2023

Zacateco Herrera, R. (2023), *Automatic validation of production planning documents through data analysis methods*. *Bachelorthesis. Karlsruher Institut für Technologie (KIT). wbk - Institut für Produktionstechnik*.

Abaza 2020

Abaza, S. (2020), „What is and why do we have to know the phylogenetic tree?“, *Parasitologists United Journal*, 13(2), S. 68–71. https://doi.org/10.21608/puj.2020.35843.1082.

Abbasi et al. 2013

Abbasi, E. K.; Hubaux, A.; Acher, M.; Boucher, Q. & Heymans, P. (2013), „The Anatomy of a Sales Configurator: An Empirical Study of 111 Cases“. *Proceedings of the 25th international conference on Advanced information systems engineering,* Hrsg. C. Salinesi, C. B. Achour-Salinesi, M. C. Norrie & O. Pastor, Springer, Berlin, Heidelberg, S. 162–177. ISBN: 9783642387098. https://doi.org/10.1007/978-3-642-38709-8_11. https://link.springer.com/chapter/10.1007/978-3-642-38709-8_11.

acatech (Hrsg.) 2013

acatech (Hrsg.) (2013), *Technikwissenschaften*. *Erkennen – Gestalten – Verantworten,* Springer Vieweg, Berlin, Heidelberg. https://link.springer.com/book/10.1007/978-3-642-34605-7. ISBN: 9783642346040.

Aggarwal 2017

Aggarwal, C. C. (2017), *Outlier Analysis,* Springer, Cham. ISBN: 9783319475783.

Aggarwal 2020

Aggarwal, C. C. (2020), *Linear Algebra and Optimization for Machine Learning*. *A Textbook,* Springer, Cham. ISBN: 9783030403430.

Aggarwal 2021

Aggarwal, C. C. (2021), *Artificial Intelligence,* Springer, Cham. ISBN: 9783030723569.

Akutsu et al. 2011

Akutsu, T.; Fukagawa, D.; Takasu, A. & Tamura, T. (2011), „Exact algorithms for computing the tree edit distance between unordered trees“, *Theoretical Computer Science*, 412(4-5), S. 352–364. https://doi.org/10.1016/j.tcs.2010.10.002.

Alabdulmohsin et al. 2015

Alabdulmohsin, I.; Gao, X. & Zhang, X. (2015), „Efficient Active Learning of Halfspaces via Query Synthesis“, *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1). https://doi.org/10.1609/aaai.v29i1.9563.

Alonso et al. 1997

Alonso, L.; Rémy, J. L. & Schott, R. (1997), „A linear-time algorithm for the generation of trees“, *Algorithmica*, 17(2), S. 162–182. https://doi.org/10.1007/BF02522824. https://link.springer.com/article/10.1007/bf02522824.

Altemeier et al. 2009

Altemeier, S.; Brodkorb, D. & Dangelmaier, W. (2009), „A top-down approach for an automatic precedence graph construction under the influence of high product variety“. *IFIP Advances in Information and Communication Technology,* Hrsg. J. A. Turner, Springer, Berlin, Heidelberg, New York, S. 73–80.

Amaitik 2012

Amaitik, S. M. (2012), „An integrated CAD/CAPP system based on STEP features“. *Proceedings of International Conference on Industrial Engineering and Operations Management,* S. 665–673.

Augusto et al. 2022

Augusto, A.; Carmona, J. & Verbeek, E. (2022), „Advanced process discovery techniques“ in *Process mining handbook,* Hrsg. van der Aalst, Wil M. P. & J. Carmona, Springer Nature Switzerland, Cham, Schweiz, S. 76–107.

Bastide et al. 2018

Bastide, P.; Solís-Lemus, C.; Kriebel, R.; William Sparks, K. & Ané, C. (2018), „Phylogenetic Comparative Methods on Phylogenetic Networks with Reticulations“, *Systematic Biology*, 67(5), S. 800–820. https://doi.org/10.1093/SYSBIO/SYY033.

Baumeister & Freiberg 2011

Baumeister, J. & Freiberg, M. (2011), „Knowledge visualization for evaluation tasks“, *Knowledge and Information Systems*, 29(2), S. 349–378. https://doi.org/10.1007/s10115-010-0350-8. https://link.springer.com/article/10.1007/s10115-010-0350-8.

Bender & Gericke 2021

Bender, B. & Gericke, K. (2021), *Pahl/Beitz Konstruktionslehre. Methoden und Anwendung erfolgreicher Produktentwicklung,* Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN: 978-3-662-57302-0.

Bergroth et al. 2000

Bergroth, L.; Hakonen, H. & Raita, T. (2000), „A survey of longest common subsequence algorithms“. *Proceedings Seventh International Symposium on String Processing and Information Retrieval, SPIRE 2000,* Hrsg. P. de La Fuente, IEEE, New York, S. 39–48.

Blumöhr et al. 2019

Blumöhr, U.; Neuhaus, M. & Ukalovic, M. (2019), *Variantenkonfiguration mit SAP. Das Praxishandbuch,* Rheinwerk Verlag, Bonn. https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=3105868. ISBN: 9783836281935.

Boroumand et al. 2021

Boroumand, S.; Bouganis, C.-S. & Constantinides, G. A. (2021), „Learning Boolean Circuits from Examples for Approximate Logic Synthesis“. *Proceedings of the 26th Asia and South Pacific Design Automation Conference,* Hrsg. T. Hattori, ACM, New York, S. 524–529. https://doi.org/10.1145/3394885.3431559.

Bougleux et al. 2017

Bougleux, S.; Brun, L.; Carletti, V.; Foggia, P.; Gaüzère, B. & Vento, M. (2017), „Graph edit distance as a quadratic assignment problem“, *Pattern Recognition Letters*, 87, S. 38–46. https://doi.org/10.1016/j.patrec.2016.10.001.

Bratcu et al. 1999

Bratcu, A.; Minzu, V. & Henrioud, J. M. (1999), „Construction of the precedence graphs equivalent to a given set of assembly sequences“. *IEEE Int. Symp. on Assembly and Task Planning,* S. 14–20.

Braun, F. 2021

Braun, F. (2021), *Application of algorithm-based validation tools for the validation of complex, multi-variant products*. Dissertation, Universität der Bundeswehr München, Neubiberg.

Bredahl Rasmussen et al. 2021

Bredahl Rasmussen, J.; Haug, A.; Shafiee, S.; Hvam, L.; Henrik Mortensen, N. & Myrodia, A. (2021), „The costs and benefits of multistage configuration: A framework and case study“, *Computers & Industrial Engineering*, 153. https://doi.org/10.1016/j.cie.2020.107095. https://www.sciencedirect.com/science/article/pii/S0360835220307658.

Breiman 2001

Breiman, L. (2001), „Random Forests“, *Machine Learning*, 45(1), S. 5–32. https://doi.org/10.1023/A:1010933404324. https://link.springer.com/article/10.1023/A:1010933404324#citeas.

Brinkop et al. 2012

Brinkop, A.; Krebs, T. & Schlee, H. (2012), „K-Model - Structured Design of Configuration Models“. *CONFWS'12: Proceedings of the 2012 International Conference on Configuration,* Hrsg. W. Mayer & P. Albert, CEUR, Aachen, S. 8–14. https://api.semanticscholar.org/CorpusID:14140295.

Cannas et al. 2022

Cannas, V. G.; Masi, A.; Pero, M. & Brunø, T. D. (2022), „Implementing configurators to enable mass customization in the Engineer-to-Order industry: a multiple case study research“, *Production Planning & Control*, 33(9-10), S. 974–994. https://doi.org/10.1080/09537287.2020.1837941.

Cao et al. 2020

Cao, H. E. C.; Sarlin, R. & Jung, A. (2020), „Learning explainable decision rules via maximum satisfiability“, *IEEE Access*, 8, S. 218180–218185.

Chatras et al. 2016

Chatras, C.; Giard, V. & Sali, M. (2016), „Mass customisation impact on bill of materials structure and master production schedule development“, *International Journal of Production Research*, 54(18), S. 5634–5650. https://doi.org/10.1080/00207543.2016.1194539.

Chatterjee 2018

Chatterjee, S. (2018), „Learning and Memorization“. *Proceedings of the 35th International Conference on Machine Learning,* Hrsg. J. Dy & A. Krause, ICML, San Diego, S. 755–763. http://proceedings.mlr.press/v80/chatterjee18a.html?ref=https://githubhelp.com.

Chen et al. 2017

Chen, L.; Hassani, H. & Karbasi, A. (2017), „Near-Optimal Active Learning of Halfspaces via Query Synthesis in the Noisy Setting“. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence,* Hrsg. S. Singh & S. Markovitch, AAAI, Washington, S. 1798–1804. https://doi.org/10.1609/aaai.v31i1.10783.

Chen & Fuge 2018

Chen, W. & Fuge, M. (2018), „Active expansion sampling for learning feasible domains in an unbounded input space“, *Structural and Multidisciplinary Optimization*, 57, S. 925–945. https://doi.org/10.1007/s00158-017-1894-y. https://link.springer.com/article/10.1007/s00158-017-1894-y.

Chen & Wang 2009

Chen, Z. & Wang, L. (2009), „Adaptable product configuration system based on neural network“, *International Journal of Production Research*, 47(18), S. 5037–5066. https://doi.org/10.1080/00207540802007571.

Costamagna & Micheli 2023

Costamagna, A. & Micheli, G. de (2023), „Accuracy recovery: A decomposition procedure for the synthesis of partially-specified Boolean functions“, *Integration*, 89, S. 248–260. https://doi.org/10.1016/j.vlsi.2022.12.008.

Creignou & Hermann 1996

Creignou, N. & Hermann, M. (1996), „Complexity of Generalized Satisfiability Counting Problems“, *Information and Computation*, 125(1), S. 1–12. https://doi.org/10.1006/inco.1996.0016. https://www.sciencedirect.com/science/article/pii/S0890540196900164.

Dash et al. 2018

Dash, S.; Gunluk, O. & Wei, D. (2018), „Boolean Decision Rules via Column Generation“. *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems,* Hrsg. S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman & N. Cesa-Bianchi, Neural Information Processing Systems Foundation, La Jolla, S. 4660–4670. ISBN: 9781713807933.

Deng & Fernández-Baca 2018

Deng, Y. & Fernández-Baca, D. (2018), „Fast Compatibility Testing for Rooted Phylogenetic Trees“, *Algorithmica*, 80(8), S. 2453–2477. https://doi.org/10.1007/s00453-017-0330-4. https://link.springer.com/article/10.1007/s00453-017-0330-4.

DIN 8580:2022-12

DIN 8580:2022-12 (2022), *Fertigungsverfahren - Begriffe, Einteilung,* Beuth Verlag, Berlin.

DIN EN ISO/IEC 22989:2023-04

DIN EN ISO/IEC 22989:2023-04 (2023), *Informationstechnik - Künstliche Intelligenz - Konzepte und Terminologie der Künstlichen Intelligenz,* Beuth Verlag, Berlin.

Domingos 2012

Domingos, P. (2012), „A few useful things to know about machine learning“, *Communications of the ACM*, 55(10), S. 78–87. https://doi.org/10.1145/2347736.2347755.

Duffy & Andreasen 1995

Duffy, A. H. & Andreasen, M. M. (1995), „Enhancing the evolution of design science“. *Proceedings of ICED 95,* Hrsg. V. Hubka, o. A., o. A., S. 29–35.

Dwork et al. 2001

Dwork, C.; Kumar, R.; Naor, M. & Sivakumar, D. (2001), „Rank aggregation methods for the Web“. *Proceedings of the 10th international conference on World Wide Web,* Hrsg. V. Y. Shen, N. Saito, M. R. Lyu & M. E. Zurko, ACM, New York, S. 613–622. https://doi.org/10.1145/371920.372165.

ElMaraghy et al. 2013

ElMaraghy, H.; Schuh, G.; ElMaraghy, W.; Piller, F.; Schönsleben, P.; Tseng, M. &

Bernard, A. (2013), „Product variety management“, *CIRP Annals*, 62(2), S. 629–652. https://doi.org/10.1016/j.cirp.2013.05.007. https://www.sciencedirect.com/science/article/pii/S0007850613001972.

Emmert-Streib & Dehmer 2019

Emmert-Streib, F. & Dehmer, M. (2019), „Evaluation of regression models: Model assessment, model selection and generalization error“, *Machine learning and knowledge extraction*, 1(1), S. 521–551. https://doi.org/10.3390/make1010032.

Englhardt & Böhm 2020

Englhardt, A. & Böhm, K. (2020), „Exploring the Unknown – Query Synthesis in One-Class Active Learning“. *Proceedings of the 2020 SIAM International Conference on Data Mining,* Hrsg. C. Demeniconi & N. Chawla, Society for Industrial and Applied Mathematics, Philadelphia, S. 145–153. https://doi.org/10.1137/1.9781611976236.17.

European Commission 2024

European Commission (2024), *Living guidelines on the responsible use of generative ai in research*. *First Version, March 2024*, European Commission. https://research-and-innovation.ec.europa.eu/document/download/2b6cf7e5-36ac-41cb-aab5-0d32050143dc_en?filename=ec_rtd_ai-guidelines.pdf.

Eversheim 2002

Eversheim, W. (2002), *Organisation in der Produktionstechnik 3*. *Arbeitsvorbereitung,* Springer, Berlin, Heidelberg. https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6306735. ISBN: 9783642563362.

Felfernig et al. 2013

Felfernig, A.; Reiterer, S.; Stettinger, M.; Reinfrank, F.; Jeran, M. & Ninaus, G. (2013), „Recommender Systems for Configuration Knowledge Engineering“. *Proceedings of the 15th International Configuration Workshop,* Hrsg. M. Aldanondo & A. Falkner, CEUR, Aachen, S. 51–54. https://doi.org/10.48550/arXiv.2102.08113.

Felfernig et al. 2014

Felfernig, A.; Hotz, L.; Bagley, C. & Tiihonen, J. (Hrsg.) (2014), *Knowledge-Based Configuration,* Elsevier Inc., Waltham, MA, USA. ISBN: 978-0-12-415817-7.

Felfernig et al. 2015

Felfernig, A.; Reiterer, S.; Stettinger, M. & Tiihonen, J. (2015), „Towards

Understanding Cognitive Aspects of Configuration Knowledge Formalization“. *Proceedings of the Ninth International Workshop on Variability Modelling of Software-intensive Systems - VaMoS '15,* Hrsg. Klaus Schmid, Øystein Haugen, Johannes Müller, ACM Press, New York, S. 117–123. https://doi.org/10.1145/2701319.2701327.

Frey et al. 2023

Frey, A. M.; May, M. C. & Lanza, G. (2023), „Creation and validation of systems for product and process configuration based on data analysis“, *Production Engineering*, 17(2), S. 263–277. https://doi.org/10.1007/s11740-022-01176-1. https://link.springer.com/article/10.1007/s11740-022-01176-1.

Gauss et al. 2021

Gauss, L.; Lacerda, D. P. & Cauchick Miguel, P. A. (2021), „Module-based product family design: systematic literature review and meta-synthesis“, *Journal of Intelligent Manufacturing*, 32(1), S. 265–312. https://doi.org/10.1007/s10845-020-01572-3.

Ghosh et al. 2022

Ghosh, B.; Malioutov, D. & Meel, K. S. (2022), „Efficient Learning of Interpretable Classification Rules“, *Journal of Artificial Intelligence Research*, 74, S. 1823–1863. https://doi.org/10.1613/jair.1.13482. https://www.jair.org/index.php/jair/article/view/13482.

Ghosh & Meel 2019

Ghosh, B. & Meel, K. S. (2019), „IMLI: An Incremental Framework for MaxSAT-Based Learning of Interpretable Classification Rules“. *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society,* Hrsg. V. Conitzer, S. Kambhampati, S. Koenig, F. Rossi & B. Schnabel, ACM, New York, S. 203–210. https://doi.org/10.1145/3306618.3314283.

Glos et al. 2023

Glos, A.; Kundu, A. & Salehi, Ö. (2023), „Optimizing the Production of Test Vehicles Using Hybrid Constrained Quantum Annealing“, *SN Computer Science*, 4. https://doi.org/10.1007/s42979-023-02071-x. https://link.springer.com/article/10.1007/s42979-023-02071-x.

Guiza et al. 2022

Guiza, O.; Mayr-Dorn, C.; Mayrhofer, M.; Egyed, A. & Brandt, H. R. (2022),

„Assembly Precedence Graph Mining Based on Similar Products“. *Proceedings of the 2022 IEEE International Conference on Industrial Technology (ICIT),* Hrsg. L. Gomes, V. Vyatkin, M. Y. Chow & X. Guan, IEEE, New York. https://doi.org/10.1109/icit48603.2022.10002729.

Haag 1998

Haag, A. (1998), „Sales configuration in business processes“, *IEEE Intelligent Systems and their Applications*, 13(4), S. 78–85. https://doi.org/10.1109/5254.708436. http://dx.doi.org/10.1109/5254.708436.

Hanna et al. 2023

Hanna, M.; Wöller, L.-N.; Dambietz, F. M. & Krause, D. (2023), „A Model-Based Approach for the Methodical Development and Configuration of Modular Product Families“, *Systems*, 11(9), S. 449. https://doi.org/10.3390/systems11090449. https://www.mdpi.com/2079-8954/11/9/449.

Hashimoto & Nakamoto 2021

Hashimoto, M. & Nakamoto, K. (2021), „Process planning for die and mold machining based on pattern recognition and deep learning“, *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 15(2), JAMDSM0015-JAMDSM0015. https://doi.org/10.1299/jamdsm.2021jamdsm0015. https://www.jstage.jst.go.jp/article/jamdsm/15/2/15_2021jamdsm0015/_article/-char/ja/.

Haug et al. 2010

Haug, A.; Hvam, L. & Mortensen, N. H. (2010), „A layout technique for class diagrams to be used in product configuration projects“, *Computers in Industry*, 61(5), S. 409–418. https://doi.org/10.1016/j.compind.2009.10.002. https://www.sciencedirect.com/science/article/pii/S0166361509001936.

Haug et al. 2012

Haug, A.; Hvam, L. & Mortensen, N. H. (2012), „Definition and evaluation of product configurator development strategies“, *Computers in Industry*, 63(5), S. 471–481. https://doi.org/10.1016/j.compind.2012.02.001.

Haug et al. 2019a

Haug, A.; Shafiee, S. & Hvam, L. (2019), „The causes of product configuration project failure“, *Computers in Industry*, 108, S. 121–131.

https://doi.org/10.1016/j.compind.2019.03.002. http://dx.doi.org/10.1016/j.compind.2019.03.002.

Haug et al. 2019b
Haug, A.; Shafiee, S. & Hvam, L. (2019), „The costs and benefits of product configuration projects in engineer-to-order companies“, *Computers in Industry*, 105, S. 133–142. https://doi.org/10.1016/j.compind.2018.11.005. http://dx.doi.org/10.1016/j.compind.2018.11.005.

Hawkins 1980
Hawkins, D. M. (1980), *Identification of outliers,* Springer, Dordrecht. ISBN: 9789401539944.

He et al. 2021
He, C.; Li, Z.; Wang, S. & Liu, D. (2021), „A systematic data-mining-based methodology for product family design and product configuration“, *Advanced Engineering Informatics*, 48, S. 101302. https://doi.org/10.1016/j.aei.2021.101302. https://www.sciencedirect.com/science/article/pii/S1474034621000562.

Henrioud et al. 2002
Henrioud, J. M.; Relange, L. & Perrard, C. (2002), „Generation of precedence hypergraphs for assembly system design“, *IFAC Proceedings Volumes*, 35(1), S. 91–96.

Hotz et al. 2014
Hotz, L.; Felfernig, A.; Günter, A. & Tiihonen, J. (2014), „A Short History of Configuration Technologies“ in *Knowledge-Based Configuration,* Hrsg. A. Felfernig, L. Hotz, C. Bagley & J. Tiihonen, Elsevier Inc., Waltham, MA, USA, S. 9–19. http://dx.doi.org/10.1016/b978-0-12-415817-7.00002-5.

Huson & Linz 2018
Huson, D. H. & Linz, S. (2018), „Autumn Algorithm-Computation of Hybridization Networks for Realistic Phylogenetic Trees“, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 15(2), S. 398–410. https://doi.org/10.1109/tcbb.2016.2537326.

Hussong et al. 2021
Hussong, M.; Glatt, M.; Rüdiger-Flore, P.; Varshneya, S.; Liznerski, P.; Kloft, M. & Aurich, J. C. (2021), „Deep Learning zur Unterstützung der Arbeitsplanung“,

Zeitschrift für wirtschaftlichen Fabrikbetrieb, 116(10), S. 648–651. https://doi.org/10.1515/zwf-2021-0170. https://www.degruyter.com/document/doi/10.1515/zwf-2021-0170/html.

Hvam et al. 2008

Hvam, L.; Mortensen, N. H. & Riis, J. (2008), *Product customization,* Springer, Berlin, Heidelberg. ISBN: 9783540714491.

Hvam et al. 2019

Hvam, L.; Kristjansdottir, K.; Shafiee, S.; Mortensen, N. H. & Herbert-Hansen, Z. N. L. (2019), „The impact of applying product-modelling techniques in configurator projects“, *International Journal of Production Research*, 57(14), S. 4435–4450. https://doi.org/10.1080/00207543.2018.1436783.

Igenewari et al. 2019

Igenewari, V. R.; Skaf, Z. & Jennions, I. K. (2019), „A survey of flight anomaly detection methods: Challenges and opportunities“. *Proceedings of the 11th Annual Conference of the Prognostics and Health Management Society,* Hrsg. N. S. Clements, o. A., o. A. ISBN: 9781936263295. https://doi.org/10.36001/phmconf.2019.v11i1.898.

Ignatiev et al. 2018

Ignatiev, A.; Pereira, F.; Narodytska, N. & Marques-Silva, J. (2018), „A SAT-based approach to learn explainable decision sets“. *Automated Reasoning: 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings 9,* S. 627–645.

Ignatiev et al. 2021

Ignatiev, A.; Lam, E.; Stuckey, P. J. & Marques-Silva, J. (2021), „A Scalable Two Stage Approach to Computing Optimal Decision Sets“. *AAAI-21 Technical Tracks 5,* Hrsg. Q. Yang, AAAI Press, Palo Alto, S. 3806–3814. https://doi.org/10.1609/aaai.v35i5.16498. https://ojs.aaai.org/index.php/aaai/article/view/16498.

ISO 7573:2008-11

ISO 7573:2008-11 (2008), *Technische Produktdokumentation - Stücklisten,* ISO Internationale Organisation für Normung, Genf, Schweiz.

Jiao et al. 2000

Jiao, J.; Tseng, M. M.; Ma, Q. & Zou, Y. (2000), „Generic Bill-of-Materials-and-Operations for High-Variety Production Management“, *Concurrent Engineering: Research and Application*, 8(4), S. 297–322. https://doi.org/10.1177/1063293X0000800404.

Jiao et al. 2007

Jiao, J.; Zhang, L.; Pokharel, S. & He, Z. (2007), „Identifying generic routings for product families based on text mining and tree matching“, *Decision Support Systems*, 43(3), S. 866–883. https://doi.org/10.1016/j.dss.2007.01.001. https://www.sciencedirect.com/science/article/pii/S0167923607000085.

Jiménez 2013

Jiménez, P. (2013), „Survey on assembly sequencing: a combinatorial and geometrical perspective“, *Journal of Intelligent Manufacturing*, 24(2), S. 235–250. https://doi.org/10.1007/s10845-011-0578-5. https://link.springer.com/article/10.1007/s10845-011-0578-5.

Joo et al. 2001

Joo, J.; Sungsik Park & Hyunbo Cho (2001), „Adaptive and dynamic process planning using neural networks“, *International Journal of Production Research*, 39(13), S. 2923–2946. https://doi.org/10.1080/00207540110049034.

Jung 2022

Jung, A. (2022), *Machine learning*. *The basics,* Springer, Singapur. ISBN: 9789811681929.

Jungnickel 2013

Jungnickel, D. (2013), *Graphs, networks and algorithms,* Springer, Berlin, Heidelberg, New York, Dordrecht, London. ISBN: 9783642322778.

Júnior et al. 2023

Júnior, F.; Souza, A. C. & Rocha, T. A. (2023), „An Incremental MaxSAT-Based Model to Learn Interpretable and Balanced Classification Rules“. *Intelligent Systems. Proceedings of the 12th Brazilian Conference on Intelligent Systems, Part I,* Hrsg. M. C. Naldi & R. A. C. Bianchi, Springer Nature Switzerland, Cham, S. 227–242. https://doi.org/10.1007/978-3-031-45368-7_15. https://link.springer.com/chapter/10.1007/978-3-031-45368-7_15.

Kashkoush & ElMaraghy 2014

Kashkoush, M. & ElMaraghy, H. (2014), „Consensus tree method for generating master assembly sequence“, *Production Engineering*, 8(1-2), S. 233–242. https://doi.org/10.1007/s11740-013-0499-6. https://link.springer.com/article/10.1007/s11740-013-0499-6.

Kashkoush & ElMaraghy 2015

Kashkoush, M. & ElMaraghy, H. (2015), „Knowledge-based model for constructing master assembly sequence“, *Journal of Manufacturing Systems*, 34, S. 43–52. https://doi.org/10.1016/j.jmsy.2014.10.004. https://www.sciencedirect.com/science/article/pii/S0278612514001137.

Kashkoush & ElMaraghy 2016

Kashkoush, M. & ElMaraghy, H. (2016), „Product family formation by matching Bill-of-Materials trees“, *CIRP Journal of Manufacturing Science and Technology*, 12, S. 1–13. https://doi.org/10.1016/j.cirpj.2015.09.004. http://dx.doi.org/10.1016/j.cirpj.2015.09.004.

Klindworth et al. 2012

Klindworth, H.; Otto, C. & Scholl, A. (2012), „On a learning precedence graph concept for the automotive industry“, *European Journal of Operational Research*, 217(2), S. 259–269. https://doi.org/10.1016/j.ejor.2011.09.024. http://dx.doi.org/10.1016/j.ejor.2011.09.024.

Knebl 2021

Knebl, H. (2021), *Algorithmen und Datenstrukturen*. *Grundlagen und probabilistische Methoden für den Entwurf und die Analyse,* Springer Vieweg, Wiesbaden, Heidelberg. ISBN: 9783658327132.

Kneppelt 1984

Kneppelt, L. R. (1984), „Product structuring considerations for master production scheduling“, *Production and Inventory Management*, 25(1), S. 83–99.

Kotsiantis & Kanellopoulos 2006

Kotsiantis, S. & Kanellopoulos, D. (2006), „Discretization techniques: A recent survey“, *GESTS International Transactions on Computer Science and Engineering*, 32(1), S. 47–58.

Kourtis et al. 2024

Kourtis, G. K.; Hvam, L. & Johnsen, S. H. M. (2024), „Configuring System Products With CTO And ETO Components: Framework and Conceptual Modelling“. *11th International Conference on Customization and Personalization, MCP 2024. Proceedings,* Hrsg. Z. Anisic, Univ. of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbien, S. 152–160.

Kristjansdottir et al. 2018a

Kristjansdottir, K.; Shafiee, S.; Hvam, L.; Bonev, M. & Myrodia, A. (2018), „Return on investment from the use of product configuration systems – A case study“, *Computers in Industry*, 100, S. 57–69. https://doi.org/10.1016/j.compind.2018.04.003. http://dx.doi.org/10.1016/j.compind.2018.04.003.

Kristjansdottir et al. 2018b

Kristjansdottir, K.; Shafiee, S.; Hvam, L.; Forza, C. & Mortensen, N. H. (2018), „The main challenges for manufacturing companies in implementing and utilizing configurators“, *Computers in Industry*, 100, S. 196–211. https://doi.org/10.1016/j.compind.2018.05.001. http://dx.doi.org/10.1016/j.compind.2018.05.001.

Küchlin 2020

Küchlin, W. (2020), „Symbolische KI für die Produktkonfiguration in der Automobilindustrie“ in *Philosophisches Handbuch Künstliche Intelligenz,* Hrsg. K. Mainzer, Springer, Wiesbaden, S. 1–15. http://dx.doi.org/10.1007/978-3-658-23715-8_53-1.

Kumar & Gupta 2020

Kumar, P. & Gupta, A. (2020), „Active Learning Query Strategies for Classification, Regression, and Clustering: A Survey“, *Journal of Computer Science and Technology*, 35(4), S. 913–945. https://doi.org/10.1007/s11390-020-9487-4. https://link.springer.com/article/10.1007/s11390-020-9487-4.

Lakkaraju et al. 2016

Lakkaraju, H.; Bach, S. H. & Leskovec, J. (2016), „Interpretable Decision Sets“. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* Hrsg. B. Krishnapuram & M. Shah, ACM, New York, S. 1675–1684. https://doi.org/10.1145/2939672.2939874.

Lawless et al. 2023

Lawless, C.; Dash, S.; Gunluk, O. & Wei, D. (2023), „Interpretable and Fair

Boolean Rule Sets via Column Generation“, *Journal of Machine Learning Research*, 24, S. 1–50. https://doi.org/10.48550/arXiv.2111.08466. https://www.jmlr.org/papers/v24/22-0880.html.

Li 2024

Li, H. (2024), *Machine Learning Methods,* Springer Nature Singapore, Singapur. ISBN: 9789819939169.

Li & van Leeuwen 2023

Li, Z. & van Leeuwen, M. (2023), „Explainable contextual anomaly detection using quantile regression forests“, *Data Mining and Knowledge Discovery*, 37(6), S. 2517–2563. https://doi.org/10.1007/s10618-023-00967-z. https://link.springer.com/article/10.1007/s10618-023-00967-z.

Ling & Du 2008

Ling, C. X. & Du, J. (2008), „Active learning with direct query construction“. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining,* Hrsg. Y. Li, ACM, New York, S. 480–487. https://doi.org/10.1145/1401890.1401950.

Liu et al. 2016

Liu, H.; Gegov, A. & Cocea, M. (2016), „Complexity Control in Rule Based Models for Classification in Machine Learning Context“. *Advances in Computational Intelligence Systems. Contributions Presented at the 16th UK Workshop on Computational Intelligence,* Hrsg. J. Kacprzyk, Springer, Cham, S. 125–143. https://doi.org/10.1007/978-3-319-46562-3_9. https://link.springer.com/chapter/10.1007/978-3-319-46562-3_9.

Lu et al. 2020

Lu, S.; Liu, L.; Li, J.; Le, T. D. & Liu, J. (2020), „LoPAD: A Local Prediction Approach to Anomaly Detection“. *Advances in Knowledge Discovery and Data Mining. 24th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part II,* Hrsg. H. W. Lauw, R. C.-W. Wong, A. Ntoulas, E.-P. Lim, S.-K. Ng & S. J. Pan, Springer, Cham, S. 660–673. https://doi.org/10.1007/978-3-030-47436-2_50. https://link.springer.com/chapter/10.1007/978-3-030-47436-2_50.

Mahalle 2022

Mahalle, P. N. (2022), *Foundations of Data Science for Engineering Problem Solving,* Springer, Singapur. ISBN: 9789811651595.

Mailharro 1998

Mailharro, D. (1998), „A classification and constraint-based framework for configuration“, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AI EDAM*, 12(4), S. 383–397. https://doi.org/10.1017/s0890060498124101.

Malioutov & Meel 2018

Malioutov, D. & Meel, K. S. (2018), „MLIC: A MaxSAT-Based Framework for Learning Interpretable Classification Rules“. *Proceedings of the 24th International Conference on Principles and Practice of Constraint Programming,* Hrsg. J. Hooker, Springer, Cham, S. 312–327. https://doi.org/10.1007/978-3-319-98334-9_21. https://link.springer.com/chapter/10.1007/978-3-319-98334-9_21.

Malmi et al. 2015

Malmi, E.; Tatti, N. & Gionis, A. (2015), „Beyond rankings: comparing directed acyclic graphs“, *Data Mining and Knowledge Discovery*, 29(5), S. 1233–1257. https://doi.org/10.1007/s10618-015-0406-1.

Medeiros et al. 2016

Medeiros, F.; Kästner, C.; Ribeiro, M.; Gheyi, R. & Apel, S. (2016), „A comparison of 10 sampling algorithms for configurable systems“. *Proceedings of the 38th International Conference on Software Engineering,* Hrsg. L. Dillon, ACM, New York, S. 643–654. https://doi.org/10.1145/2884781.2884793.

Méndez-Díaz & Zabala 2006

Méndez-Díaz, I. & Zabala, P. (2006), „A Branch-and-Cut algorithm for graph coloring“, *Discrete Applied Mathematics*, 154(5), S. 826–847. https://doi.org/10.1016/j.dam.2005.05.022.

Meseguer & Preece 1995

Meseguer, P. & Preece, A. D. (1995), „Verification and validation of knowledge-based systems with formal specifications“, *The Knowledge Engineering Review*, 10(4), S. 331–343. https://doi.org/10.1017/s0269888900007542.

Mînzu & Bratcu 1999

Mînzu, V. & Bratcu, A. (1999), „Precedence graphs generation using assembly

sequences“. https://www.researchgate.net/profile/viorel-minzu/publication/274457768_precedence_graphs_generation_using_assembly_sequences.

Mitchell 1977

Mitchell, T. M. (1977), „Version Spaces: A Candidate Elimination Approach to Rule Learning“. *Proceedings of the 5th international joint conference on Artificial intelligence - Volume 1,* Hrsg. R. Reddy, Morgan Kaufmann Publishers, San Francisco, S. 305–310. https://www.semanticscholar.org/paper/Version-Spaces%3A-A-Candidate-Elimination-Approach-to-Mitchell/19bfa432237dc1bd82113774727fe0307005e430.

Miyamoto 2022

Miyamoto, S. (2022), *Theory of agglomerative hierarchical clustering,* Springer, Singapur. ISBN: 9789811904196.

Moussa & ElMaraghy 2018

Moussa, M. & ElMaraghy, H. (2018), „Master Assembly Network Generation“. *Procedia CIRP 72. 51st CIRP Conference on Manufacturing Systems,* Hrsg. L. Wang, Elsevier, Amsterdam, S. 756–761. https://doi.org/10.1016/j.procir.2018.03.089. https://www.sciencedirect.com/science/article/pii/S2212827118301938.

Moussa & ElMaraghy 2019

Moussa, M. & ElMaraghy, H. (2019), „Master assembly network for alternative assembly sequences“, *Journal of Manufacturing Systems*, 51, S. 17–28. https://doi.org/10.1016/j.jmsy.2019.02.001.

Myrodia et al. 2018

Myrodia, A.; Randrup, T. & Hvam, L. (2018), „Configuration Lifecycle Management – Future of Product Configurators“. *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM),* Hrsg. A. K. Varma, IEEE, New York, S. 1315–1319. ISBN: 978-1-5386-6786-6. https://doi.org/10.1109/ieem.2018.8607471. http://dx.doi.org/10.1109/ieem.2018.8607471.

Natarajan & Gokulachandran 2020

Natarajan, K. K. & Gokulachandran, J. (2020), „Artificial Neural Network Based Machining Operation Selection for Prismatic Components“, *International Journal on Advanced Science, Engineering and Information Technology*, 10(2), S. 618–628.

Navaei & ElMaraghy 2018

Navaei, J. & ElMaraghy, H. (2018), „Optimal operations sequence retrieval from master operations sequence for part/product families“, *International Journal of Production Research*, 56(1-2), 140-163. https://doi.org/10.1080/00207543.2017.1391417.

Nijenhuis & Wilf 1975

Nijenhuis, A. & Wilf, H. S. (1975), *Combinatorial algorithms,* Acad. Press, New York. ISBN: 0125192509.

Noto et al. 2010

Noto, K.; Brodley, C. & Slonim, D. (2010), „Anomaly detection using an ensemble of feature models“. *2010 ieee international conference on data mining,* S. 953–958.

Oddsson & Ladeby 2014

Oddsson, G. & Ladeby, K. R. (2014), „From a literature review of product configuration definitions to a reference framework“, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AI EDAM*, 28(4), S. 413–428. https://doi.org/10.1017/S0890060413000620. https://www.cambridge.org/core/journals/ai-edam/article/from-a-literature-review-of-product-configuration-definitions-to-a-reference-framework/EC52A274C4D916395A6B696EDAF918EF.

Osisanwo et al. 2017

Osisanwo, F.; Akinsola, J.; Awodele, O.; Hinmikaiye, J. O.; Olakanmi, O. & Akinjobi, J. (2017), „Supervised Machine Learning Algorithms: Classification and Comparison“, *International Journal of Computer Trends and Technology*, 48(3), S. 128–138. https://doi.org/10.14445/22312803/ijctt-v48p126.

Otto & Otto 2014

Otto, C. & Otto, A. (2014), „Multiple-source learning precedence graph concept for the automotive industry“, *European Journal of Operational Research*, 234(1), S. 253–265. https://doi.org/10.1016/j.ejor.2013.09.034. https://www.sciencedirect.com/science/article/pii/S0377221713007893.

Ouyang & Chou 2020

Ouyang, R. & Chou, C.-A. (2020), „Integrated optimization model and algorithm for pattern generation and selection in logical analysis of data“, *Computers &*

Operations Research, 124. https://doi.org/10.1016/j.cor.2020.105049. https://www.sciencedirect.com/science/article/pii/S0305054820301660.

Patil et al. 2012

Patil, N.; Lathi, R. & Chitre, V. (2012), „Comparison of C5. 0 & CART classification algorithms using pruning technique“, *International Journal of Engineering Research and Technology*, 1(4), S. 1–5.

Paulheim & Meusel 2015

Paulheim, H. & Meusel, R. (2015), „A decomposition of the outlier detection problem into a set of supervised learning problems“, *Machine Learning*, 100(2-3), S. 509–531. https://doi.org/10.1007/s10994-015-5507-y. https://link.springer.com/article/10.1007/s10994-015-5507-y.

Peffers et al. 2007

Peffers, K.; Tuunanen, T.; Rothenberger, M. A. & Chatterjee, S. (2007), „A Design Science Research Methodology for Information Systems Research“, *Journal of Management Information Systems*, 24(3), S. 45–77. https://doi.org/10.2753/MIS0742-1222240302.

Potdar et al. 2017

Potdar, K.; S., T. & D., C. (2017), „A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers“, *International Journal of Computer Applications*, 175(4), S. 7–9. https://doi.org/10.5120/ijca2017915495. https://www.researchgate.net/profile/kedar-potdar-2/publication/320465713_a_comparative_study_of_categorical_variable_encoding_techniques_for_neural_network_classifiers.

Prote et al. 2017

Prote, J.-P.; Luckert, M. & Hünnekes, P. (2017), „Automatisierung in der Arbeitsplanung“, *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 112(12), S. 827–830.

Qamar & Raza 2020

Qamar, U. & Raza, M. S. (2020), *Data Science Concepts and Techniques with Applications,* Springer, Singapur. ISBN: 9789811561320.

Ram Babu et al. 2014

Ram Babu, D.; Lenin, A. & Bhaskar, G. B. (2014), „Advanced Product Configuration in Manufacturing Using Enterprise Resource Planning Variant Configuration

with Optimization in Manufacturing and Assembly Processes“, *Applied Mechanics and Materials*, 591, S. 94–97. https://doi.org/10.4028/www.scientific.net/AMM.591.94. https://www.scientific.net/amm.591.94.

Rasmussen et al. 2020

Rasmussen, J.; Hvam, L.; Kristjansdottir, K. & Mortensen, N. (2020), „Guidelines for Structuring Object-Oriented Product Configuration Models in Standard Configuration Software“, *JUCS - Journal of Universal Computer Science*, 26(3), S. 374–401. https://doi.org/10.3897/jucs.2020.020. https://lib.jucs.org/article/24005/.

Ren et al. 2021

Ren, J.; Xia, F.; Chen, X.; Liu, J.; Hou, M.; Shehzad, A.; Sultanova, N. & Kong, X. (2021), „Matching Algorithms: Fundamentals, Applications and Challenges“, *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(3), S. 332–350. https://doi.org/10.1109/tetci.2021.3067655.

Rijayanti et al. 2023

Rijayanti, R.; Hwang, M. & Jin, K. (2023), „Detection of Anomalous Behavior of Manufacturing Workers Using Deep Learning-Based Recognition of Human–Object Interaction“, *Applied Sciences*, 13(15). https://doi.org/10.3390/app13158584.

Ris-Ala 2023

Ris-Ala, R. (2023), *Fundamentals of Reinforcement Learning,* Springer Nature Switzerland; Imprint Springer, Cham. ISBN: 9783031373442.

Rocks & Mehta 2022

Rocks, J. W. & Mehta, P. (2022), „Memorizing without overfitting: Bias, variance, and interpolation in overparameterized models“, *Physical review research*, 4(1). https://doi.org/10.1103/PhysRevResearch.4.013201.

Romanowski & Nagi 2004

Romanowski, C. J. & Nagi, R. (2004), „A Data Mining Approach to Forming Generic Bills of Materials in Support of Variant Design Activities“, *Journal of Computing and Information Science in Engineering*, 4(4), S. 316–328. https://doi.org/10.1115/1.1812556.

Rönnberg & Larsson 2014

Rönnberg, E. & Larsson, T. (2014), „All-integer column generation for set

partitioning: Basic principles and extensions“, *European Journal of Operational Research*, 233(3), S. 529–538. https://doi.org/10.1016/j.ejor.2013.08.036.

Rudin 2019

Rudin, C. (2019), „Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead“, *Nature Machine Intelligence*, 1(5), S. 206–215. https://doi.org/10.1038/s42256-019-0048-x. https://www.nature.com/articles/s42256-019-0048-x.

Rudin et al. 2022

Rudin, C.; Chen, C.; Chen, Z.; Huang, H.; Semenova, L. & Zhong, C. (2022), „Interpretable machine learning: Fundamental principles and 10 grand challenges“, *Statistics Surveys*, 16, S. 1–85. https://doi.org/10.1214/21-SS133. https://projecteuclid.org/journals/statistics-surveys/volume-16/issue-none/interpretable-machine-learning-fundamental-principles-and-10-grand-challenges/10.1214/21-ss133.short.

Safaei & Beigy 2007

Safaei, J. & Beigy, H. (2007), „Quine-McCluskey Classification“. *Proceedings of the 2007 IEEE/ACS International Conference on Computer Systems and Applications,* Hrsg. M. S. Obaidat, IEEE, New York, S. 404–411. https://doi.org/10.1109/aiccsa.2007.370913. http://dx.doi.org/10.1109/aiccsa.2007.370913.

Salvador & Forza 2007

Salvador, F. & Forza, C. (2007), „Principles for efficient and effective sales configuration design“, *International Journal of Mass Customisation*, 2(1-2), S. 114–127. https://doi.org/10.1504/IJMASSC.2007.012816.

Sasao 2023

Sasao, T. (2023), „Easily Reconstructable Logic Functions“. *Proceedings of the 2023 IEEE 53rd International Symposium on Multiple-Valued Logic (ISMVL),* Hrsg. S. Nagayama & R. Miyauchi, IEEE, New York, S. 12–17. https://doi.org/10.1109/ismvl57333.2023.00014.

Satopaa et al. 2011

Satopaa, V.; Albrecht, J.; Irwin, D. & Raghavan, B. (2011), „Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior“. *Proceedings of the 31st*

International Conference on Distributed Computing Systems Workshops, Hrsg. H. Ho & D. Xu, IEEE, New York, S. 166–171. https://doi.org/10.1109/icdcsw.2011.20.

Schaller et al. 2021

Schaller, D.; Hellmuth, M. & Stadler, P. F. (2021), „A simpler linear-time algorithm for the common refinement of rooted phylogenetic trees on a common leaf set“, *Algorithms for Molecular Biology*, 16. https://doi.org/10.1186/s13015-021-00202-8. https://almob.biomedcentral.com/articles/10.1186/s13015-021-00202-8.

Schenk 2014

Schenk, M. (2014), *Fabrikplanung und Fabrikbetrieb*. *Methoden für die wandlungsfähige, vernetzte und ressourceneffiziente Fabrik,* Springer Vieweg, Berlin, Heidelberg. ISBN: 9783642054594.

Schierholt 2001

Schierholt, K. (2001), „Process configuration — combining the principles of product configuration and process planning“, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AI EDAM*, 15(5), S. 411–424. https://doi.org/10.1017/S0890060401155046. https://doi.org/10.1017/S0890060401155046.

Schuh et al. 2017

Schuh, G.; Prote, J.-P.; Luckert, M. & Hünnekes, P. (2017), „Knowledge Discovery Approach for Automated Process Planning“, *Procedia CIRP*, 63, S. 539–544. https://doi.org/10.1016/j.procir.2017.03.092. http://dx.doi.org/10.1016/j.procir.2017.03.092.

Schuh et al. 2019

Schuh, G.; Prote, J.-P. & Hünnekes, P. (2019), „Data mining methods for macro level process planning“. *Procedia CIRP 88. 13th CIRP Conference on Intelligent Computation in Manufacturing Engineering,* Hrsg. R. Teti & D. M. D'Addona, Elsevier, Amsterdam, S. 48–53. https://doi.org/10.1016/j.procir.2020.05.009. http://dx.doi.org/10.1016/j.procir.2020.05.009.

Shafiee et al. 2017

Shafiee, S.; Kristjansdottir, K.; Hvam, L. & Forza, C. (2017), „How to scope configuration projects and manage the knowledge they require“, *Journal of Knowledge Management*, 22(5), S. 982–1014. https://doi.org/10.1108/JKM-01-2017-0017. https://www.emerald.com/insight/content/doi/10.1108/jkm-01-2017-0017/full/pdf.

Shafiee et al. 2020

Shafiee, S.; Wautelet, Y.; Hvam, L.; Sandrin, E. & Forza, C. (2020), „Scrum versus Rational Unified Process in facing the main challenges of product configuration systems development“, *Journal of Systems and Software*, 170. https://doi.org/10.1016/j.jss.2020.110732.

Shao et al. 2006

Shao, X.-Y.; Wang, Z.-H.; Li, P.-G. & Feng, C.-X. J. (2006), „Integrating data mining and rough set for customer group-based discovery of product configuration rules“, *International Journal of Production Research*, 44(14), S. 2789–2811. https://doi.org/10.1080/00207540600675777.

Sinz, C. 2004

Sinz, C. (2004), *Verifikation regelbasierter Konfigurationssysteme*. Dissertation, Universität Tübingen, Tübingen.

Sipes et al. 2014

Sipes, T.; Jiang, S.; Moore, K.; Li, N.; Karimabadi, H. & Barr, J. R. (2014), „Anomaly Detection in Healthcare: Detecting Erroneous Treatment Plans in Time Series Radiotherapy Data“, *International Journal of Semantic Computing*, 8(3), S. 257–278. https://doi.org/10.1142/S1793351X1440008X.

Skiena 2020

Skiena, S. S. (2020), *The Algorithm Design Manual,* Springer International Publishing; Imprint Springer, Cham. ISBN: 9783030542566.

Sluban et al. 2010

Sluban, B.; Gamberger, D. & Lavra, N. (2010), „Advances in Class Noise Detection“ in *ECAI 2010,* IOS Press, S. 1105–1106. https://ebooks.iospress.nl/doi/10.3233/978-1-60750-606-5-1105.

Sluban et al. 2014

Sluban, B.; Gamberger, D. & Lavrač, N. (2014), „Ensemble-based noise detection: noise ranking and visual performance evaluation“, *Data Mining and Knowledge Discovery*, 28(2), S. 265–303. https://doi.org/10.1007/s10618-012-0299-1. https://link.springer.com/article/10.1007/s10618-012-0299-1.

Soos et al. 2020

Soos, M.; Gocht, S. & Meel, K. S. (2020), „Tinted, Detached, and Lazy CNF-XOR

Solving and Its Applications to Counting and Sampling“. *Computer Aided Verification. 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part I,* Hrsg. S. K. Lahiri & C. Wang, Springer, Cham, S. 463–484. https://doi.org/10.1007/978-3-030-53288-8_22. https://link.springer.com/chapter/10.1007/978-3-030-53288-8_22.

Spiegelhalter et al. 2002

Spiegelhalter, D. J.; Best, N. G.; Carlin, B. P. & van der Linde, A. (2002), „Bayesian Measures of Model Complexity and Fit“, *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 64(4), S. 583–639. https://doi.org/10.1111/1467-9868.00353. https://academic.oup.com/jrsssb/article/64/4/583/7098621?login=true.

Su et al. 2016

Su, G.; Wei, D.; Varshney, K. R. & Malioutov, D. M. (2016), „Learning sparse two-level boolean rules“. *Proceedings ot the 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP),* Hrsg. F. A. N. Palmieri, IEEE, New York. https://doi.org/10.1109/mlsp.2016.7738856.

Tangirala 2020

Tangirala, S. (2020), „Evaluating the Impact of GINI Index and Information Gain on Classification using Decision Tree Classifier Algorithm“, *International Journal of Advanced Computer Science and Applications*, 11(2). https://doi.org/10.14569/ijacsa.2020.0110277.

Tarjan 1973

Tarjan, R. (1973), „Enumeration of the elementary circuits of a directed graph“, *SIAM Journal on Computing*, 2(3), S. 211–216. https://doi.org/10.1137/0202017.

Tharwat & Schenck 2023

Tharwat, A. & Schenck, W. (2023), „A Survey on Active Learning: State-of-the-Art, Practical Challenges and Research Directions“, *Mathematics*, 11(4). https://doi.org/10.3390/math11040820. https://www.mdpi.com/2227-7390/11/4/820.

Tidake & Sane 2018

Tidake, V. S. & Sane, S. S. (2018), „Multi-label Classification: a survey“, *International Journal of Engineering & Technology*, 7(4.19), S. 1045–1054.

https://doi.org/10.14419/ijet.v7i4.19.28284. https://www.researchgate.net/profile/vaishali-tidake-2/publication/334456720_multi-label_classification_a_survey.

Tidstam et al. 2016

Tidstam, A.; Malmqvist, J.; Voronov, A.; Åkesson, K. & Fabian, M. (2016), „Formulating constraint satisfaction problems for the inspection of configuration rules“, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AI EDAM*, 30(3), S. 313–328. https://doi.org/10.1017/S0890060415000487.

Tiihonen et al. 1998

Tiihonen, J.; Lehtonen, T.; Soininen, T.; Puikkinen, A.; Sulonen, R. & Riitahuhta, A. (1998), „Modeling Configurable Product Families“. *Proceedings of the 4th WDK Workshop on Product Structuring,* Hrsg. Marcel Tichem, Mogens Myrup Andreasen, Alex Duffy, Delft University of Technology, Delft, S. 29–50.

Trentin et al. 2012a

Trentin, A.; Perin, E. & Forza, C. (2012), „Product configurator impact on product quality“, *International Journal of Production Economics*, 135(2), S. 850–859. https://doi.org/10.1016/j.ijpe.2011.10.023. https://www.sciencedirect.com/science/article/pii/S0925527311004531.

Trentin et al. 2012

Trentin, A.; Perin, E. & Forza, C. (2012), „Sales Configurator Capabilities to Prevent Product Variety From Backfiring“. *CONFWS'12: Proceedings of the 2012 International Conference on Configuration,* Hrsg. W. Mayer & P. Albert, CEUR, Aachen, S. 47–54.

Trentin et al. 2014

Trentin, A.; Perin, E. & Forza, C. (2014), „Increasing the consumer-perceived benefits of a mass-customization experience through sales-configurator capabilities“, *Computers in Industry*, 65(4), S. 693–705. https://doi.org/10.1016/j.compind.2014.02.004.

Triantaphyllou 2006

Triantaphyllou, E. (2006), „The One Clause at a Time (OCAT) Approach to Data Mining and Knowledge Discovery“ in *Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques,* Springer, Boston, MA, S. 45–87. https://link.springer.com/chapter/10.1007/0-387-34296-6_2.

van der Aalst 2022

van der Aalst, W. (2022), „Foundations of process discovery“ in *Process mining handbook,* Hrsg. van der Aalst, Wil M. P. & J. Carmona, Springer Nature Switzerland, Cham, Schweiz, S. 37–75.

vom Brocke et al. 2020

vom Brocke, J.; Hevner, A. & Maedche, A. (2020), „Introduction to Design Science Research“ in *Design Science Research. Cases,* Hrsg. J. vom Brocke, A. Hevner & A. Maedche, Springer Nature Switzerland AG, Cham, Schweiz, S. 1–13. https://link.springer.com/chapter/10.1007/978-3-030-46781-4_1.

Voronov, A. 2013

Voronov, A. (2013), *On formal methods for large-scale product configuration*. Dissertation, Chalmers University of Technology, Göteborg, Schweden.

Wagstaff et al. 2013

Wagstaff, K.; Lanza, N.; Thompson, D.; Dietterich, T. & Gilmore, M. (2013), „Guiding Scientific Discovery with Explanations Using DEMUD“. *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence,* Hrsg. S. Kambhampati, AAAI Press, Palo Alto, S. 905–911. https://doi.org/10.1609/aaai.v27i1.8561. https://ojs.aaai.org/index.php/aaai/article/view/8561.

Walter et al. 2016

Walter, R.; Kübart, T. & Küchlin, W. (2016), „Optimal Coverage in Automotive Configuration“. *Mathematical Aspects of Computer and Information Sciences,* Hrsg. I. S. Kotsireas, S. M. Rump & C. K. Yap, Springer, Cham, Heidelberg, S. 611–626. https://doi.org/10.1007/978-3-319-32859-1_52. https://link.springer.com/chapter/10.1007/978-3-319-32859-1_52.

Wang et al. 2015

Wang, L.; Hu, X.; Yuan, B. & Lu, J. (2015), „Active learning via query synthesis and nearest neighbour search“, *Neurocomputing*, 147, S. 426–434. https://doi.org/10.1016/j.neucom.2014.06.042.

Wang et al. 2017

Wang, L.; Zhong, S.-S. & Zhang, Y.-J. (2017), „Process configuration based on generative constraint satisfaction problem“, *Journal of Intelligent Manufacturing*, 28(4), S. 945–957. https://doi.org/10.1007/s10845-014-1031-3. https://link.springer.com/article/10.1007/s10845-014-1031-3.

Wang et al. 2023

Wang, Z.; Ge, W.; Qiu, L.; Zhang, S.; Zhou, J.; Hu, K. & Fang, N. (2023), „Customized Product Configuration Rule Intelligent Extraction and Dynamic Updating Method Based on the Least Recently Used Dynamic Decision Tree“, *Journal of Mechanical Design*, 145(5). https://doi.org/10.1115/1.4056498. https://asmedigitalcollection.asme.org/mechanicaldesign/article/145/5/051701/1154774/Customized-Product-Configuration-Rule-Intelligent.

Westkämper 2006

Westkämper, E. (2006), *Einführung in die Organisation der Produktion,* Springer, Berlin, Heidelberg. ISBN: 3540260390.

Wiendahl 2019

Wiendahl, H.-P. (2019), *Betriebsorganisation für Ingenieure,* Carl Hanser Verlag, München. ISBN: 9783446460614.

Wilhelm 2001

Wilhelm, W. E. (2001), „A Technical Review of Column Generation in Integer Programming“, *Optimization and Engineering*, 2(2), S. 159–200. https://doi.org/10.1023/A:1013141227104. https://link.springer.com/article/10.1023/a:1013141227104.

Wu et al. 2019

Wu, H.; Nie, C.; Petke, J.; Jia, Y. & Harman, M. (07. August 2019), *A Survey of Constrained Combinatorial Testing*.

Wu et al. 2020

Wu, J.; Sheng, V. S.; Zhang, J.; Li, H.; Dadakova, T.; Swisher, C. L.; Cui, Z. & Zhao, P. (2020), „Multi-Label Active Learning Algorithms for Image Classification: Overview and Future Promise“, *ACM Computing Surveys*, 53(2), S. 1–35. https://doi.org/10.1145/3379504.

Xie et al. 2021

Xie, Y.; Zhang, H.; Zhang, B.; Babar, M. A. & Lu, S. (2021), „LogDP: Combining Dependency and Proximity for Log-Based Anomaly Detection“. *Service-Oriented Computing. 19th International Conference on Service-Oriented Computing, ICSOC 2021,* Hrsg. H. Hacid, O. Kao, M. Mecella, N. Moha & H. Paik, Springer, Cham, S. 708–716. https://doi.org/10.1007/978-3-030-91431-8_47. https://link.springer.com/chapter/10.1007/978-3-030-91431-8_47.

Xu et al. 2016

Xu, S.; Qiao, X.; Zhu, L.; Zhang, Y.; Xue, C. & Li, L. (2016), „Reviews on Determining the Number of Clusters“, *Applied Mathematics & Information Sciences*, 10(4), S. 1493–1512. https://doi.org/10.18576/amis/100428.

Xuelei Hu et al. 2012

Xuelei Hu; Liantao Wang & Bo Yuan (2012), „Querying representative points from a pool based on synthesized queries“. *The 2012 International Joint Conference on Neural Networks (IJCNN),* IEEE. https://doi.org/10.1109/ijcnn.2012.6252607.

Yang et al. 2021

Yang, F.; He, K.; Yang, L.; Du, H.; Yang, J.; Yang, B. & Sun, L. (2021), „Learning Interpretable Decision Rule Sets: A Submodular Optimization Approach“. *NIPS'21: Proceedings of the 35th International Conference on Neural Information Processing Systems,* Hrsg. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang & J. Wortman Vaughan, Curran Associates, Red Hook, S. 27890–27902. https://doi.org/10.48550/arXiv.2206.03718.

Yu et al. 2020

Yu, J.; Ignatiev, A.; Stuckey, P. J. & Le Bodic, P. (2020), „Computing optimal decision sets with SAT“. *Principles and Practice of Constraint Programming: 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings 26,* S. 952–970.

Zhang et al. 2008

Zhang, L.; Jiao, J. & Pokharel, S. (2008), „Process Platform-Based Production Configuration“. *Proceedings of the ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 2, Parts A and B,* Hrsg. o. A., ASME, S. 1091–1102. https://doi.org/10.1115/DETC2005-85559.

Zhang et al. 2010

Zhang, L. L.; Lee, C. K. & Xu, Q. (2010), „Towards product customization: An integrated order fulfillment system“, *Computers in Industry*, 61(3), S. 213–222. https://doi.org/10.1016/j.compind.2009.09.003. https://www.sciencedirect.com/science/article/pii/S0166361509001614.

Zhang 2012

Zhang, L. L. (2012), „Constructing generic processes based on tree unification for

process family planning“. *Proceedings of the 2012 IEEE International Conference on Industrial Engineering and Engineering Management,* Hrsg. D. Berg, IEEE, New York, S. 433–437.

Zhang et al. 2013

Zhang, L. L.; Vareilles, E. & Aldanondo, M. (2013), „Generic bill of functions, materials, and operations for SAP² configuration“, *International Journal of Production Research*, 51(2), S. 465–478. https://doi.org/10.1080/00207543.2011.652745.

Zhang et al. 2015

Zhang, L. L.; Helo, P. T.; Kumar, A. & You, X. (2015), „Implications of product configurator applications: An empirical study“. *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM),* Hrsg. T. L. MAGNANTI, IEEE, S. 57–61. https://doi.org/10.1109/ieem.2015.7385608.

Zhang et al. 2020

Zhang, L. L.; Lee, C. K. M. & Akhtar, P. (2020), „Towards customization: Evaluation of integrated sales, product, and production configuration“, *International Journal of Production Economics*, 229. https://doi.org/10.1016/j.ijpe.2020.107775. http://dx.doi.org/10.1016/j.ijpe.2020.107775.

Zhang & Rodrigues 2009

Zhang, L. L. & Rodrigues, B. (2009), „A tree unification approach to constructing generic processes“, *IIE Transactions*, 41(10), S. 916–929. https://doi.org/10.1080/07408170903026049.

Abbildungsverzeichnis

Tabellenverzeichnis

Liste eigener Veröffentlichungen

Albers, A.; Lanza, G.; Klippert, M.; Schäfer, L.; Frey, A.; Hellweg, F.; Müller-Welt, P.; Schöck, M.; Krahe, C.; Nowoseltschenko, K. & Rapp, S. (2022), „Product-Production-CoDesign: An Approach on Integrated Product and Production Engineering Across Generations and Life Cycles“. *Procedia CIRP 109. 32nd CIRP Design Conference,* Hrsg. Nabil Anwer. 32nd CIRP Design Conference. 28.-30.03.2022. Paris, Frankreich, Elsevier, Amsterdam, S. 167–172. https://doi.org/10.1016/j.procir.2022.05.231.

Frey, A. M. & Lanza, G. (2021), „Adaptive Manufacturing Based on Active Sampling for Multi-component Individual Assembly“. *Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems. Proceedings of the 8th Changeable, Agile, Reconfigurable and Virtual Production Conference (CARV2021) and the 10th World Mass Customization & Personalization Conference (MCPC2021),* Hrsg. A.-L. Andersen, R. Andersen, T. D. Brunoe, M. S. S. Larsen, K. Nielsen, A. Napoleone & S. Kjeldgaard. 8th Changeable, Agile, Reconfigurable and Virtual Production Conference (CARV2021) and the 10th World Mass Customization & Personalization Conference (MCPC2021). 01.-02.11.2021. Aalborg, Dänemark, S. 372–380. https://doi.org/10.1007/978-3-030-90700-6_42.

Frey, A. M.; May, M. C. & Lanza, G. (2023), „Creation and validation of systems for product and process configuration based on data analysis“, *Production Engineering*, 17(2), S. 263–277. https://doi.org/10.1007/s11740-022-01176-1.

Frey, A. M.; Maul, T.; Hörsting, R.; Stindt, J.; May, M. C.; Mark, P. & Lanza, G. (2025), „A method for considering aleatory and epistemic uncertainties as well as product variance in discrete event simulation of production systems“, *Journal of Manufacturing Systems*, *82*, S. 547–560. https://doi.org/10.1016/j.jmsy.2025.06.007.

Frey, A. M.; Pampus, O.; Stadler, F.; Erdler, G.-A. & Lanza, G. (2022), „Anwendung von Datenanalyse im Qualitätsmanagement (Application of Data Analysis in Quality Management)“, *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 117(4), S. 182–186. https://doi.org/10.1515/zwf-2022-1043.

Frey, A. M.; Stindt, J.; Lanza, G. & Mark, P. (2021), „Geometrische Bewertung und Optimierung der Modulanordnung in Tragwerken – Ein Beitrag zur adaptiven Fertigung im Bauwesen“, *Bautechnik*, 98(9), S. 662–670. https://doi.org/10.1002/bate.202100027.

Mark, P.; Lanza, G.; Lordick, D.; Albers, A.; König, M.; Borrmann, A.; Stempniewski, L.; Forman, P.; Frey, A. M.; Renz, R.; Manny, A. & Stindt, J. (2021), „Industrializing precast productions - Adaptive modularized constructions made in a flux“, *Civil Engineering Design*, 3(3), S. 87–98. https://doi.org/10.1002/cend.202100019.

Mark, P.; Lanza, G.; Lordick, D.; Albers, A.; König, M.; Borrmann, A.; Stempniewski, L.; Forman, P.; Frey, A. M.; Renz, R. & Manny, A. (2021), „Vom Handwerk zur individualisierten Serienfertigung - Schwerpunkt adaptive Modulbauweisen mit Fließfertigungsmethoden“, *Bautechnik*, 98(3), S. 243–256. https://doi.org/10.1002/bate.202000110.

May, M. C.; Kiefer, L.; Frey, A.; Duffie, N. A. & Lanza, G. (2023), „Applying frequency based forecasting for resource allocation“. *Procedia CIRP 120. 56th CIRP Conference on Manufacturing Systems, CIRP CMS ‘23,* Hrsg. K. Mpofu, N. Sacks & O. Damm. 56th CIRP Conference on Manufacturing Systems, CIRP CMS ‘23. 24.-26.10.2023. Cape Town, Südafrika, Elsevier, Amsterdam, S. 147–152. https://doi.org/10.1016/j.procir.2023.08.027.

May, M. C.; Kiefer, L.; Frey, A.; Duffie, N. A. & Lanza, G. (2023), „Solving sustainable aggregate production planning with model predictive control“, CIRP Annals, 72(1), S. 421–424. https://doi.org/10.1016/j.cirp.2023.04.023.

May, M. C.; Schäfer, L.; Frey, A.; Krahe, C. & Lanza, G. (2023), „Towards Product-Production-CoDesign for the Production of the Future“. *Procedia CIRP 119. 33rd CIRP Design Conference,* Hrsg. A. Liu & S. Kara. 33rd CIRP Design Conference. 17.-23.05.2023. Sydney, Australien, Amsterda, Elsevier, S. 944–949. https://doi.org/10.1016/j.procir.2023.02.172.

Stindt, J.; Frey, A. M.; Forman, P.; Lanza, G. & Mark, P. (2022), „Genauigkeitsgrenzen modularer Betontragwerke – Teil 1: Beschreibung von geometrischen Abweichungen infolge Schwinden“, *Beton- und Stahlbetonbau*, 117(5), S. 296–309. https://doi.org/10.1002/best.202200010.

Stindt, J.; Frey, A. M.; Forman, P.; Lanza, G. & Mark, P. (2022), „Genauigkeitsgrenzen modularer Betontragwerke – Teil 2: Probabilistische Bewertung der Montage mit Schraubenverbindung“, *Beton- und Stahlbetonbau*, 117(5), S. 310–323. https://doi.org/10.1002/best.202200011.

Stindt, J.; Frey, A. M.; Forman, P.; Lanza, G. & Mark, P. (2023), „Toleranzfreie Montage modularer Betontragwerke - Optimierung der Modulanordnung mit Metaheuristiken“, *Bautechnik*, 100(11), S. 674–688. https://doi.org/10.1002/bate.202300052.

Stindt, J.; Frey, A. M.; Forman, P.; Mark, P. & Lanza, G. (2024), „CO2 reduction of resolved wall structures: A load-bearing capacity-based modularization and assembly approach“, *Engineering Structures*, 300. 117197. https://doi.org/10.1016/j.engstruct.2023.117197.

Erklärung zum Einsatz von generativer künstlicher Intelligenz

Nach den „Living guidelines on the responsible use of generative ai in research“ (März 2024) der europäischen Kommission ist der Einsatz generativer künstlicher Intelligenz durch Wissenschaftler transparent darzustellen (European Commission 2024, S. 6). Im Folgenden werden deshalb Werkzeuge, die auf generativer künstlicher Intelligenz basieren und für die Erstellung der vorliegenden Arbeit verwendet wurden, vollständig aufgeführt. Dabei wird erklärt in welcher Weise und welchem Umfang sie eingesetzt wurden. Die mittels generativer künstlicher Intelligenz erstellten Inhalte wurden vom Autor der vorliegenden Arbeit stets überprüft und nur insoweit verwendet, als sie für korrekt befunden wurden. Die Verantwortung für die Inhalte der vorliegenden Arbeit liegt somit entsprechend European Commission (2024, S. 6) ausschließlich beim Autor. Die folgenden Werkzeuge wurden verwendet:

DeepL Translator
Link: https://www.deepl.com/de/translator (zuletzt überprüft am 07.06.2025)
Funktion: Übersetzung
Einsatz: Das Werkzeug wurde genutzt um die Kurzzusammenfassung der vorliegenden Arbeit ins Englische zu übersetzen. Die vorgeschlagene Übersetzung wurde an mehreren Stellen korrigiert.

DeepL Write und LanguageTool Premium
Link: https://www.deepl.com/de/write (zuletzt überprüft am 07.06.2025)
Funktion: Lektorat
Einsatz: Die Werkzeuge wurden genutzt um große Teile der vorliegenden Arbeit auf Rechtschreibung, Grammatik und Stil zu überprüfen und Korrekturvorschläge zu erhalten. Hinweise zu Rechtschreibung und Grammatik wurden in vielen Fällen übernommen. Hinweise zu stilistischen Verbesserungen des Textes wurden in wenigen Fällen übernommen.

Elicit, Scispace und scite Assistant
Links: https://elicit.com/ (zuletzt überprüft am 07.06.2025), https://scispace.com/ (zuletzt überprüft am 07.06.2025) bzw. https://scite.ai/assistant (zuletzt überprüft am 07.06.2025)
Funktion: Zusammenfassende Beschreibung des Stands der Forschung

Einsatz: Die Werkzeuge wurden genutzt um auf Basis einer textlichen Eingabe (z. B. einer Frage) eine Beschreibung des entsprechenden Stands der Forschung zu erstellen. Auf Basis dessen wurde potenziell relevante Literatur ausgewählt. Die Werkzeuge wurden insbesondere nicht genutzt um Inhalte für die vorliegende Arbeit zu erstellen. Ein kleiner Teil der in der vorliegenden Arbeit verwendeten Literatur wurde auf diese Weise gefunden.

GitHub Copilot und OpenAI ChatGPT
Links: https://github.com/features/copilot (zuletzt überprüft am 07.06.2025) bzw. https://chatgpt.com/ (zuletzt überprüft am 07.06.2025)
Funktion: Generierung, Korrektur und Dokumentation von Programmcode
Einsatz: Die Werkzeuge wurden genutzt um die im Rahmen der vorliegenden Arbeit entwickelten Algorithmen effizient in Programmcode zu implementieren. Für einen kleinen Teil des Programmcodes, insbesondere einfache Teilfunktionen, wurde mit diesen Werkzeugen ein erster Entwurf auf Basis einer textlichen Beschreibung erstellt. Dieser wurde i. d. R. anschließend überarbeitet. Vereinzelt wurden mit diesen Werkzeugen Ursachen für Fehler im Programmcode ermittelt. Für einen großen Teil des Programmcodes wurden diese Werkzeuge genutzt um einen Entwurf für Dokumentationen (Beschreibung von Funktionen, Parametern und Ausgaben) zu erstellen. Dieser wurde i. d. R. anschließend überarbeitet.

Anhang

A1 Anhang zu Kapitel 2.3

A1.1 Spaltengenerierung

Spaltengenerierung (CG) ist ein Prinzip, um Probleme der linearen Optimierung (LP) mit einer großen Anzahl von Variablen effizient zu lösen. Die konkrete Ausgestaltung entsprechender Ansätze ist vom Anwendungsfall abhängig. Im Folgenden wird auf einen Typ von CG näher eingegangen, der von Wilhelm (2001) als Typ II bezeichnet wird. Für die vorliegende Arbeit ist ausschließlich dieser Typ von CG relevant, weshalb in der vorliegenden Arbeit mit CG ausschließlich CG Typ II gemeint ist. Die Erklärung von CG im vorliegenden Kapitel basiert auf Wilhelm (2001).

Ein allgemeines LP-Problem lässt sich wie folgt darstellen:

$$\text{MP} \qquad \begin{aligned} min \quad & c_1 \ u_1 + \ \dots + c_N \ u_N \\ s.t.: \quad & a_{1,1}\, u_1 + \ \dots + a_{N,1}\, u_N \geq b_1 \\ & \vdots \qquad \ddots \qquad \vdots \qquad \vdots \\ & a_{M,1} u_1 + \ \dots + a_{M,N} u_N \geq b_M \\ & u_1, \dots, u_N \geq 0 \quad . \end{aligned} \qquad \text{A1.1}$$

Dabei bezeichnen $\boldsymbol{c} = (c_1, \dots, c_N)$ die Zielfunktionskoeffizienten, $\boldsymbol{u} = (u_1, \dots, u_N)$ die Variablen, $\boldsymbol{A} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{M,1} & \cdots & a_{M,N} \end{bmatrix}$ die Koeffizienten der Nebenbedingungen und $\boldsymbol{b} = (b_1, \dots, b_M)$ die rechten Seiten der Nebenbedingungen. Im Kontext der CG wird dieses Problem als Master-Problem (MP) bezeichnet. Die Koeffizienten der Nebenbedingungen bilden eine Matrix $\boldsymbol{A}$ mit M Zeilen und N Spalten, wobei jeder Zeile einer Nebenbedingung und jeder Spalte einer Variablen zugeordnet werden kann. Das Problem kann kompakt wie folgt dargestellt werden:

$$\text{MP} \qquad \begin{aligned} min \quad & \boldsymbol{c}\boldsymbol{u}^T \\ s.t.: \quad & \boldsymbol{A}\boldsymbol{u}^T \geq \boldsymbol{b}^T \\ & \boldsymbol{u}^T \geq \mathbb{0} \ . \end{aligned} \qquad \text{A1.2}$$

Die Berechnungsdauer für die Lösung des MP steigt mit der Anzahl von Variablen, weshalb LP-Probleme mit einer großen Anzahl von Variablen nicht in vertretbarer Zeit gelöst werden können. Es wird angenommen, dass eine zulässige, i. d. R. nicht optimale, Lösung des MP bekannt ist. Diese kann z. B. durch eine problemspezifische Heuristik generiert werden. Seien u_i^b die Basisvariablen dieser Lösung, d. h. es gilt $u_i^b \neq 0$ und für alle anderen Variablen $u_i^{nb} = 0$. Nun kann ein neues, reduziertes Master-Problem (RMP) aufgestellt werden, das nur die Basisvariablen enthält:

RMP
$$min \quad \boldsymbol{c}\,\boldsymbol{u}^{b^T} \qquad \text{A1.3}$$
$$s.t.: \quad \boldsymbol{A}^b\boldsymbol{u}^{b^T} \geq \boldsymbol{b}^T$$
$$\boldsymbol{u}^{b^T} \geq \mathbb{0} \ .$$

Im Folgenden wird zur einfachen Lesbarkeit auf das Superskript b verzichtet, wenn ersichtlich ist, dass ein RMP vorliegt. Da für jede Lösung des RMP implizit angenommen wird, dass alle initialen Nicht-Basisvariablen 0 sind, ist der Lösungsraum gegenüber dem MP eingeschränkt. Eine optimale Lösung des RMP entspricht damit nicht zwingend einer optimalen Lösung des MP. Die Idee hinter CG ist es, dem RMP schrittweise noch nicht berücksichtigte Variablen und damit der Matrix A Spalten hinzuzufügen. Mit jeder hinzugefügten Variablen wird der Lösungsraum erweitert und es können evtl. bessere Lösungen gefunden werden. Um das Verfahren effizient zu gestalten, werden nur solche Spalten hinzugefügt, die bessere Lösungen ermöglichen. Die Methode terminiert, sobald keine solche Spalte mehr existiert. Um Spalten zu ermitteln, die eine bessere Lösung des RMP ermöglichen, wird ein Subproblem (SP), oft als Pricing-Problem bezeichnet, aufgestellt. Hierfür wird zunächst das zum RMP gehörende Dualproblem (DP) aufgestellt. Es lautet:

DP
$$max \quad \boldsymbol{b}\boldsymbol{v}^T \qquad \text{A1.4}$$
$$s.t.: \quad \boldsymbol{A}^T\boldsymbol{v}^T \geq \boldsymbol{c}^T$$
$$\boldsymbol{v}^T \geq \mathbb{0} \ .$$

Dabei bezeichnen $\boldsymbol{v} = (v_1, \dots, v_M)$ die Variablen des DP. Entsprechend der Dualitätseigenschaft linearer Optimierungsprobleme ist jeder Variable im RMP eine Nebenbedingung im DP zugeordnet und jeder Nebenbedingung im RMP eine Variable im DP. Für das DP wird eine optimale Lösung $\boldsymbol{v}^*$ berechnet. Die Ausprägungen der Dualvariablen $\boldsymbol{v}^* = (v_1^*, \dots, v_M^*)$ in der optimalen Lösung stellen Opportunitätskosten der

Nebenbedingungen im primalen Problem dar. Für eine Nebenbedingung i im RMP gibt v_i^* an, um wie viel sich der optimale Zielfunktionswert des RMP verringern würde, wenn b_i um 1 verringert werden und damit der Lösungsraum von RMP vergrößert werden würde. Diese Verbesserung kann so weit ausgeschöpft werden, bis die Nebenbedingung i nicht mehr aktiv ist. Wird nun dem RMP eine neue Spalte $N+1$ hinzugefügt und hat diese für eine Nebenbedingung i einen Koeffizienten $a_{i,N+1}$, ergibt sich die neue Nebenbedingung i als

$$a_{i1}u_1 + \cdots + u_N a_{iN} + u_{N+1}a_{i,N+1} \geq b_i, \qquad \text{A1.5}$$

was

$$a_{i1}u_1 + \cdots + u_N a_{iN} \geq b_i - u_{N+1}a_{i,N+1} \qquad \text{A1.6}$$

entspricht. Die neue Spalte entspricht für $a_{i,N+1} \geq 0$ also einer Verringerung von b_i um $u_{N+1}a_{i,N+1}$ und ermöglicht damit eine Verbesserung des optimalen Zielfunktionswerts des RMP um $u_{N+1}a_{i,N+1}v_i^*$. Dieser Verbesserung des Zielfunktionswerts steht eine Verschlechterung des Zielfunktionswerts durch die Wahl von $u_{N+1} > 0$ aufgrund des Zielfunktionskoeffizienten c_{N+1} von u_{N+1} gegenüber. Um also dem RMP Spalten hinzuzufügen, die eine möglichst große Verbesserung des optimalen Zielfunktionswerts des RMP ermöglichen, ist ein Subproblem mit der folgenden Zielfunktion zu lösen.

$$\begin{aligned} \text{SP} \qquad max \qquad & \min c_{N+1}(a_{i,N+1}) - \sum_{i\in\{1,\dots,M\}} a_{i,N+1}v_i^* \\ s.t.: \qquad & a_{i,N+1} \geq 0 \qquad \forall i \in \{1,\dots,M\} \end{aligned} \qquad \text{A1.7}$$

Ist das MP vollständig explizit formuliert, sind die Kosten c_{N+1} für jede mögliche Spalte bekannt. Außerdem ist bekannt, welche Spalten, d. h. welche $\boldsymbol{a}_{N+1} = (a_{1,N+1}, \dots, a_{M,N+1})$ gewählt werden können. Entsprechend sind die Nebenbedingungen des SP zu formulieren. In vielen Fällen, in denen CG zum Einsatz kommt, ist jedoch die Anzahl von Variablen und damit Spalten des MP so groß, dass es zu aufwendig wäre, sie explizit zu formulieren. Wenn diese Spalten und deren Kosten jedoch gewissen Regeln folgen, können die Nebenbedingungen des SP und die Kosten in Abhängigkeit der gewählten Spalte entsprechend formuliert werden. Dadurch müssen nicht alle möglichen Spalten vorab bekannt sein – eine Stärke der CG.

Zuletzt sei darauf hingewiesen, dass die hier vorgestellte Form der CG nur für stetige LP und nicht für ganzzahlige lineare Optimierung (ILP) gilt, weil für ILP-Probleme zum RMP kein DP aufgestellt werden kann. Um CG auf ILP-Probleme anwenden zu können,

müssen das MP und das RMP auf stetige LP-Probleme relaxiert werden. Über das relaxierte reduzierte Master-Problem (XRMP) kann eine optimale Lösung des relaxierten Master-Problems (XMP) berechnet werden. Diese ist jedoch nicht zwingend ganzzahlig. Ggf. müssen zusätzlich Metaverfahren zur Lösung von ILP-Problemen – wie z. B. Branch and Bound – angewandt werden.

A2 Anhang zu Kapitel 3.2.2

A2.1 Phylogenetik

Die Phylogenetik ist ein Fachgebiet, das sich mit den evolutionären Beziehungen zwischen Lebewesen befasst (Abaza 2020, S. 68). Abstammungsbeziehungen zwischen Lebewesen werden in der Phylogenetik u. a. als Bäume dargestellt. Es existieren zahlreiche Ansätze, um die Abstammungsbeziehungen aus mehreren phylogenetischen Bäumen zusammenzuführen. Diese Problemstellung ist mit der Erstellung einer Maximalstückliste (MSTL) auf Basis variantenbezogener Stücklisten (VSTLs) vergleichbar. Deshalb sind zugehörige Ansätze der Phylogenetik auch für die vorliegende Arbeit relevant. Im Folgenden wird auf die relevanten Begriffe der Phylogenetik und die Übertragbarkeit bestehender Ansätze eingegangen.

Die Blätter phylogenetischer Bäume stellen Taxa, d. h. Gruppen genetisch ähnlicher Organismen, dar, die jeweils durch ein Label[76] annotiert sind (Abaza 2020, S. 68). Innere Knoten stellen Verzweigungspunkte in der Evolutionären Entwicklung dar (Abaza 2020, S. 68). Wenn die Bäume als VSTLs interpretiert werden, stellen die Taxa Zukaufkomponenten (ZKs) dar, welche z. B. mit ihrer Bezeichnung als Label versehen sind. Die inneren Knoten entsprechen Baugruppen. Abstammungsbeziehungen zwischen Lebewesen können sog. Retikulationen aufweisen, d. h. Fälle in den ein Organismus von mehr als einem anderen Organismus abstammt (Bastide et al. 2018, S. 800). Damit können verschiedene phylogenetische Bäume verschiedene Abstammungen für denselben Organismus enthalten. Werden somit Abstammungsbäume zusammengeführt resultiert deshalb u.U. kein Baum, sondern ein allgemeiner Graph. Dieser wird auch als Netzwerk bezeichnet (Huson & Linz 2018, S. 398). Moussa & ElMaraghy (2018), die einen Ansatz der Phylogenetik auf die industrielle Montage übertragen, interpretieren

[76] Der Begriff Label wird hier im Sinne der Phylogenetik verwendet und ist nicht zu verwechseln mit dem Begriff Label im Kontext des maschinellen Lernens (siehe Kapitel 2.3.1).

Retikulationen als alternative Fügereihenfolgen. Sie nutzen deshalb ebenfalls Netzwerke um Fügereihenfolgen darzustellen.

Abbildung A2.1 zeigt das von Moussa & ElMaraghy (2018) verwendete Beispiel für die Synthese eines Master Assembly Networks aus sieben Assembly Sequence Trees (T1 bis T7). In ihrer Funktion sind Master Assembly Networks mit MSTLs und Assembly Sequence Trees mit VSTLs vergleichbar. Aus dem Master Assembly Network können Assembly Sequence Trees konfiguriert werden indem ausgewählte Retikulationskanten (in der Abbildung farbig hervorgehoben) sowie ausgewählte Blätter entfernt werden. Es ist ersichtlich, dass z. B. der Baum T6 nicht aus dem Netzwerk konfiguriert werden kann, obwohl er verwendet wurde um das Netzwerk zu erstellen. Dabei handelt es sich um ein grundsätzliches Problem bei der Übertragung von Ansätzen der Phylogenetik auf die industrielle Montage. In der Phylogenetik existiert die Anforderung, dass ein

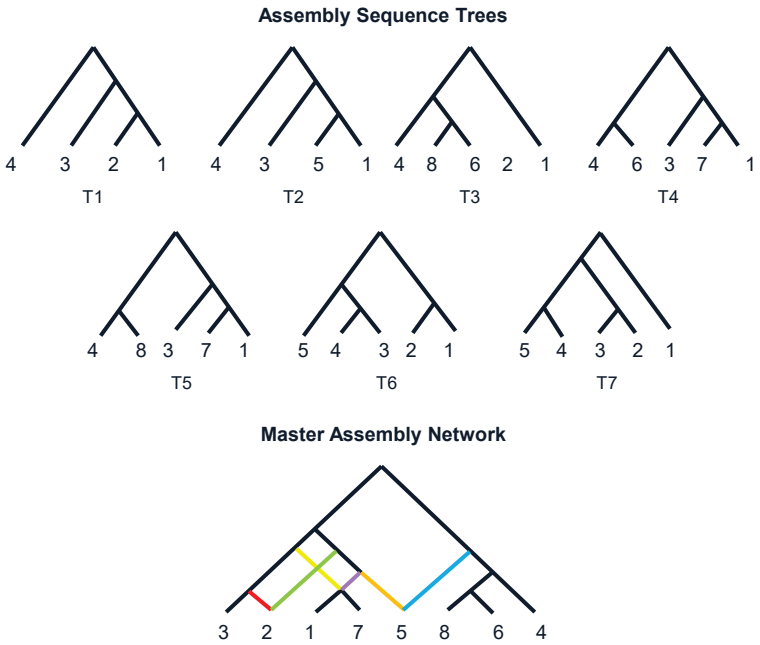
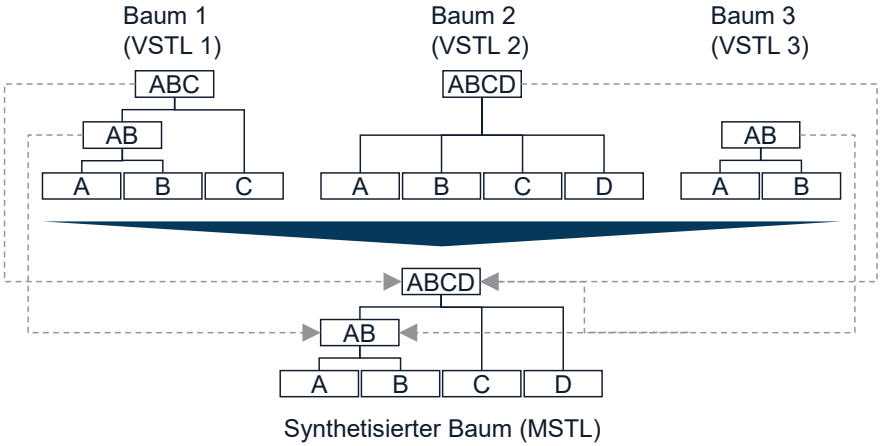


Abbildung A2.1: Beispiel von Moussa & ElMaraghy (2018) für die Synthese eines Master Assembly Networks aus Assembly Sequence Trees (eigene Darstellung auf Basis von Moussa & ElMaraghy (2018, S. 795))

synthetisierter Baum oder ein synthetisiertes Netzwerk in der Lage sein muss die zu seiner Synthese verwendeten Bäume darzustellen (engl. display). Im Folgenden wird gezeigt, dass dies nicht identisch mit der Anforderung, dass aus einer MSTL, alle zu ihrer Erstellung verwendeten VSTLs konfiguriert werden können, ist. Damit garantieren Verfahren der Phylogenetik nicht die in Kapitel 3.2.1 geforderte Informationserhaltung.

Sei ein Cluster $L(N)$ eines Knotens N definiert als Menge aller Labels, die dem Knoten direkt oder indirekt untergeordnet sind[77]. Beispielsweise entspricht die Wurzel von Baum 1 in Abbildung A2.2 dem Cluster $\{A, B, C\}$.



MSTL = Maximalstückliste, VSTL = Variantenbezogene Stückliste

Abbildung A2.2: Darstellbarkeit im Sinne der Phylogenetik

Sei $L(T)$ für einen Baum T definiert als das Cluster seiner Wurzel, d. h. als alle Labels, die in ihm auftreten. Sei $Cl(T)$ die Menge aller Cluster in einem Baum T. Es gilt damit z. B. $Cl(Baum\ 1) = \{\{A, B, C\}, \{A, B\}, \{A\}, \{B\}, \{C\}\}$. Sei $Cl(T|L)$ die Menge aller nichtleeren Cluster in $Cl(T)$ jeweils projiziert auf eine Menge L an Labels. Im Beispielfall gilt damit

$$Cl(Synt.Baum|L(Baum\ 1)) = Cl(Synt.Baum|\{A, B, C\})) \\ = \{\{A, B, C\}, \{A, B\}, \{A\}, \{B\}, \{C\}\} \quad \text{A2.1}$$

[77] Die hier und im Folgenden verwendete Notation entspricht der von Deng & Fernández-Baca (2018).

Damit gilt also im Beispielfall $Cl(Baum\ 1) = Cl(Synt.Baum|L(Baum\ 1))$. Generell gilt, dass ein Baum T einen Baum T' darstellt, falls

$$Cl(T') \subseteq Cl(T|L(T')) \quad \text{A2.2}$$

gilt (Deng & Fernández-Baca 2018, S. 2455). Gemäß dieser Definition stellt also der synthetisierte Baum den Baum 1 dar. Für Baum 2 gilt

$$Cl(Baum\ 2) = \{\{A,B,C,D\},\{A\},\{B\},\{C\},\{D\}\} \quad \text{A2.3}$$
$$\subset \{\{A,B,C,D\},\{A,B\},\{A\},\{B\},\{C\},\{D\}\} = Cl(Synt.Baum|L(Baum\ 2)$$

und für Baum 3

$$Cl(Baum\ 3) = \{\{A,B\},\{A\},\{B\}\} \subset \{\{A,B\},\{A\},\{B\}\} \quad \text{A2.4}$$
$$= Cl(Synt.Baum|L(Baum\ 3)).$$

D. h. der synthetisierte Baum stellt auch die Bäume 2 und 3 dar. Damit existiert ein Baum, der die Bäume 1, 2 und 3 darstellen kann. Damit gelten die Bäume 1, 2 und 3 im Sinne der Phylogenetik als kompatibel. Werden die Bäume jedoch als VSTLs bzw. MSTLs aufgefasst, ist es nicht möglich, VSTL 2 aus der MSTL zu konfigurieren. Wird die Komponente AB der MSTL nicht instanziiert, können auch A und B nicht instanziiert werden. Wird Komponente AB hingegen instanziiert, ergibt sich eine VSTL, die eine Baugruppe AB enthält und somit nicht VSTL 2 entspricht. Die Darstellbarkeit im Sinne der Phylogenetik ist damit zwar eine notwendige, aber keine hinreichende Bedingung dafür, dass eine VSTL aus einer MSTL konfiguriert werden kann. Damit eine VSTL aus einer MSTL konfiguriert werden kann, muss stattdessen

$$Cl(T') = Cl(T|L(T')) \quad \text{A2.5}$$

gelten. Deshalb sind Ansätze der Phylogenetik zur Synthese von Netzwerken aus Bäumen nicht unmittelbar geeignet um MSTLs aus VSTLs datenbasiert zu erstellen.

Es sei angemerkt, dass die Definition der Konfigurierbarkeit nicht impliziert, dass die Anzahl der Komponenten in der MSTL identisch zur Anzahl der Komponenten in der VSTL sein muss. Zum einen gibt es Komponenten in der MSTL, deren Cluster durch die Projektion auf $L(T')$ zu leeren Mengen und damit nicht berücksichtigt werden, wie z. B. die Komponente C für $Cl(MSTL|L(VSTL\ 3))$. Zum anderen gibt es Komponenten in der MSTL, deren Cluster auf dieselben Mengen projiziert und im Sinne der klassischen Mengenlehre nicht mehrfach betrachtet werden, wie z. B. die Komponenten ABCD und AB für $Cl(MSTL|L(VSTL\ 3))$. Da es in Baum 3 weder eine ZK C noch eine ZK D gibt, werden sowohl das Cluster von ABCD als auch das von AB auf das Cluster $\{A,B\}$ projiziert. Zwei Komponenten der MSTL steht somit eine Komponente der VSTL 3 gegenüber. Wenn VSTL 3 aus der MSTL konfiguriert wird, ist nicht eindeutig, aus

welcher Klasse der MSTL die Komponente AB in VSTL 3 instanziiert wird. Generell können Fälle auftreten, in denen einer Baugruppe einer VSTL mehrere Komponentenklassen (KKs) der MSTL zugeordnet werden können. Dieser Sachverhalt ist für den im Rahmen der vorliegenden Arbeit entwickelten Algorithmus Alg^{MSTL} (siehe Kapitel A3.1) relevant.

Zuletzt sei angemerkt, dass VSTL 1 und VSTL 2 unterschiedliche Angaben über die Fügereihenfolge der ZKs A, B und C machen. Nach VSTL 1 werden zunächst A und B gefügt und anschließend die resultierende Baugruppe AB mit C. In VSTL 2 werden A, B und C zugleich gefügt. Es liegen somit Strukturalternativen (STAs) in den VSTLs vor. Deshalb kann es keine MSTL mit Baumstruktur geben, die jeweils einmal die ZK A, B und C enthält und die Konfiguration von VSTL 1 und VSTL 2 zulässt. Kapitel 4.1.1 geht auf die Berücksichtigung von STAs in MSTLs ein.

A3 Anhang zu Kapitel 4.2

A3.1 Beschreibung und Pseudocode zu Algorithmus Alg^{MSTL}

Alg^{MSTL} basiert auf dem Algorithmus BuildST von Deng & Fernández-Baca (2018). BuildST berücksichtigt die Darstellbarkeit der eingehenden Bäume bei der Synthese eines Netzwerks im Sinne der Phylogenetik. Demgegenüber berücksichtigt Alg^{MSTL} die Konfigurierbarkeit von variantenbezogenen Stücklisten (VSTLs) aus einer Maximalstückliste (MSTL) (siehe Anhang A2). Der Algorithmus wird anhand des Beispiels in Abbildung A3.1 (1) erläutert und findet sich am Ende des Kapitels in Pseudocode.

Zu Beginn liegt eine Menge S^{VSTL} an VSTLs ohne Multikomponenten vor. Der Produktklasse der zu erstellenden MSTL muss direkt oder indirekt je eine Zukaufkomponentenklasse (ZKK) je Zukaufkomponente (ZK) in allen VSTLs aus S^{VSTL} untergeordnet sein. Damit entspricht ihr Cluster immer der Vereinigung aller Cluster der Wurzeln aller VSTLs aus S^{VSTL}. Aus einer solchen Produktklasse lassen sich immer alle Wurzeln der VSTLs instanziieren. Es wird deshalb zu Beginn eine entsprechende Produktklasse in die MSTL eingefügt und die Wurzeln der VSTLs dieser Produktklasse zugeordnet und markiert (1). Die Cluster der KKs der nächsten Ebene müssen eine Partition des Clusters der Produktklasse darstellen, d. h. im Beispielfall paarweise disjunkte Teilmengen der Menge $\{A, B, C, D, E\}$. Die Erstellung der MSTL folgt dem Prinzip, die Cluster der Komponentenklassen (KKs) sukzessive zu partitionieren, um so die MSTL von oben

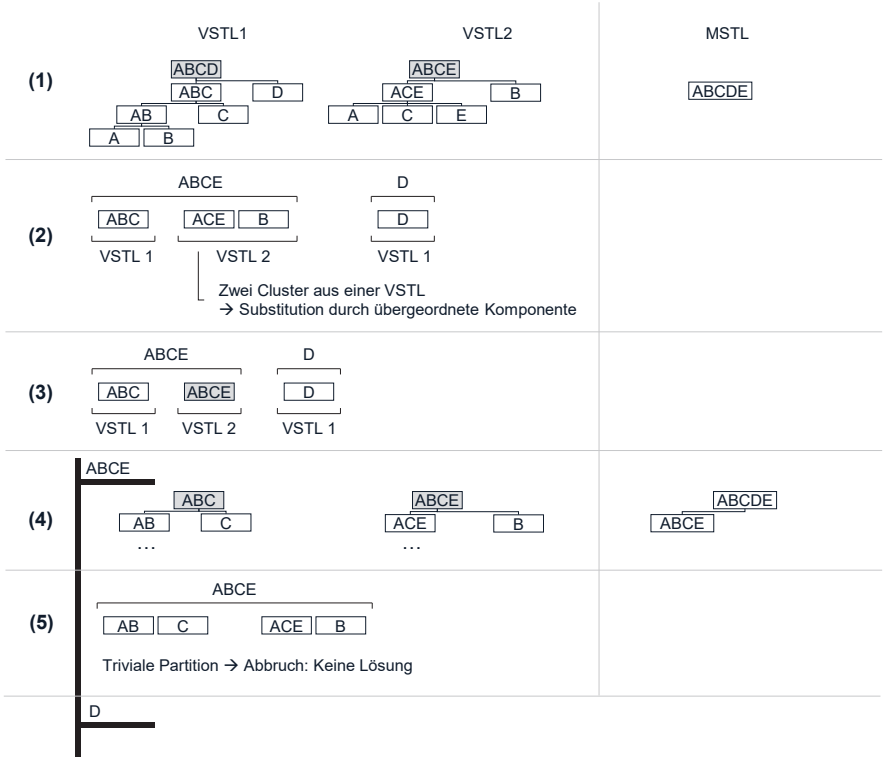


Abbildung A3.1: Beispielhafte Ausführung von Alg^{MSTL}

nach unten aufzubauen. Partitionen können Cluster von Komponenten zerteilen. In diesem Fall kann diese Komponente weder durch eine aus der Partitionierung entstehende KK noch durch eine KK einer untergeordneten Ebene instanziiert werden. Das Cluster einer Komponente darf durch eine Partitionierung nur dann zerteilt werden, wenn diese Komponente bereits einer KK der MSTL zugeordnet ist und somit instanziiert werden kann. Es wäre also im Beispielfall nicht möglich, das Cluster $\{A, B, C, D, E\}$ der Klasse ABCDE als $\{\{A, B\}, \{C, D, E\}\}$ zu partitionieren, da in diesem Fall u. a. die Komponente ABC, welche noch keiner KK zugeordnet ist, nicht mehr instanziiert werden könnte.

Um eine zulässige Partition unterhalb der zuletzt eingefügten Klasse zu bestimmen, werden die Cluster der Komponenten der nächsten Ebene betrachtet. Es werden Inseln

von überlappenden Clustern gebildet (Abbildung A3.1, 2). Jede Insel entspricht einer neu hinzuzufügenden KK der MSTL. Die Inseln bilden eine zulässige Partition, da bei der Inselbildung keine Cluster von nicht zugeordneten Komponenten geteilt werden können. Größere Partitionen, d. h. solche mit mehr Inseln, sind nicht möglich. Eine kleinere Partition, wie sie z. B. entstehen würde, wenn alle betrachteten Komponenten in einer Insel zusammengefasst würden, erscheint zunächst möglich. Allerdings würde eine solche Partition zu KKs in der MSTL führen, die keine Entsprechungen in den VSTLs aufweisen. Dies würde der Definition der Konfigurierbarkeit in Anhang A2 widersprechen. Damit ist die so gebildete Partition – im Beispielfall $\{\{A, B, C, E\}, \{D\}\}$ – zwingend. Es sei angemerkt, dass das Cluster der Komponente ABCD durch diese Partition geteilt wird. Damit kann die Komponente aus keiner der resultierenden KKs und keiner untergeordneten KK instanziiert werden. Dies ist jedoch kein Hindernis, da ABCD bereits der Produktklasse ABCDE zugeordnet ist und somit aus dieser instanziiert werden kann.

Bevor je Insel eine KK in der MSTL erstellt wird, wird zunächst überprüft, welche Komponenten der Inseln den entsprechenden KKs zugeordnet, d. h. aus diesen instanziiert, werden können. Komponenten, die alleinige Repräsentanten ihrer VSTL in einer Insel sind, wie ABC und D im Beispielfall, können unmittelbar den entsprechenden KKs zugeordnet werden. Liegen in einer Insel jedoch mehrere Komponenten derselben VSTL vor, können diese nicht der entsprechenden KK zugeordnet werden, da nicht mehrere Komponenten einer VSTL aus derselben KK instanziiert werden können[78]. Im Beispielfall liegen mit den Clustern ACE und B zwei Cluster aus VSTL 2 in der Insel ABCE vor. Damit können weder ACE noch B einer KK ABCE in der MSTL zugeordnet werden. Eine KK ABCE hätte somit zunächst keine ihr zugeordnete Komponente in VSTL 2. Ist lediglich eine Darstellbarkeit der VSTL durch die MSTL gefordert, ist dies nicht relevant. Ist jedoch eine Konfigurierbarkeit aller VSTLs aus der MSTL gefordert, darf nach Definition der Konfigurierbarkeit in Anhang A2 keine KK in der MSTL existieren, die für eine VSTL aus S^{VSTL} keine Instanziierung zulässt. Wie in Anhang A2 erläutert, ist es jedoch möglich, dass bestimmte Komponenten aus mehreren KKs der MSTL instanziiert werden können. Es ist also u. U. möglich, der neu zu erstellenden KK die übergeordnete Komponente in der entsprechenden VSTL zuzuordnen. Deshalb wird statt der

[78] Lediglich eine dieser Komponenten zuzuordnen ist ebenfalls nicht möglich, da dadurch ihre Klasse in der MSTL der Klasse der anderen Komponenten übergeordnet wäre, wodurch sich die Struktur der VSTL nicht mehr aus der MSTL ergeben würde.

Komponenten der betroffenen VSTLs ihre übergeordnete Komponente betrachtet und die Inselbildung erneut durchgeführt. Dieser Vorgang wird so lange wiederholt, bis in jeder Insel jeweils höchstens eine Komponente je VSTL vorhanden ist (Abbildung A3.1, 3). Nachdem im Beispielfall die Komponenten ACE und B durch ihre übergeordnete Komponente ABCE ersetzt wurden, ist eine vollständige Zuordnung von Komponenten zu den zu erstellenden KKs möglich. Die Komponenten ABC und ABCE können einer KK ABCE und die Komponente D einer KK D in der MSTL zugeordnet werden. Im Ersetzen der Komponenten und dem erneuten Berechnen der Inseln besteht der wesentliche Unterschied zum Algorithmus BuildST von Deng & Fernández-Baca (2018).

Nach dem erfolgreichen Bilden der Inseln verzweigt sich der Algorithmus und wird jeweils für jede Insel mit den Teilbäumen je Komponente der Insel rekursiv fortgesetzt (Abbildung A3.1, 4). Zu Beginn einer Rekursion wird, wie oben beschrieben, eine KK in der MSTL erstellt und die Wurzeln der Teilbäume dieser KK zugeordnet. Aufgrund der oben vorgenommenen Betrachtung sind diese Zuordnungen immer zulässig. Ein Ast des Algorithmus wird nicht weiter betrachtet, sobald die Teilbäume des Astes insgesamt nur noch ein oder zwei verschiedene Labels aufweisen. In diesem Fall sind die resultierenden KKs der MSTL trivial gegeben. Sobald alle Äste erfolgreich berechnet wurden, wird die erstellte MSTL zurückgegeben.

Es kann jedoch der Fall auftreten, dass sich bei der Inselbildung nur eine Insel, d. h. eine triviale Partition, ergibt (Abbildung A3.1, 5). Durch das Ersetzen von Komponenten durch ihre übergeordneten Komponenten und erneute Inselbildung kann die Anzahl der Inseln in keinem Fall vergrößert werden. Damit lässt sich auf diese Weise keine nichttriviale Partition bilden. Die Umsetzung der trivialen Partition würde dazu führen, dass eine KK in der MSTL entsteht, die keine Entsprechung in den VSTLs besitzt. Die Umsetzung einer anderen Partition würde dazu führen, dass mindestens eine der noch nicht zugeordneten Komponenten nicht aus einer KK der MSTL instanziiert werden könnte. Im Beispielfall würde z. B. die Partition $\{\{A, B\}, \{C, E\}\}$ dazu führen, dass die Komponente ACE nicht instanziiert werden könnte. Die zuletzt hinzugefügte KK kann somit nicht weiter partitioniert werden. Da wie oben beschrieben alle Partitionierungen zwingend sind, lässt sich dieser Zustand auch nicht durch die Vornahme anderer Partitionierungen in übergeordneten Ebenen vermeiden. Damit lässt sich insgesamt durch Partitionierung keine MSTL erstellen, aus der alle VSTL aus S^{VSTL} konfiguriert werden können. Der Algorithmus terminiert mit einer entsprechenden Ausgabe. Für den Beispielfall liegt dies offensichtlich daran, dass VSTL 1 und VSTL 2 unterschiedliche

Angaben über die Fügereihenfolge der ZKs A, B und C machen, d. h. es liegen Strukturalternativen (STAs) vor.

Im Folgenden ist der Pseudocode von Alg^{MSTL} dargestellt. Aus jedem Aufruf des rekursiven Algorithmus ergibt sich eine weitere KK $\mathrm{KK}^{\mathrm{neu}}$ der MSTL. Der rekursive Algorithmus wird mit der Menge $S^{markiert}$ der Wurzeln der betrachteten VSTLs oder den Wurzeln der betrachteten Teilbäume aufgerufen. Für alle Aufrufe außer dem ersten wird darüber hinaus diejenige KK $KK^{über}$ übergeben, die der zu erstellenden KK übergeordnet ist. In Abbildung A3.1 (4) würde also die KK ABCDE übergeben werden. Es wird davon ausgegangen, dass alle Komponenten über Referenzen auf ihre Vorgänger und Nachfolger in ihrer VSTL verfügen. Gibt der Algorithmus None zurück, existiert keine Lösung.

Alg^{MSTL}: Algorithmus zur Erzeugung einer Maximalstückliste aus einer Menge von variantenbezogenen Stücklisten ohne Multikomponenten und Strukturalternativen

Input: $S^{markiert}$, $KK^{über}$

Output: KK^{neu} oder None

```
# KK erstellen
S^ClNeu := ∪_{k∈S^markiert} get_cluster(k)
KK^neu := new KK(S^ClNeu)
verbinde_mit_gerichteter_Kante(KK^über, KK^neu)

# Positive Abbruchkriterien prüfen
if |S^ClNeu| == 1 then
        return KK^neu
if |S^ClNeu| == 2 then
        KK^neu,sub1 := new KK(pop(S^ClNeu))
        verbinde_mit_gerichteter_Kante(KK^neu , KK^neu,sub1)
        KK^neu,sub2 := new KK(pop(S^ClNeu))
        verbinde_mit_gerichteter_Kante(KK^neu , KK^neu,sub2)
        return KK^neu

# Partition bestimmen
S^betrachtet := ∪_{k∈S^markiert} get_nachfolger(k)
while True do
        S^Inseln := bestimme_überlappende_komponenten(S^betrachtet)
        if |S^Inseln| == 1 then
                return None
        S^VSTLZuErsetzen := new Set()
        for S^Insel in S^Inseln do
                S^VSTLZuErsetzen := S^VSTLZuErsetzen ∪ mehrfache_VSTL(S^Insel)
        if |S^VSTLZuErsetzen| == 0 then
                break
        S^KompZuErsetzen := new Set()
        for k in S^markiert do
                if get_VSTL(k) in S^VSTLZuErsetzen then
```

```
                    S^KompZuErsetzen := S^KompZuErsetzen ∪ {k}
        for k in S^KompZuErsetzen do
            S^betrachtet := S^betrachtet \ {k}
            S^betrachtet := S^betrachtet ∪ {get_vorgaenger(k)}

# Nächste Ebene aufrufen
for S^Insel in S^Inseln do
    if Alg^MSTL(S^Insel, KK^neu) == None then
        return None

# KK zurückgeben
return KK^neu
```

Der Pseudocode geht von den folgenden Funktionen sowie von Standardfunktionen objektorientierter Programmiersprachen aus:

- verbinde_mit_gerichteter_Kante: Verbindet zwei Knoten in einem Graph mit einer gerichteten Kante
- bestimme_überlappende_komponenten: Gibt für eine Menge an Komponenten eine Menge von Listen von Komponenten zurück. Jede Liste entspricht einer Insel von Komponenten, deren Cluster sich überlappen.[79]
- mehrfache_VSTL: Gibt für eine Menge von Komponenten diejenigen VSTLs zurück, auf die mehr als eine der Komponenten referenziert

Der Pseudocode geht darüber hinaus von der Klasse KK aus, die KKs der MSTL darstellt (siehe Kapitel 4.1.1).

A3.2 Beschreibung und Pseudocode zu Algorithmus $Alg^{GemPfad}$

Zwei Pfade sind als Liste von Knoten gegeben, wobei jeder Knoten i einer Menge S_i^{Kn} von Labels entspricht. Im Folgenden werden die Pfade $L_I^{Pf} = (\{B2\}, \{B3, B4\})$ und $L_V^{Pf} = (\{B3\}, \{B2, B3, B4\})$ aus Abbildung 4.10 in Kapitel 4.2.1.1.2 als Beispiel verwendet. Seien im Allgemeinen n_1^{PfL}und n_2^{PfL} die Längen der beiden Pfade. Wie in Kapitel 4.2.1.1.2 erwähnt, ähnelt das Problem einem Longest-Common-Subsequence-Problem. Analog zu diesem müssen Knoten der beiden Pfade einander zugeordnet werden, um den gemeinsamen Pfad mit der größten Übereinstimmung zu erhalten. Aus dem Beispiel in Kapitel 4.2.1 ist bekannt, dass der gemeinsame Pfad mit der größten

[79] Diese Funktion kann z. B. effizient über Union-Find-Strukturen (siehe hierfür Knebl 2021, S. 256–264) realisiert werden.

Übereinstimmung ($\{B3, B4\}$) ist. Um diesen zu erhalten, müssen die Knoten $\{B2, B3, B4\}$ und $\{B3, B4\}$ einander zugeordnet werden. Der gemeinsame Pfad ergibt sich, indem alle zugeordneten Knoten paarweise aufeinander projiziert und die resultierenden Knoten in die Reihenfolge gebracht werden, in der sie in den Pfaden auftreten. Eine Zuordnung von Knoten über Kreuz ist nicht zulässig, d. h. die Reihenfolge der resultierenden Knoten in den beiden Pfaden muss identisch sein, da ansonsten der gemeinsame Pfad nicht in beiden Pfaden enthalten sein kann. Z. B. wäre eine Zuordnung des ersten Knotens aus L_I^{Pf} zum zweiten Knoten aus L_V^{Pf} und des zweiten Knotens aus L_I^{Pf} zum ersten Knoten aus L_V^{Pf} nicht zulässig. Das Problem lässt sich ebenso wie das Longest-Common-Subsequence-Problem in Teilprobleme zerlegen. Sei $P_{i,j}^{GePf}$ dasjenige Teilproblem des betrachteten Problems P^{GePf}, bei dem nur die ersten i Knoten des ersten Pfads und die ersten j Knoten des zweiten Pfads betrachtet werden. Damit entspricht $P_{n_1^{pfl}, n_2^{pfl}}^{GePf}$ gerade P^{GePf}. Sei s^{GePf} die Länge des längsten gemeinsamen Pfads.

Es werden nacheinander je ein Knoten S_i^{Kn} aus dem ersten Pfad, L_1^{Pf}, und ein Knoten S_j^{Kn} aus dem zweiten Pfad, L_2^{Pf}, betrachtet. Sei $n_{i,j}^{IdLa}$ die Anzahl übereinstimmender Labels in zwei Knoten i und j. Die Betrachtung beginnt mit dem jeweils letzten Knoten $S_{n_1^{PfL}}^{Kn}$ in Pfad 1 und $S_{n_2^{PfL}}^{Kn}$ in Pfad 2. Im Beispielfall wären dies $\{B3, B4\}$ und $\{B2, B3, B4\}$. Für die Zuordnung der Knoten zu Knoten des jeweils anderen Pfads existieren folgende Möglichkeiten:

1. Knoten S_i^{Kn} wird Knoten S_j^{Kn} zugeordnet. In diesem Fall gilt
$$s^{GePf}\left(P_{i,j}^{GePf}\right) = s^{GePf}\left(P_{i-1,j-1}^{GePf}\right) + n_{i,j}^{IdLa} \quad \text{A3.1}$$
weil die Gesamtübereinstimmung gegenüber dem Problem ohne S_i^{Kn} und S_j^{Kn} um die Übereinstimmung von S_i^{Kn} und S_j^{Kn} erhöht wird.
2. S_i^{Kn} wird einem Knoten in Pfad 2 vor S_j^{Kn} zugeordnet. In diesem Fall gilt, dass Knoten S_j^{Kn} nicht zugeordnet werden kann, weil ansonsten eine Zuordnung über Kreuz vorliegen würde. Da also S_j^{Kn} nicht zugeordnet wird, gilt
$$s^{GePf}\left(P_{i,j}^{GePf}\right) = s^{GePf}\left(P_{i,j-1}^{GePf}\right). \quad \text{A3.2}$$
3. S_j^{Kn} wird einem Knoten in Pfad 1 vor S_i^{Kn} zugeordnet. In diesem Fall gilt analog

$$s^{GePf}\left(P_{i,j}^{GePf}\right) = s^{GePf}\left(P_{i-1,j}^{GePf}\right). \quad \text{A3.3}$$

4. Weder Kn_i noch Kn_j werden überhaupt zugeordnet. In diesem Fall gilt

$$s^{GePf}\left(P_{i,j}^{GePf}\right) = s^{GePf}\left(P_{i,j-1}^{GePf}\right) = s^{GePf}\left(P_{i-1,j}^{GePf}\right). \quad \text{A3.4}$$

Da sich die optimale Lösung des Problems nach dem Bellmannschen Optimalitätsprinzip aus optimalen Teillösungen zusammensetzt, wird jeweils diejenige Möglichkeit gewählt, die den Wert des Teilproblems $s^{GePf}\left(P_{i,j}^{GePf}\right)$ maximiert. Damit gilt:

$$s^{GePf}\left(P_{i,j}^{GePf}\right) = \max(s^{GePf}\left(P_{i-1,j-1}^{GePf}\right) + n_{i,j}^{IdLa}, s^{GePf}\left(P_{i,j-1}^{GePf}\right), s^{GePf}\left(P_{i-1,j}^{GePf}\right)). \quad \text{A3.5}$$

Seien $P_{0,j}^{GePf}$ und $P_{i,0}^{GePf}$ Teilprobleme bei denen einer der beiden Pfade leer ist, dann gilt

$$s^{GePf}\left(P_{0,j}^{GePf}\right) = P_{i,0}^{GePf} = 0. \quad \text{A3.6}$$

Davon ausgehend lässt sich die Lösung von $P_{n_1^{pfl},n_2^{pfl}}^{GePf}$ durch sukzessives Lösen der untergeordneten Probleme lösen. Die Einträge $A_{i,j}^{GePf}$ der null-basiert indizierten Matrix A^{GePf} enthalten jeweils die optimalen Zielfunktionswerte der Probleme $P_{i,j}^{GePf}$. Im Folgenden ist der Algorithmus als Pseudocode dargestellt.

Alg^{GemPfad}: Algorithmus zur Berechnung der Übereinstimmung zweier Pfade

Input: L_1^{Pf}, L_2^{Pf}

Output: l^{GePf}

```
1: n_1^PfL := |L_1^Pf|
2: n_2^PfL := |L_2^Pf|
3: A^GePf := 0^(n_1^PfL) × 0^(n_2^PfL)
4: for i=1 to n_1^PfL do
5:     for j=1 to n_2^PfL do
6:         A_{i,j}^GePf = max(s^GePf(P_{i-1,j-1}^GePf) + n_{i,j}^IdLa, s^GePf(P_{i,j-1}^GePf), s^GePf(P_{i-1,j}^GePf))
7: return A^GePf_{n_1^pfl, n_2^pfl}
```

A3.3 Pseudocode zu Algorithmus Alg^{MinMSTL}

Der Algorithmus geht von einer Menge S^{VSTL} von variantenbezogenen Stücklisten (VSTLs) aus, die Multikomponenten und Strukturalternativen (STAs) enthalten können. Er gibt ein Dictionary zurück, das einen Eintrag je Komponentenklasse (KK) der Maximalstückliste (MSTL) enthält – identifiziert durch eine Bezeichnung und eine Klassennummer (KN) – und diesem die zugehörigen Zukaufkomponenten (ZKs) der VSTLs aus S^{VSTL} zuordnet. Umgekehrt ordnet das Dictionary damit jeder ZK ihre Klasse zu.

$Alg^{MinMSTL}$: Algorithmus zur Erzeugung einer Maximalstückliste aus einer Menge von variantenbezogenen Stücklisten mit mehrfach auftretenden Zukaufkomponenten und Strukturalternativen

Input: S^{VSTL}

Output: $D^{BesteZuordnung}$

```
# 1. Initialisierung
# Initialisiere Variablen
L^alleZK := bestimme_alle_ZK(S^VSTL)
L^alleBez := bestimme_alle_bezeichnungen(L^alleZK)
D^ZKJeBez := bestimme_alle_ZK_je_bezeichnung(L^alleBez, L^alleZK)
D^maxAnzZKjeBez := bestimme_max_anz_zk_je_bez(L^alleBez, S^VSTL)
D^AnzLaJeBez := new Dict()
for str^bez in L^alleBez do
        D^AnzLaJeBez[str^bez] := 0
D^KompJeLabel := new Dict()
L^Pfad := new Zustand(None, None, None, new Set())
b^Un := ∞

# Bestimme Betrachtungsreihenfolge
for str^bez in L^alleBez do
        L^ZKMitBez := D^ZKJeBez[str^bez]
        A^Dist := 0^|L^ZKMitBez| × 0^|L^ZKMitBez|
        for i = 0 to |L^ZKMitBez| do
                Komp^1 := L^ZKMitBez[i]
                for j = i + 1 to |L^ZKMitBez| do
                        Komp^2 := L^ZKMitBez[i]
                        s^ZK_Komp^1,Komp^2 := bestimme_relative_aehnlichkeit(Komp^1, Komp^2)
                        A^Dist_i,j := 1 − s^ZK_Komp^1,Komp^2
        n^PrCl, d^MICD := kneedle(A^Dist)
        for k in D^ZKJeBez[str^bez] do
                set_erw_anz_clusters(k, n^ProgCl)
                set_mittl_intra_cluster_distanz(k, d^MICD)
                set_VSTL_groesse(k, get_groesse(get_vstl(k))
L^alleZK := sortiere_nach(L^alleZK, get_num_clusters(), get_clustering_bewertung(), -get_VSTL_groesse())
L^verbleibend := copy(L^alleZK )

# 2. Iteration
while True do
        # Initialisiere temporäre Variablen
        Zustand^aktuell := get_letztes_element(L^Pfad)
        Komp^betr := get_erstes_element(L^verbleibend)
        str^Bez := get_bezeichnung(Komp^betr )

        # Bestimme mögliche Aktionen
        S^moeglAkt := {0, ..., |D^AnzLaJeBez[str^Bez]|}
        for i in S^moeglAkt do
                if i in get_verbotene_aktionen(Zustand^aktuell) or komponente_aus_vstl_enthalten(D^KompJeLabel[(str^Bez, i)], get_vstl(Komp^betr)) then
                        entferne_element(S^moeglAkt, i)

        # Wähle eine Aktion aus und führe sie aus
```

```
46:     if |S^moeglAkt| > 0 then
47:         if |S^moeglAkt| == 1 then
48:             n^Akt := get_erstes_element(S^moeglAkt)
49:             D^AnzLaJeBez[str^Bez] := D^AnzLaJeBez[str^Bez] + 1
50:             D^KompJeLabel[(str^Bez, n^Akt )] := {Komp^betr}
51:             bool^NeuesLabel := True
52:         else
53:             S^moeglAkt := S^moeglAkt\{i|D^AnzLaJeBez[str^Bez]|}
54:             D^DisJeAkt := new Dict()
55:             for i in S^moeglAkt do
56:                 S^KompZuLabel := D^KompJeLabel[(str^Bez, i)]
57:                 s^mittel := 0
58:                 for Komp^2 in S^KompZuLabel do
59:                     s^mittel := s^mittel + (1−s^ZK(Komp^betr,Komp^2)) / |S^KompZuLabel|
60:                 D^DisJeAkt[i] := s^mittel
61:             n^Akt := arg_min(D^DisJeAkt)
62:             D^KompJeLabel[(str^Bez, n^Akt)] := D^KompJeLabel[(str^Bez, n^Akt)] ∪ {Komp^betr}
63:             bool^NeuesLabel := False
64:     else #Falls keine zulässige Aktion existiert
65:         if |L^verbleibend| == |L^alleZK| then
66:             break
67:         else
68:             backtracking(D^AnzLaJeBez, D^KompJeLabel, L^verbleibend, L^Pfad)
69:     entferne_erstes_element(L^verbleibend)
70:     fuege_element_hinzu(L^Pfad, new Zustand(Komp^betr, n^Akt, bool^NeuesLabel, new Set()))
71:
72:     # Beurteile den neuen Zustand und entscheide über das weitere Vorgehen
73:     b^Un := bestimme_untere_schranke(D^maxAnzZKJeBez, D^AnzLaJeBez)
74:     b^Ob := bestimme_obere_schranke(|L^verbleibend|, D^AnzLaJeBez)
75:     S^VSTLRed := reduziere_s_vstl(S^VSTL, D^KompJeLa)
76:     S^Wurzeln := bestimme_alle_wurzeln(S^VSTLRed)
77:     bool^zustandIstZulassessig := bool^NeuesLabel or (Alg^MSTL(S^Wurzeln, None) != None)
78:     bool^zustandIstVollstaendig := size(L^verbleibend) == 0
79:     bool^zustandIstUeberlegen := b^Ob < b^Min
80:     bool^zustandIstUnterlegen := b^Un ≥ b^Min
81:     if bool^zustandIstZulassessig then
82:         if bool^zustandIstUeberlegen then
83:             b^Min := b^Ob
84:             D^BesteZuordnung := kopieren(D^KompJeLabel)
85:         if bool^zustandIstVollstaendig or bool^zustandIstUnterlegen then
86:             backtracking(D^AnzLaJeBez, D^KompJeLabel, L^verbleibend, L^Pfad)
87:     else
88:         backtracking(D^AnzLaJeBez, D^KompJeLabel, L^verbleibend, L^Pfad)
89: return D^BesteZuordnung
```

Der Pseudocode geht von den folgenden Funktionen sowie von Standardfunktionen objektorientierter Programmiersprachen aus:

- bestimme_alle_ZK: Gibt eine Liste mit Referenzen auf alle ZKs von S^{VSTL} zurück

- bestimme_alle_bezeichnungen: Gibt eine Liste mit allen Bezeichnungen von ZKs in S^{alleZK} zurück
- bestimme_alle_ZK_je_bezeichnung: Gibt ein Dictionary zurück, das jeder Bezeichnung aus $L^{alleBez}$ die Menge zugehöriger ZKs aus L^{alleZK} zuordnet
- bestimme_max_anz_zk_je_bez: Gibt ein Dictionary zurück, das jeder Bezeichnung aus $L^{alleBez}$ die maximale Anzahl von ZKs mit dieser Bezeichnung in einer VSTL aus S^{VSTL} zuordnet
- kneedle: Gibt für eine Distanzmatrix die optimale Anzahl von Clustern nach dem Kneedle-Verfahren (siehe Kapitel 4.2.1.1.3) sowie die mittlere Intraclusterdistanz zurück
- bestimme_relative_aehnlichkeit: Ermittelt die relative kontextuelle Ähnlichkeit zweier Komponenten, wie in Kapitel 4.2.1.1.2 beschrieben
- sortiere_nach: Sortiert die Elemente einer Liste aufsteigend lexikographisch nach gegebenen Merkmalen
- komponente_aus_vstl_enthalten: Gibt an ob eine Menge von Komponenten eine Komponente aus einer bestimmten VSTL enthält
- backtracking: Führt ein Backtracking aus, wie in Kapitel 4.2.1.2 beschrieben, und aktualisiert dabei die als Argumente übergebenen Objekte
- bestimme_untere_schranke: Bestimmt die untere Schranke für einen Zustand, wie in Kapitel 4.2.1.2.3 beschrieben
- bestimme_obere_schranke: Bestimmt die obere Schranke für einen Zustand, wie in Kapitel 4.2.1.2.3 beschrieben
- reduziere_s_vstl: Leitet aus einer Menge von VSTLs die zugehörige Menge der reduzierten VSTLs ab
- bestimme_alle_wurzeln: Gibt die Wurzeln einer Menge von VSTLs zurück

Der Pseudocode geht darüber hinaus von der Klasse Zustand aus, die einen Zustand der Baumsuche und damit einen Knoten im Entscheidungsbaum darstellt. Objekte dieser Klasse speichern alle Informationen, um die Aktion, durch die ein Zustand erreicht wurde, im Zuge des Backtrackings rückgängig zu machen: die zuletzt getroffene Entscheidung, die Menge der in diesem Zustand verbotenen Aktionen sowie eine boolesche Variable, die angibt, ob zuletzt eine neue KK erstellt wurde.

A3.4 Heuristik zur Ermittlung der Anzahl von Platzhaltern für Schritt 2 der Methode 2

Analog zum Graph-Coloring-Problem hat die Anzahl n^{Pl} der Strukturoptionen-Platzhalter (STO-Platzhalter) einen großen Einfluss auf die Lösbarkeit und die Recheneffizienz des Optimierungsproblems 4.6 in Kapitel 4.2.2.2. Wird n^{Pl} zu klein gewählt, ist das Problem unlösbar. Wird n^{Pl} zu groß gewählt, steigt die Rechenzeit der Optimierung, weil die Anzahl der Variablen des Problems von n^{Pl} abhängt. n^{Pl} kann mit $n^{Pl} = |S^{VSTL}|$ trivial gewählt werden. Um die Effizienz der Optimierung zu steigern, kann jedoch, wie im Folgenden skizziert, heuristisch ein geringeres n^{Pl} bestimmt werden, das die Lösbarkeit des Problems garantiert. Um die Anzahl benötigter STO-Platzhalter zu bestimmen, kann zunächst die Zuordnung von Komponenten zu STO-Platzhaltern vernachlässigt und nur die Zuordnung von variantenbezogenen Stücklisten (VSTLs) zu STO-Platzhaltern berücksichtigt werden. Aus Schritt 1 der Methode 2 sind die KNs der ZKs der VSTLs bekannt. Die VSTLs aus S^{VSTL} werden in eine beliebige Reihenfolge gebracht. Zunächst wird ein STO-Platzhalter aktiviert und die erste VSTL diesem zugeordnet. Die zweite VSTL wird demselben Platzhalter zugeordnet, sofern dadurch in beiden VSTLs zusammen für keine Bezeichnung l mehr als n_l^{MaxB} verschiedene Klassennummern (KNs) vorliegen würden. Andernfalls wird ein weiterer Platzhalter eröffnet und die zweite VSTL diesem zugeordnet. Jede weitere VSTL wird stets dem ersten Platzhalter zugeordnet, für den ihre Zuordnung zu keiner Überschreitung von n_l^{MaxB} führt. Falls es keinen solchen Platzhalter gibt, wird ein neuer Platzhalter eingeführt. Ordnet man nun alle Zukaufkomponenten (ZKs) je VSTL genau dem Platzhalter zu, dem die VSTL zugeordnet ist, ergibt sich eine zulässige Lösung des Optimierungsproblems 4.6 in Kapitel 4.2.2.2. Diese ist nicht zwangsläufig optimal. Da jedoch eine Lösung mit der entsprechenden Anzahl von Platzhaltern existiert, können für die optimale Lösung nicht mehr Platzhalter benötigt werden. Aus der Anzahl der in der Heuristik verwendeten Platzhalter ergibt sich somit eine obere Abschätzung für die benötigte Anzahl von Platzhaltern im Optimierungsproblem 4.6 und n^{Pl} kann entsprechend gewählt werden.

A4 Anhang zu Kapitel 4.3

A4.1 Beschreibung und Pseudocode zu Algorithmus Alg^{MAPL}

Alg^{MAPL} erstellt für eine Menge S^{VAPL} von variantenbezogenen Arbeitsplänen (VAPLs) ohne Multivorgänge einen Maximalarbeitsplan (MAPL), aus dem alle VAPLs aus S^{VAPL}

konfiguriert werden können, falls die VAPLs keine Strukturalternativen (STAs) enthalten. Andernfalls gibt er einen Hinweis zurück, dass keine Lösung gefunden werden kann. Der Algorithmus wird im Folgenden zunächst hergeleitet. Es wird das Beispiel aus Abbildung 4.22 in Kapitel 4.3 zur Veranschaulichung verwendet. Dabei wird davon ausgegangen, dass die Arbeitsvorgänge (AVOs) mit Klassennummern (KNs) annotiert wurden, sodass sie eindeutig über ihre Labels identifiziert werden können (siehe Abbildung A4.1 (1)).

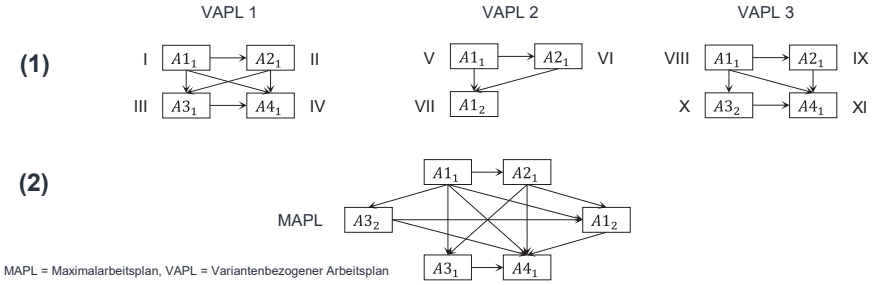


Abbildung A4.1: Variantenbezogene Arbeitspläne mit annotierten Arbeitsvorgängen und zugehörigem Maximalarbeitsplan für den Beispielfall

Ein VAPL i ist als eine Menge S_i^V von Knoten und eine Menge $S_i^{E1} \subseteq S_i^V \times S_i^V$ von Kanten definiert, die jeweils AVOs bzw. Vorrangbeziehungen darstellen. Existiert für zwei AVOs j und k eine Kante $(\mathrm{AVO}\ j, \mathrm{AVO}\ k)$ in S_i^{E1} bedeutet das, dass AVO j vor AVO k ausgeführt werden muss. Außerdem sei eine zu S_i^{E1} komplementäre Menge $S_i^{E0} \subseteq S_i^V \times S_i^V$ von gerichteten Kanten definiert, die angibt, welche Vorrangbeziehungen im Vorranggraphen nicht existieren. Existiert für AVO j und AVO k ein Element $(\mathrm{AVO}\ j, \mathrm{AVO}\ k)$ in S_i^{E0} bedeutet dies, dass AVO j nicht vor AVO k durchgeführt werden muss. Für VAPL 1 in Abbildung A4.1 (1) gilt z. B.

$$S_1^V = \{I, II, III, IV\}, \tag{A4.1}$$

$$S_1^{E1} = \{(I, II), (I, III), (I, IV), (II, III), (II, IV), (III, IV)\}, \tag{A4.2}$$

$$S_1^{E0} = \{(II, I), (III, I), (III, II), (IV, I), (IV, II), (IV, III)\}. \tag{A4.3}$$

Für einen zulässigen Vorranggraphen gilt immer

$$S_i^{E0} \cap S_i^{E1} = \emptyset. \tag{A4.4}$$

Außerdem ist ein zulässiger Vorranggraph immer zyklenfrei.

Aus dem resultierenden MAPL müssen alle VAPLs aus S^{VAPL} in einem Konfigurationsprozess wie in Kapitel 4.1.1 beschrieben konfiguriert werden können. Damit ein VAPL i sich aus dem MAPL konfigurieren lässt muss Folgendes gelten: Es existiert eine injektive Abbildung zwischen den AVOs aus VAPL i und den Arbeitsvorgangsklassen (AVKs) des MAPL, sodass zwischen den AVOs in VAPL i und ihren Bildern im MAPL die gleichen Vorrangbeziehungen bestehen.

Sei $S_i^{V,L}$ die Menge der Labels zu den in einem VAPL i vorhandenen AVOs. Seien $S_i^{E0,L} \subseteq S_i^{V,L} \times S_i^{V,L}$ bzw. $S_i^{E1,L} \subseteq S_i^{V,L} \times S_i^{V,L}$ die Vorrangbeziehungen in VAPL i bezogen auf dessen Labels. Für VAPL 1 gilt damit beispielsweise

$$S_1^{V,L} = \{A1_1, A2_1, A3_1, A4_1\}, \quad \text{A4.5}$$

$$S_1^{E1,L} = \{(A1_1, A2_1), (A1_1, A3_1), (A1_1, A4_1), (A2_1, A3_1), (A2_1, A4_1), (A3_1, A4_1)\}, \quad \text{A4.6}$$

$$S_1^{E0,L} = \{(A2_1, A1_1), (A3_1, A1_1), (A3_1, A2_1), (A4_1, A1_1), (A4_1, A2_1), (A4_1, A3_1)\}. \quad \text{A4.7}$$

Sei $S^{V,MAPL,L}$ die Menge der Labels der im MAPL vorhandenen AVKs. Seien $S^{E0,MAPL,L} \subseteq S^{V,MAPL,L} \times S^{V,MAPL,L}$ bzw. $S^{E1,MAPL,L} \subseteq S^{V,MAPL,L} \times S^{V,MAPL,L}$ die im MAPL geltenden Vorrangbeziehungen bezogen auf die Labels von dessen AVKs. Dann muss

$$S_i^{E0,L} \subseteq S^{E0,MAPL,L}, S_i^{E1,L} \subseteq S^{E1,MAPL,L} \quad \text{A4.8}$$

gelten, damit ein VAPL i aus einem MAPL konfiguriert werden kann. Das bedeutet, dass alle Vorrangbeziehungen, die im VAPL vorliegen, auch im MAPL vorliegen müssen und alle Vorrangbeziehungen, die im VAPL nicht vorliegen, auch im MAPL nicht vorliegen dürfen. Es kann nachvollzogen werden, dass dies für den MAPL und die VAPLs in Abbildung A4.1 gilt.

Für Probleme ohne Multivorgänge, wie z. B. das in Abbildung A4.1 (1) dargestellte Problem, lässt sich bestimmen, ob ein zulässiger MAPL existiert, indem die VAPLs aus S^{VAPL} wie folgt zu einem MAPL aggregiert werden:

$$S^{V,MAPL,L} = \bigcup_{i \in \{1,\dots,|S_{VAPL}|\}} S_i^{V,L}, \quad \text{A4.9}$$

$$S^{E0,MAPL,L} = \bigcup_{i \in \{1,\dots,|S_{VAPL}|\}} S_i^{E0,L}, \quad \text{A4.10}$$

$$S^{E1,MAPL,L} = \bigcup_{i \in \{1,\dots,|S_{VAPL}|\}} S_i^{E1,L}. \quad \text{A4.11}$$

Da es eine bijektive Beziehung zwischen Labels und AVKs des MAPL gibt, ergeben sich hieraus $S^{V,MAPL}$, $S^{E0,MAPL}$ und $S^{E1,MAPL}$. Der resultierende MAPL ist nicht zulässig, wenn $S^{E0,MAPL} \cap S^{E1,MAPL} \neq \emptyset$ gilt. In diesem Fall liegen in den VAPLs widersprüchliche Informationen darüber vor, ob bestimmte Vorrangbeziehungen gelten oder nicht. Der resultierende MAPL ist außerdem nicht zulässig, wenn er einen Zyklus enthält, da ggf. die VAPLs Informationen enthalten, die die Transitivität von Vorrangbeziehungen

verletzen. Ob in einem MAPL ein Zyklus vorliegt, kann z. B. mittels Tiefensuche[80] überprüft werden. In beiden Fällen kann aufgrund der Inkonsistenz der Vorrangbeziehungen in den zugrundeliegenden VAPLs kein zulässiger MAPL existieren, aus dem die VAPLs aus S^{VAPL} konfiguriert werden können.

Alg^{MAPL} bildet somit zunächst die Mengen $S^{V,MAPL,L}$, $S^{E0,MAPL,L}$ und $S^{E1,MAPL,L}$ und prüft dann, ob $S^{E0,MAPL} \cap S^{E1,MAPL} \neq \emptyset$ gilt oder der resultierende Graph einen Zyklus enthält. Je nachdem gibt er den Graphen, der sich aus $S^{V,MAPL,L}$, $S^{E0,MAPL,L}$ und $S^{E1,MAPL,L}$ ergibt, oder den Hinweis, dass keine Lösung existiert, zurück. Ist letzteres der Fall kann daraus geschlossen werden, dass die VAPLs aus S^{VAPL} STAs enthalten. Im Folgenden findet sich der Algorithmus in Pseudocode.

Alg^{MSTL}: Algorithmus zur Erzeugung eines Maximalarbeitsplans aus einer Menge von variantenbezogenen Arbeitsplänen ohne Multivorgänge und ohne Strukturalternativen

Input: S^{VAPL}

Output: G^{MAPL} oder None

1: $S^{V,MAPL,L} := \bigcup_{i \in \{1,\dots,|S_{VAPL}|\}} S_i^{V,L}$
2: $S^{E0,MAPL,L} := \bigcup_{i \in \{1,\dots,|S_{VAPL}|\}} S_i^{E0,L}$
3: $S^{E1,MAPL,L} := \bigcup_{i \in \{1,\dots,|S_{VAPL}|\}} S_i^{E1,L}$
4: **if** $S^{E0,MAPL} \cap S^{E1,MAPL} \neq \emptyset$ **then**
5: **return** None
6: G^{MAPL} := new MAPL($S^{V,MAPL,L}$, $S^{E0,MAPL,L}$, $S^{E1,MAPL,L}$)
7: **if** enthaelt_zyklus(G^{MAPL}) **then**
8: **return** None
9: **return** G^{MAPL}

Der Pseudocode geht von folgender Funktion sowie von Standardfunktionen objektorientierter Programmiersprachen aus:

- enthaelt_zyklus: Gibt wahr zurück, falls der gerichtete Graph, der als Argument übergeben wird, einen Zyklus enthält

Außerdem geht der Pseudocode davon aus, dass eine Klasse MAPL definiert ist, die einen MAPL in Form eines gerichteten Graphen darstellt, der durch die Mengen $S^{V,MAPL,L}$, $S^{E0,MAPL,L}$ und $S^{E1,MAPL,L}$ instanziiert wird.

[80] Für die Ermittlung von Zyklen in Graphen mittels Tiefensuche sei auf Tarjan (1973) verwiesen

A5 Anhang zu Kapitel 4.4

A5.1 Pseudocode zu Algorithmus $Alg^{InitRMP}$

$Alg^{InitRMP}$ akzeptiert eine Tabelle von Datenpunkten. Deren Spalten, mit Ausnahme der letzten, entsprechen den Features der Datenpunkte. Die letzte Spalte entspricht den Labels der Datenpunkte. Die Rückgabe ist ein relaxiertes reduziertes Master-Problem (XRMP).

$Alg^{InitRMP}$: Algorithmus zur Initialisierung eines RMP

Input: $T^{Training}$

Output: P^{XRMP}

```
#Initialisiere Variablen und den Stapel
L^Monome := new List()
L^AkzPosInst := new List()
L^Stapel := new List()
N^F:= get_anz_spalten(T^Training) -1
L^BetrTerm := {0}^(2*N^F) # Dual-Rail-Darstellung
anfuegen(L^Stapel,( T^Training, L^BetrTerm))

#Bearbeite den Stapel
while |L^Stapel| > 0 do
        T^TrainingTemp, L^BetrMonom := pop(L^Stapel)
        if alle_labels_gleich_eins(T^TrainingTemp) then
                #Das betrachtete Monom schließt alle negativen Datenpunkte aus
                L^Monome := anfuegen(L^Monome, L^BetrMonom)
                L^AkzPosInst := anfuegen(L^AkzPosInst, get_alle_datenpunkte(T^TrainingTemp))
        else
                #Ermittle den besten Split
                L^giniIndices := berechne_gini_index_fuer_jeden_split(T^TrainingTemp)
                n^bestesFeature := argmin(L^giniIndices)
                #Führe den Split aus
                T^TrainingTemp,0 := reduziere_datensatz(T^TrainingTemp, n^bestesFeature, 0)
                T^TrainingTemp,1 := reduziere_datensatz(T^TrainingTemp, n^bestesFeature, 1)
                L^BetrMonom,0 := copy(L^BetrMonom)
                L^BetrMonom,0[n^bestesFeature] := 0
                L^BetrMonom,1 := copy(L^BetrMonom)
                L^BetrMonom,1[n^bestesFeature] := 1
                L^Stapel := anfuegen(L^Stapel,( T^TrainingTemp,0, L^BetrMonom,0))
                L^Stapel := anfuegen(L^Stapel,( T^TrainingTemp,1, L^BetrMonom,1))
        end if
end while

#Erstelle das XRMP
P^RMP := erstelle_RMP(L^Monome, L^AkzPosInst)
```

34: P^{XRMP} := relaxiere_RMP(P^{RMP})
35: **return** P^{XRMP}, L^{Monome}

Der Pseudocode geht von den folgenden Funktionen sowie von Standardfunktionen objektorientierter Programmiersprachen aus:

- alle_labels_gleich_eins: Gibt wahr zurück, falls alle Labels einer Tabelle gleich 1 sind, ansonsten falsch
- berechne_gini_index_fuer_jeden_split: Gibt eine Liste zurück, die für jedes der Features den Gini-Index bei Split in diesem Feature enthält
- reduziere_datensatz: Reduziert eine Tabelle, indem alle Zeilen gelöscht werden, für die das angegebene Feature nicht den angegebenen Wert aufweist
- erstelle_RMP: Gibt ein reduziertes Master-Problem (RMP) zurück. Dessen Zielfunktionskoeffizienten entsprechen der Anzahl der Literale in den gegebenen Monomen und dessen Spalten geben wieder, welche der positiven Datenpunkte ein Monom akzeptiert.
- relaxiere_RMP: Relaxiert ein RMP, indem es die Ganzzahligkeitsbedingungen des RMP aufhebt

A5.2 Pseudocode zu Algorithmus Alg^{CG}

Der Algorithmus geht von einem bereits existierenden relaxierten reduzierten Master-Problem (XRMP), einer Liste der darin berücksichtigten Monome, einer Liste der hierfür ausgeschlossenen Monome und einem Trainingsdatensatz aus. Außerdem geht er von vier Matrizen aus, die angeben, welche Literale welche positiven bzw. negativen Datenpunkte ausschließen bzw. einschließen.

Alg^{CG}: Algorithmus zur Optimierung eines reduzierten Master-Problems mittels Spaltengenerierung
Input: P^{XRMP}, L^{Monome}, $L^{ExklTerme}$, $T^{Training}$, $A^{ExklPos}$, $A^{InklPos}$, $A^{ExklNeg}$, $A^{InklNeg}$
Output: P^{XRMP}, L^{Monome}, z^{RMP^*}, u^{RMP^*}, z^{XRMP^*}, u^{XRMP^*}

1: # Löse das Pricing-Problem
2: $bool^{VerbesserungMoeglich}$:= True
3: **while** $bool^{VerbesserungMoeglich}$ == True **do**
4: P^{DP} := dualisiere(P^{XRMP})
5: v^*:= optimiere(P^{DP})
6: L^{Monom}, z^{SP^*} := Alg^{PricingHeuristik}($L^{ExklMonome}$, v^*, $A^{ExklPos}$, $A^{InklPos}$, $A^{ExklNeg}$, $A^{InklNeg}$)
7: **if** $z^{SP^*} \geq 0$ **then**
8: P^{SP} := erstelle_pricing_problem($\boldsymbol{v}^*$, $T^{Training}$, $L^{ExklMonome}$)
9: L^{Monome}, z^{SP^*} := optimiere (P^{SP})

```
10:                 if z^SP* ≥ 0 then
11:                     bool^VerbesserungMoeglich := False
12:                 end if
13:             end if
14:             # Aktualisiere das relaxierte reduzierte Master-Problem
15:             if bool^VerbesserungMoeglich then
16:                 L^Monome := anfuegen(L^Monome, L^Monom)
17:                 P^XRMP := spalte_einfuegen(P^XRMP, L^Monom)
18:             end if
19:         end while
20:         # Löse das reduzierte Master-Problem
21:         z^XRMP*, u^XRMP* := optimiere(P^XRMP)
22:         if ist_ganzzahlig (u^XRMP*) then
23:             z^RMP* := z^XRMP*
24:             u^RMP* := u^XRMP*
25:         else
26:             P^RMP := derelaxiere(P^XRMP)
27:             z^RMP*, u^RMP* := optimiere(P^RMP)
28:         end if
29:         return P^XRMP, L^Terme, z^RMP*, u^RMP*, z^XRMP*, u^XRMP*
```

Der Pseudocode geht von den folgenden Funktionen sowie von Standardfunktionen objektorientierter Programmiersprachen aus:

- dualisiere: Gibt ein zu einem eingegebenen XRMP duales Problem zurück
- optimiere: Ermittelt den optimalen Zielfunktionswert und die optimale Lösung eines Optimierungsproblems
- erstelle_pricing_problem: Gibt ein Subproblem (SP) zurück
- spalte_einfuegen: Fügt eine Spalte in ein XRMP ein
- ist_ganzzahlig: Gibt wahr zurück, falls jeder Eintrag eines Vektors ganzzahlig ist, ansonsten falsch
- derelaxiere: Derelaxiert ein Optimierungsproblem, indem alle Entscheidungsvariablen ganzzahlig beschränkt werden

A5.3 Beschreibung und Pseudocode zu Algorithmus $Alg^{PricingHeuristik}$

Zur Lösung des Subproblems (SP) wird in der vorliegenden Arbeit die folgende Heuristik verwendet. Diese lässt sich zu großen Teilen auf Matrix-Vektor-Multiplikation und das Berechnen von Skalarprodukten zurückführen; Operationen, die nach Stand der Technik effizient ausgeführt werden können.

Die optimale Lösung $\boldsymbol{\nu}^*$ des DP gibt an, in welchem Maße ein Monom, das bestimmte positive Datenpunkte akzeptiert, zur Verbesserung des optimalen Zielfunktionswerts z^{XRMP^*} des relaxierten reduzierten Master-Problems (XRMP) beiträgt. Sie gewichtet damit die positiven Datenpunkte des Trainingsdatensatzes. Jedes zulässige Monom schließt alle negativen Datenpunkte aus, d. h. enthält für jeden negativen Datenpunkt mindestens ein Literal, das diesen ausschließt. Die Eigenschaft eines Monoms, Datenpunkte ein oder auszuschließen, kann auf seine Literale heruntergebrochen werden. Enthält das Monom ein bestimmtes Feature als positives Literal, schließt es alle Datenpunkte des Datensatzes, die für dieses Feature einen Eintrag falsch aufweisen, aus. Das gleiche gilt für negative Literale und Einträge wahr. Umgekehrt betrachtet schließt ein positives Literal ohne Berücksichtigung weiterer Literale des Monoms einen entsprechenden Eintrag wahr ein und ein negatives Literal einen Eintrag falsch. Ein Monom, das für das Beispiel in Abbildung 4.28 (1) in Kapitel 4.4.2.1.1 das Literal x_1 enthält, schließt z. B. die negativen Datenpunkte 1, 2 und 3 sowie den positiven Datenpunkt 2 aus. Demgegenüber schließt ein Monom, das das Literal x_1 enthält ohne Berücksichtigung weiterer Literale des Monoms den positiven Datenpunkt 1 ein.

Die Heuristik folgt der folgenden Überlegung. Ein gutes zulässiges Monom schließt alle negativen Datenpunkte aus, akzeptiert positive Datenpunkte mit einem möglichst hohen kumulierten Gewicht und verfügt über möglichst wenige Literale, da diese sich nachteilig auf z^{XRMP^*} auswirken. Das Monom als Lösung des Pricing-Problems wird Literal für Literal aufgebaut. Die Heuristik folgt einer Greedy-Strategie. In jeder Iteration wird dasjenige Literal hinzugefügt, das möglichst viele der verbleibenden negativen Datenpunkte ausschließt (Gewinn) und dabei möglichst wenige verbleibende positive Datenpunkte nach Gewicht ausschließt (Verlust). Für jedes infrage kommende Literal wird der Quotient aus Gewinn und Verlust berechnet. Es wird dasjenige Literal hinzugefügt, dass den höchsten Quotienten aufweist. Um eine Division durch 0 zu vermeiden und der Tatsache Rechnung zu tragen, dass jedes hinzugefügte Literal z^{XRMP^*} um 1 verschlechtert, wird zuvor der Verlust jeweils um 1 erhöht. Es werden so lange Literale hinzugefügt, bis alle negativen Datenpunkte ausgeschlossen sind und das Monom damit zulässig ist. Die Heuristik lässt sich recheneffizient umsetzen, indem für einen Trainingsdatensatz nur einmalig die Matrizen $A^{ExklPos}$, $A^{InklPos}$, $A^{ExklNeg}$ und $A^{InklNeg}$ berechnet werden, die angeben, ob ein Literal einen bestimmten positiven bzw. negativen Datenpunkt ausschließt bzw. akzeptiert. Jede der Matrizen verfügt über $2n^F$ Spalten, wobei jede Spalte einem möglichen Literal entspricht und über n^{Dp} bzw. n^{Dn} Zeilen,

wobei jede Zeile einem positiven bzw. negativen Datenpunkt entspricht. Die Gewinne und Verluste je Literal lassen sich damit durch Matrix-Vektor-Multiplikation berechnen. Der Pseudocode des Algorithmus $Alg^{PricingHeuristik}$, der im Folgenden gezeigt wird, gibt die Vorgehensweise der Berechnung wieder.

Alg^{CG}: Algorithmus zur heuristischen Lösung des Pricing-Problems

Input: $L^{ExklMonom}$, v^*, $A^{ExklPos}$, $A^{InklPos}$, $A^{ExklNeg}$, $A^{InklNeg}$, n^{In}, n^F

Output: $L^{bestesMonom}$, z^{SP^*}

1:
2: $L^{bestesMonom}$:= $\mathbf{0}^{2n^F}$ #Dual Rail-Darstellung
3: $L^{NegInstVerbleibend}$:= $\mathbf{1}^{n^{In}}$
4: $L^{PosInstVerbleibendGew}$:= $\boldsymbol{v}^*$
5:
6: # Vermeide, dass ausgeschlossene Terme eingefügt werden, indem sie wie negative Datenpunkte behandelt werden
7: **while** $|L^{ExklMonome}| > 0$ **do**
8: $L^{BetrMonom}$:= pop($L^{ExklMonome}$)
9: $A^{ExklNeg}$:= zeile_einfuegen($A^{ExklNeg}$, $L^{BetrMonom}$)
10: $A^{InklNeg}$:= zeile_einfuegen($A^{InklNeg}$, invertiere_eintraege($L^{BetrMonom}$))
11: **end while**
12:
13: #Iteration
14: **while** summe($L^{NegInstVerbleibend}$) > 0 **do**
15: #Berechne bestes Literal
16: L^{Gewinn} := $A^{ExklNeg} \cdot L^{NegInstVerbleibend}$
17: $L^{Verlust}$:= $A^{ExklPos} \cdot L^{PosInstVerbleibendGew} + \mathbf{1}^{2n^F}$
18: $\boldsymbol{r}^{Quotient}$:= elementweise_division(L^{Gewinn}, $L^{Verlust}$)
19: $\boldsymbol{r}^{Quotient}$:= elementweise_multiplikation($\boldsymbol{r}^{Quotient}$, invertiere_eintraege($L^{bestesMonom}$))
20: $n^{BesterIndex}$:= argmax($\boldsymbol{r}^{Quotient}$)
21: $L^{bestesMonom}[n^{BesterIndex}]$:= 1
22: #Aktualisiere verbleibende Datenpunkte
23: $L^{NegInstVerbleibend}$:= elementweise_multiplikation($L^{NegInstVerbleibend}$, $A^{InklNeg}[:, n^{BesterIndex}]$)
24: $L^{PosInstVerbleibendGew}$:= elementweise_multiplikation($L^{PosInstVerbleibendGew}$, $A^{InklPos}[:, n^{BesterIndex}]$)
25: **end while**
26: z^{SP^*} := summe($L^{bestesMonom}$) – summe($L^{PosInstVerbleibendGew}$)
27: **return** $L^{bestesMonom}$, z^{SP^*}

Der Pseudocode geht von den folgenden Funktionen sowie von Standardfunktionen objektorientierter Programmiersprachen aus:

- invertiere_eintraege: Ersetzt jeden Eintrag 0 in einer Liste mit 1 und umgekehrt

A5.4 Ergänzung von nichtlösbaren relaxierten reduzierten Master-Problemen

Relaxierte reduzierte Master-Probleme (XRMP), für die mindestens eine Variable auf 0 fixiert wurde, besitzen u. U. keine Lösung, da mit den verbleibenden wählbaren Monomen u. U. nicht mehr alle positiven Datenpunkte eingeschlossen werden können. Es ist ein Monom zu ermitteln, dass dem XRMP hinzugefügt werden kann um eine zulässige Lösung zu ermöglichen. Hierfür kann ein modifiziertes Subproblem (SP) verwendet werden. Grundsätzlich ist das SP in der Lage, dem XRMP Spalten für zulässige Monome hinzuzufügen und bevorzugt dabei solche Monome, die positive Datenpunkte mit großen absoluten v_i^* einschließen. Es wird deshalb ein $\boldsymbol{v}^*$ definiert, das jeweils einen hohen Wert für positive Datenpunkte aufweist, die im XRMP nicht eingeschlossen werden können und ansonsten 0 ist. Damit kann ein adaptiertes SP aufgestellt werden. Dieses SP führt zu einem Monom, das möglichst viele positive Datenpunkte akzeptiert, die zuvor von keinem Monom akzeptiert wurden. Auf diese Weise kann das XRMP schnell in ein lösbares XRMP überführt werden. Es ist dafür notwendig, zu ermitteln, welche positiven Datenpunkte im XRMP nicht eingeschlossen werden können, d. h. welche Nebenbedingungen unerfüllbar sind. Solver nach Stand der Technik verfügen über Funktionen, um die minimale Menge an unerfüllbaren Nebenbedingungen (Irreducible Inconsistent Subsystem) zu berechnen. Damit können die entsprechenden Datenpunkte ermittelt werden.

A5.5 Pseudocode zu Algorithmus Alg$^{B\&P}$

Der Algorithmus geht von einem Trainingsdatensatz aus. Optional können ein relaxiertes reduziertes Master-Problem (XRMP) sowie dessen berücksichtigte und ausgeschlossene Monome übergeben werden. Er gibt einen komplexitätsminimalen booleschen Ausdruck zurück, der auf dem Trainingsdatensatz eine perfekte Trainingsgenauigkeit aufweist.

Alg$^{B\&P}$: Algorithmus zum Lernen eines komplexitätsminimalen booleschen Ausdrucks mit perfekter Trainingsgenauigkeit mittels Branch and Price

Input: $T^{Training}$, optional: P^{XRMP}, L^{Monome}, $L^{ExklMonome}$

Output: B^{opt}

```
#Initialisiere Variablen
if not existiert(P^XRMP) or not existiert(L^Monome) then
        P^XRMP, L^Monome := Alg^InitRMP(T^Training)
end if
if not existiert(L^ExklMonome) then
```

```
6:          L^ExklMonome := new List()
7:    end if
8:    B^opt := None
9:    b̂^u := None
10:   L^Stapel := new List()
11:   L^Stapel := anfuegen(L^Stapel, (P^XRMP, L^Monome, L^ExklMonome))
12:   A^ExklPos, A^InklPos, A^ExklNeg, A^InklNeg := berechne_inkl_exkl_matrizen(T^Training)

13:   #Iterationen
14:   while |L^Stapel| > 0 do
15:         #Löse nächstes Problem auf dem Stapel
16:         P^XRMP,Knoten, L^Monome,Knoten, L^ExklMonome,Knoten := pop(L^Stapel)
17:         P^XRMP,Knoten, L^Terme,Knoten, z^RMP*, u^RMP*, z^XRMP*, u^XRMP* := Alg^CG(P^XRMP, L^Monome,Knoten,
            L^ExklMonome,Knoten, T^Training, A^ExklPos, A^InklPos, A^ExklNeg, A^InklNeg)
18:         #Prüfe ob neue beste Lösung gefunden wurde
19:         if B^opt == None or z^RMP* <b̂^u then
20:               B^opt := erstelle_booleschen_ausdruck(L^Terme,Knoten, u^RMP*)
21:               b̂^u := z^RMP*
22:         end if

23:         #Verzweige
24:         if ⌈z^XRMP*⌉ < b̂^u then
25:               n^Vz := niedrigster_index_nicht_ganzzahlig(u^XRMP*)
26:               P^XRMP,Knoten,0 := fixiere_variable(P^XRMP,Knoten, n^Vz, 0)
27:               L^ExklMonome,Knoten,0 := anfuegen(L^ExklMonome,Knoten,0, L^Monome,Knoten[n^Vz ]
28:               P^XRMP,Knoten,1 := fixiere_variable_auf_wert(P^XRMP,Knoten, n^Vz, 1)
29:               L^Stapel := anfuegen(L^Stapel , (P^XRMP,Knoten,0, L^Monome,Knoten, L^ExklMonome,Knoten,0))
30:               L^Stapel := anfuegen(L^Stapel , (P^XRMP,Knoten,1, L^Monome,Knoten, L^ExklMonome,Knoten))
31:         end if
32:   end while
32:   return B^opt
```

Der Pseudocode geht von den folgenden Funktionen sowie von Standardfunktionen objektorientierter Programmiersprachen aus:

- berechne_inkl_exkl_matrizen: Gibt für einen Trainingsdatensatz vier Matrizen, zurück, die angeben, welche Literale welche positiven bzw. negativen Datenpunkte ein- bzw. ausschließen
- erstelle_booleschen_ausdruck: Erzeugt einen booleschen Ausdruck, der aus einer Liste von Monomen alle Monome enthält, deren zugehöriger Eintrag in einem beigefügten Vektor 1 ist
- niedrigster_index_nicht_ganzzahlig: Gibt für einen Vektor den Index des ersten nichtganzzahligen Eintrags zurück

- fixiere_variable_auf_wert: Fixiert in einem Optimierungsproblem eine Variable mit einem bestimmten Index auf einen bestimmten Wert

A6 Anhang zu Kapitel 4.5

Im Folgenden wird näher auf zwei Aspekte der in Kapitel 4.5.2.5 beschriebenen Modellierung eingegangen.

A6.1 Verzicht auf dynamische Gewichtung der Auswahlkriterien bei der Variantenauswahl

Das Kriterium der Diversität steht in keiner unmittelbaren Beziehung zum Ausschließen von Modellen aus den Versionenräumen (VRs). Gerade für wenige Daten können jedoch Fälle auftreten, in denen bestimmte Labels durchgehend denselben Wahrheitswert annehmen. In diesen Fällen kann keine bessere Vorhersage als dieser Wahrheitswert getroffen werden, d. h. alle Modelle des entsprechenden VR sind identisch und können nicht separiert werden. Damit kann das Separationskriterium nicht angewandt werden. Um mindestens eine Variante mit einem anderen Wahrheitswert im Konfigurationsraum zu finden, eignet sich die Diversität. Die Diversität ist somit insbesondere beim Vorliegen weniger annotierter Datenpunkte relevant. Damit erscheint es zunächst sinnvoll, w^{MS} mit zunehmender Datenmenge zu erhöhen und damit die Gewichtung der Diversität zu reduzieren. Ein solcher Effekt ist jedoch in dem in Kapitel 4.5.2.5 beschriebenen Modell implizit angelegt. Für wenige Daten existieren zum einen viele triviale VRs, die nicht geteilt werden können. Zum anderen ist es möglich, Varianten weit weg von zuvor gewählten Varianten zu wählen, d. h. das Separationskriterium nimmt tendenziell geringe Werte an und das Diversitätskriterium tendenziell hohe Werte. Mit zunehmender Datenmenge verändert sich dieser Umstand zugunsten des Separationskriteriums. Um diesen Effekt zu nutzen, ist es sinnvoll, die beiden Kriterien über alle Iterationen hinweg mit demselben Maximalwert zu normieren und nicht mit dem für die jeweilige Iteration gültigen Maximalwert. Aufgrund dieses Effekts wird in der vorliegenden Arbeit auf eine dynamische Anpassung von w^{MS} in Abhängigkeit der Datenmenge verzichtet.

A6.2 Codierung kategorischer Merkmale für die Variantenauswahl

Liegen im High-Level-Konfigurationsmodell (HLKM) kategorische Produktmerkmale vor, können diese für das Optimierungsproblem zum einen mittels One-Hot-Codierung in binäre Features überführt werden. Zum anderen können den Merkmalausprägungen

zunächst fortlaufende natürliche Zahlen zugeordnet und diese in das Dualsystem überführt werden[81]. Letzteres wird im Folgenden als Dualcodierung bezeichnet. Da die Modelle in den Versionenräumen (VR) in One-Hot-Codierung erstellt werden, muss beim Aufstellen des Optimierungsproblems in Dualcodierung eine Transformation vorgenommen werden. Je nachdem, in welcher Form die Trainingsdaten vorliegen, müssen diese ebenfalls transformiert werden. Der Vorteil der Dualcodierung ist, dass sie weniger Variablen benötigt und weniger Kombinationen von Featureausprägungen ausgeschlossen werden müssen. Das ermöglich die Verwendung von weniger Nebenbedingungen 3 in dem in Kapitel 4.5.2.5 beschriebenen Optimierungsproblem 4.23. Es ist dabei jedoch zu beachten, dass sich je nach Darstellungsform das Kriterium der Diversität verändert. Stimmen zwei Varianten in einem kategorischen Merkmal nicht überein, bedeutet dies immer einen Abstand von 2 in One-Hot-Codierung. In einer Dualcodierung kann der Abstand zwischen 1 und $\lceil \log_2 n^{MW} \rceil$ betragen, wobei n^{MW} die Anzahl der möglichen Ausprägungen des Merkmals darstellt. Es liegt somit eine unbekannte Gewichtung von Abständen vor, die das Ergebnis auf nicht triviale Weise beeinflussen kann. Für Problemstellungen einer bestimmten Größe kann der Einsatz einer Dualcodierung aufgrund der Rechenzeit jedoch notwendig sein.

A7 Anhang zu Kapitel 5.1

A7.1 Approximation der wählbaren Varianten in den Konfigurationsmodellen des Industriepartners

Die Anzahl der wählbaren Varianten ohne Berücksichtigung der Beschränkungen des High-Level-Konfigurationsmodells (HLKM) ergibt sich durch Multiplikation der Anzahlen von zulässigen Ausprägungen je Merkmal. Die Anzahl der wählbaren Varianten unter Berücksichtigung der Beschränkungen wurde im Rahmen der vorliegenden Arbeit erstmals fundiert approximiert. Die paarweisen Ausschlussbeziehungen des HLKM lassen sich als High-Level-Formel (HLF), d. h. als ein boolescher Ausdruck formulieren, wie in Kapitel 4.5.2.4 beschrieben. Die Anzahl zulässiger Varianten entspricht damit der Anzahl der Variablenbelegungen, für die dieser Ausdruck den Wert wahr annimmt, d. h. der Anzahl der Modelle im Sinne der Aussagenlogik. Diese Anzahl zu bestimmen entspricht dem #SAT-Problem, das #P-vollständig[82] ist (Creignou & Hermann 1996, S. 1)

[81] Eine solche Darstellung kategorischer Merkmale wird z.B. von Potdar et al. (2017, S. 7–8) beschrieben.
[82] #P⊃NP

und deshalb für große Probleme nicht in relevanter Zeit gelöst werden kann. Es wurde deshalb Approximate Model Counting nach Soos et al. (2020) unter Verwendung der Python-Bibliothek pyapproxmc[83] eingesetzt, um die Anzahl der Varianten approximativ zu bestimmen. Es gilt

$$P(\frac{|sol(F)|}{1+\varepsilon}) \leq\ c\ \leq\ (1+\varepsilon)|sol(F)|)\ \geq\ 1-\delta \qquad \text{A7.1}$$

mit $\varepsilon\ =\ 0{,}05$ und $\delta\ =\ 0{,}05$ wobei $sol(F)$ die Menge der Modelle des zuvor erwähnten booleschen Ausdrucks darstellt (Soos et al. 2020, S. 465). Das bedeutet, dass mit einer Wahrscheinlichkeit von größer gleich 95 % die tatsächliche Anzahl zulässiger Varianten zwischen $(\frac{1}{1{,}05})\ *\ c$ und $1{,}05\ *\ c$ liegt, wobei c dem in Tabelle 5.1 in Kapitel 5.1 angegebenen Wert entspricht.

A8 Anhang zu Kapitel 5.2

A8.1 Gleichmäßig zufällige Generierung von synthetischen Maximalstücklisten

Im Folgenden wird erläutert, wie unter Berücksichtigung der in Kapitel 5.2.1 eingeführten Parameter gleichmäßig zufällig Maximalstücklisten (MSTLs) erstellt werden. Es werden die folgenden von den Parametern abgeleiteten Größen benötigt:

- $n^{Mult} = runden(n^{ZKK} * r^{Mult})$, die Anzahl von Multipositionen je Strukturoption (STO),
- $n^{Sing} = n^{ZKK} - n^{Mult}$, die Anzahl von singulären Positionen je STO,
- $n^{ZKK,MSTL} = n^{ZKK} + n^{ZKK} * (n^{STO} - 1)$, die Anzahl von Zukaufkomponentenklassen (ZKKs) in der MSTL,
- $n^{BGK} = runden(n^{ZKK,MSTL} * r^{BGK})$, die Anzahl von Baugruppenklassen (BGKs) in der MSTL
- und $n^{Abh} = r^{Abh} * n^{ZKK}$, die Anzahl von ZKKs einer STO, die von der gültigen STO abhängen.

Die Erzeugung der Struktur der MSTL lässt sich auf die gleichmäßig zufällige Erzeugung von Wurzelbäumen mit $n^{ZKK,MSTL}$ Blättern und n^{BGK} inneren Knoten zurückführen. Es existiert ein generisches Verfahren von Alonso et al. (1997) zur gleichmäßig

[83] https://github.com/meelgroup/approxmc (zuletzt überprüft am 07.06.2025)

zufälligen Erzeugung von Wurzelbäumen, die bestimmte Muster von Knoten und Kanten enthalten. Wird dieses Verfahren genutzt, um Bäume zu erstellen, die $n^{ZKK,MSTL}$-mal das Muster Knoten ohne Kanten zu untergeordneten Knoten und n^{BGK}-mal das Muster Knoten mit einer oder mehreren Kanten zu untergeordneten Knoten enthalten, entstehen gleichmäßig zufällige Bäume mit $n^{ZKK,MSTL}$ Blättern und n^{BGK} inneren Knoten, wobei die Wurzel selbst in n^{BGK} enthalten ist. Das Verfahren wurde auf Basis der Arbeit von Alonso et al. (1997) im Rahmen der vorliegenden Arbeit in Python implementiert.

Liegt die Struktur vor werden den Blättern des Baums Bezeichnungen und Positionsnummern zugeordnet. Es werden zunächst n^{Sing} Blätter gleichmäßig zufällig ausgewählt und jeweils mit einer eindeutigen Bezeichnung versehen. Es verbleiben n^{Mult} zu bezeichnende Blätter, denen dergestalt Bezeichnungen zuzuordnen sind, dass jede Bezeichnung mindestens zweimal auftritt. Dieses Problem kann in ein erweitertes Random-Integer-Partitioning-Problem überführt werden: Es ist eine Zerlegung von n^{Mult} in Summanden größer gleich zwei zufällig auszuwählen, sodass jede Zerlegung die gleiche Wahrscheinlichkeit besitzt ausgewählt zu werden. Ist diese Zerlegung bekannt, kann je Summand n^{Sum} eine einzigartige Bezeichnung ausgewählt und gleichmäßig zufällig n^{Sum} Blättern ohne Bezeichnung zugeordnet werden. Bereits für kleine n^{Mult} kann das erweiterte Random-Integer-Partitioning-Problem nicht mehr durch vollständige Enumeration aller möglichen Zerlegungen gelöst werden. Im Rahmen der vorliegenden Arbeit wurde deshalb ein Algorithmus entwickelt, um dieses Problem mittels dynamischer Programmierung effizient zu lösen. Dieser ist in Anhang A8.2 beschrieben.

Es sind nun n^{Mult} ZKKs mit Bezeichnungen versehen. Hiervon werden gleichmäßig zufällig n^{Abh} ausgewählt und deren Bezeichnungen $n^{STO} - 1$ mal kopiert und zufälligen ZKKs ohne Bezeichnungen zugeordnet. Die daraus resultierenden n^{STO} bezeichneten ZKKs werden jeweils einer STO zugeordnet. Damit ist jede ZKK der MSTL entweder genau einer oder allen STOs zugeordnet. ZKKs, die mehreren, aber nicht allen STOs zugeordnet sind, werden nicht betrachtet. Würden MSTLs zugelassen, für die bestimmte ZKKs mehreren, aber nicht allen STOs zugeordnet sind, erscheint es zunächst möglich, die von STOs betroffenen ZKKs zufällig ausgewählten STOs zuzuordnen. Dadurch könnten jedoch STOs mit einer geringeren Anzahl von ZKKs einer Bezeichnung entstehen, was der in Kapitel 4.2.2.2 getroffenen Annahme widersprechen würde.

Die vorgestellte Methode ließe sich somit in diesem Fall nicht anwenden um gleichmäßig zufällig MSTLs aus der Gesamtheit aller möglichen MSTLs auszuwählen.

In der vorgestellten Methode zur Erzeugung synthetischer MSTLs beeinflusst jede der vorgenommenen zufälligen Auswahlen die Anzahl der Möglichkeiten für die darauffolgende zufällige Auswahl nicht. Damit besitzt jede mögliche MSTL die gleiche Auswahlwahrscheinlichkeit. Nach Durchführung der beschriebenen Methode liegt somit eine gleichmäßig zufällig ausgewählte synthetische MSTL vor, die zur Konfiguration von variantenbezogenen Stücklisten (VSTLs) genutzt werden kann.

A8.2 Gleichmäßig zufällige Generierung von Ganzzahlpartitionen mit Summanden größer gleich 2

Die gleichmäßig zufällige Auswahl von Partitionen einer natürlichen Zahl, d. h. einer Menge von Summanden, die in Summe diese Zahl ergeben, ist ein in der Literatur bekanntes Problem, das rekursiv mittels dynamischer Programmierung gelöst werden kann. Zunächst wird berechnet, wie viele Ganzzahlpartitionen der natürlichen Zahl $n^{ZuTeilen}$ existieren, deren größter Summand genau einem bestimmten $n^{SumMax} \leq n^{ZuTeilen}$ entspricht. Anschließend wird der größte Summand der Partition mit einer Wahrscheinlichkeit ausgewählt, die dieser Anzahl entspricht. Die Berechnung der Anzahl von Partitionen von $n^{ZuTeilen}$ und einem bestimmten größten Summanden n^{SumMax} lässt sich auf untergeordnete Probleme desselben Typs zurückführen und somit rekursiv durchführen. (Nijenhuis & Wilf 1975, S. 70)

Der im Rahmen der vorliegenden Arbeit entwickelte und im Folgenden vorgestellte Algorithmus stellt eine Adaption des zuvor erläuterten Algorithmus von Nijenhuis & Wilf (1975, S. 70) dar und ist in der Lage, gleichmäßig zufällig Partitionen einer natürlichen Zahl auszuwählen, deren Summanden größer gleich 2 sind. Auch hier ist die effiziente Berechnung der Anzahl bestimmter Partitionen von natürlichen Zahlen wesentlich. Sei $n_{i,j}^{AnzPart}$ die Anzahl von Partitionen von i, deren kleinster Summand größer gleich j ist. Beispielsweise entspricht $n_{i,j}^{AnzPart}$ für $i = 8$ und $j = 2$ der Anzahl von Partitionen von 8 mit einem kleinsten Summanden von größer gleich 2. Für solche Partitionen können die folgenden Fälle unterschieden werden. Erstens kann der kleinste Summand der Partition genau j sein. Dann verbleibt neben dem kleinsten Summanden ein Rest von $i - j$, der auf die anderen Summanden aufzuteilen ist. Im Beispielfall würde die Partition somit den Wert 2 enthalten und es verbliebe ein Rest von 6, der auf die anderen

Summanden zu verteilen ist. Die Anzahl von Partitionen von 8 mit einem kleinsten Summanden von genau 2 entspricht $n_{6,2}^{AnzPart}$. Zweitens kann der kleinste Summand $j+1$ sein. Dann verbleibt neben dem kleinsten Summanden ein Rest von $i-(j+1)$, der auf die anderen Summanden aufzuteilen ist. Die Anzahl von Partitionen für den Beispielfall mit Summand 3 entspricht $n_{5,3}^{AnzPart}$ usw. Allgemein gilt

$$n_{i,j}^{AnzPart} = \sum_{k=j}^{i} n_{i-k,k}^{AnzPart}, \qquad \text{A8.1}$$

sowie $n_{0,j}^{AnzPart} = 1$ und $n_{i,j}^{AnzPart} = 0 \; \forall \; (j > i)$, womit eine rekursive Berechnung mittels dynamischer Programmierung möglich ist. Damit kann insbesondere die Anzahl von Partitionen einer natürlichen Zahl berechnet werden, deren kleinster Summand größer gleich 2 ist. Für den Beispielfall gilt $n_{8,2}^{AnzPart} = 7$. Dies entspricht den 7 Partitionen (2,2,2,2), (3,3,2), (4,2,2), (4,4), (5,3), (6,2) und (8). Die Anzahl von Partitionen, für die der kleinste Summand genau j ist, kann durch

$$n_{i,j}^{AnzPart,gen} = n_{i,j}^{AnzPart} - n_{i,j+1}^{AnzPart} \qquad \text{A8.2}$$

Berechnet werden. Für 8 existieren z. B. 4 Partitionen, deren kleinster Summand genau 2 ist. Um eine gleichmäßig zufällige Partition von 8 zu bestimmen ist somit der kleinste Summand mit einer Wahrscheinlichkeit von $\frac{4}{7}$ als 2 zu wählen und mit je einer Wahrscheinlichkeit von $\frac{1}{7}$ als 3, 4 oder 8. Allgemein beträgt die Wahrscheinlichkeit, dass ein bestimmter kleinster Summand j gewählt wird

$$P_j = \frac{n_{i,j}^{AnzPart,gen}}{n_{i,j}^{AnzPart}}. \qquad \text{A8.3}$$

Wurde ein kleinster Summand ausgewählt, wird die Partitionierung für den verbleibenden Rest fortgesetzt. Dabei darf kein Summand mehr gewählt werden, der kleiner als ein bereits gewählter Summand ist. Im Folgenden finden sich der Algorithmus $Alg^{ZufaelligePartition}$ zur Auswahl einer gleichmäßig zufälligen Partition einer natürlichen Zahl mit Summanden größer gleich 2 sowie der untergeordnete Algorithmus $Alg^{ZaehlePartitionen}$ zur Bestimmung von $n_{i,j}^{AnzPart}$ als Pseudocode. Der Algorithmus nutzt Caching, um die mehrfache Berechnung identischer Probleminstanzen zu vermeiden.

$Alg^{ZufaelligePartition}$**: Algorithmus zur gleichmäßig zufälligen Auswahl einer Partition einer natürlichen Zahl mit Summanden größer gleich 2**

Input: $n^{ZuTeilen}$

Output: $L^{ZufPartition}$

1: D^{Cache} := **new** Dictionary()

2: $L^{ZufPartition}$:= **new** List()
3: n^{Rest}:= $n^{ZuTeilen}$
4: $n^{LetzterGewaehlterSummand}$:= 2
5: **while** n^{Rest} > 0 **do**
6: $n^{PartGes}$:= Alg^{ZaehlePartitionen}(n^{Rest}, $n^{LetzterGewaehlterSummand}$, D^{Cache})
7: $D^{WkeitJeSummand}$:= **new** Dictionary()
8: **for** $k \in \{n^{LetzterGewaehlterSummand}, \ldots, n^{Rest}\}$ **do**
9: $D^{WkeitJeSummand}$[k] := (Alg^{ZaehlePartitionen}(n^{Rest}, k, D^{Cache}) - Alg^{ZaehlePartitionen}(n^{Rest}, k+1, D^{Cache})) * $\frac{1}{n^{PartGes}}$
10: **end for**
11: $n^{NeuerSummand}$:= zufalllsauswahl_mit_wkeiten($D^{WkeitJeSummand}$)
12: $L^{ZufPartition}$:= anfuegen($L^{ZufPartition}$, $n^{NeuerSummand}$)
13: n^{Rest}:= n^{Rest} - $n^{NeuerSummand}$
14: $n^{LetzterGewaehlterSummand}$:= $n^{NeuerSummand}$
15: **end while**
16: **return** $L^{ZufPartition}$

Der Pseudocode geht von den folgenden Funktionen sowie von Standardfunktionen objektorientierter Programmiersprachen aus:

- zufallsauswahl_mit_wkeiten: Die Funktion akzeptiert ein Dictionary, das auszuwählende Objekte und zugehörige Wahrscheinlichkeiten enthält und wählt zufällig eines der Objekte entsprechend den Wahrscheinlichkeiten aus.

Alg^{ZaehlePartitionen}**: Algorithmus zur Bestimmung der Anzahl von Partitionen für eine Zahl i, deren Summanden mindestens die Größe j aufweisen**

Input: i, j, D^{Cache}

Output: $n_{i,j}^{AnzPart}$

1: **if** i == 0 **then**
2: **return** 1
3: **end if**
4: **if** $j > i$ **then**
5: **return** 0
6: **end if**
7: **if** (i, j) **in** D^{Cache} **then**
8: **return** D^{Cache}[(i,j)]
9: **end if**
10: $n_{i,j}^{AnzPart}$:= $\sum_{k=j}^{i}$(Alg^{ZaehlePartitionen} (i - k,k, D^{Cache}))
11: D^{Cache}[(i,j)] := $n_{i,j}^{AnzPart}$
12: **return** $n_{i,j}^{AnzPart}$

A8.3 Metrik für die Demonstration der Methode 2

Um bewerten zu können, wie effektiv Methode 2 für die Lösung von Problem 2 ist, wird im Folgenden eine Metrik definiert. Diese quantifiziert die Abweichung einer Ergebnis-Maximalstückliste (Ergebnis-MSTL), die mittels Methode 2 erstellt wurde, von einer Referenz-Maximalstückliste (Referenz-MSTL). Die Herausforderung beim Vergleich einer Ergebnis-MSTL mit Strukturoptionen (STOs) und einer Referenz-MSTL mit STOs besteht darin, dass die Bezeichnungen der STOs willkürlich sind. Beispielsweise könnte im Falle zweier STOs je MSTL davon ausgegangen werden, dass die erste bzw. zweite STO der einen MSTL der ersten bzw. zweiten STO der anderen MSTL entspricht. Dadurch könnte eine große Differenz der beiden MSTLs schlicht daraus folgen, dass diese Zuordnung nicht korrekt ist. Es ist deshalb eine paarweise Zuordnung von STOs der beiden MSTLs zu ermitteln, die dafür sorgt, dass die Abweichung zwischen den MSTLs minimal wird. Eine STO einer MSTL kann als Subgraph der MSTL dargestellt werden, der nur diejenigen Komponentenklassen (KKs) enthält, die der STO zugeordnet sind (siehe Kapitel 4.2.2.1). Damit entsprechen STOs in ihrer Datenstruktur variantenbezogenen Stücklisten (VSTLs), d. h. Bäumen mit identifizierbaren Blattknoten. Für binäre blattannotierte Bäume (engl. Leave Labeled Trees) existieren in der Literatur Abstandsmaße, allerdings sind die betrachteten STOs nicht zwingend binäre Bäume. Die Berechnung von Tree-Edit-Distanzen für allgemeine ungeordnete, annotierte Bäume gehört zur Klasse der maximal streng NP-schweren Probleme (sog. MaxSNP-Probleme; Akutsu et al. 2011). Es hat sich im Rahmen der Arbeit bestätigt, dass keine ausreichend effiziente Umsetzung für die in Kapitel 5.2.1 beschriebene, umfassende Experimentreihe möglich ist. Es wird deshalb ein eigenes Maß entwickelt, um die Differenz zweier STOs zu quantifizieren.

Dem Maß liegt die Idee zugrunde, die Unähnlichkeit zweier Stücklisten danach zu beurteilen, wie viele nichtübereinstimmende Baugruppen sie aufweisen. Zwei Baugruppen stimmen nicht überein, wenn sie nicht dieselben Zukaufkomponenten enthalten. Da STOs keine instanziierten Baugruppen, sondern Baugruppenklassen enthalten, wird diese Idee auf Baugruppenklassen (BGKs) übertragen. Sei eine BGK beschrieben durch die Menge ihrer untergeordneten Zukaufkomponentenklassen (ZKKs), identifiziert durch deren Bezeichnung. Dies entspricht dem Cluster der BGK nach der Begriffsverwendung in Anhang A2.1. Da mehrere ZKKs mit derselben Bezeichnung in einer STO existieren können, sind die Cluster im Allgemeinen Multimengen. Liegen in beiden STOs dieselben Cluster vor, sind sie identisch. Um den Abstand $d_{k,l}^{STO}$ zweier STOs k

und l zu bestimmen, werden deren identische Cluster einander zugeordnet und die Anzahl der nichtzuordenbaren Cluster bestimmt. Sei S_k^{Cl} bzw. S_l^{Cl} die Menge aller Cluster in STO k bzw. l, dann gilt

$$d_{k,l}^{STO} = \left|S_k^{Cl}\right| + \left|S_l^{Cl}\right| - 2|S_k^{Cl} \cap S_l^{Cl}|, \quad \text{A8.4}$$

wobei das Symbol ∩ in diesem Fall den Schnittmengenoperator für Multimengen bezeichnet. Damit können die Abstände aller STOs der Ergebnis-MSTL zu allen STOs der Referenz-MSTL berechnet werden. Anschließend werden die STOs der Ergebnis-MSTL den STOs der Referenz-MSTL zugeordnet, so dass die Summe der Abstände einander zugeordneter STOs minimal wird. Dabei muss jede STO der Ergebnis-MSTL mindestens einer STO der Referenz-MSTL zugeordnet werden und umgekehrt. Dies entspricht einem Zuordnungsproblem des Operations Research (engl. Matching Problem, siehe hierzu Ren et al. 2021, S. 332–335), das mittels mathematischer Optimierung gelöst werden kann.

Sei $d^{MSTL,abs}$ der optimale Zielfunktionswert dieses Zuordnungsproblems. Um die Ergebnisse verschiedener Experimente vergleichbar zu machen, muss dieser Wert normiert werden, indem er durch einen Maximalwert $\hat{d}^{MSTL}$ dividiert wird. Grundsätzlich lässt sich $\hat{d}^{\mathrm{MSTL}}$ als Summe über alle Kardinalitäten $|S_k^{Cl}|$ aller STOs beider MSTLs (S^{STO}) berechnen, da im schlechtesten Fall alle Cluster aller STOs keine Entsprechung finden. Es ist jedoch zu berücksichtigen, dass die Anzahl der STOs in der Ergebnis-MSTL und der Referenz-MSTL nicht übereinstimmen muss. Ggf. muss eine STO einer MSTL mehreren STOs der anderen MSTL zugeordnet werden, wodurch $\hat{d}^{\mathrm{MSTL}}$ größer werden kann. Da von einer Minimierung der Distanz ausgegangen wird, wird im Falle, dass überhaupt keine Übereinstimmung zwischen STOs vorliegt, die STO mit der geringsten Anzahl von Clustern mehrfach zugeordnet. Sei d^{STO} die Differenz der Anzahlen von STOs in den beiden MSTLs. In der MSTL mit weniger STOs existiert eine STO, die mehrfach zugeordnet wird, weil sie die geringste Anzahl von Clustern besitzt. Sei n^{ClMin} deren Clusteranzahl. Dann gilt:

$$\hat{d}^{MSTL} = \sum_{k \in S^{STO}} \left|S_k^{Cl}\right| + d^{STO} * n^{ClMin}, \quad \text{A8.5}$$

Damit ergibt sich abschließend die normierte Distanz einer Ergebnis- und einer Referenz-MSTL als

$$d^{MSTL} = \frac{d^{MSTL,abs}}{\hat{d}^{MSTL}}, \quad \text{A8.6}$$

Da die Baugruppen einer MSTL die Struktur der MSTL definieren, ist diese Metrik geeignet, um die strukturellen Unterschiede der Ergebnis- und der Referenz-MSTL zu quantifizieren. Außerdem wird die Übereinstimmung der jeweils enthaltenen ZKKs indirekt berücksichtigt: Stimmen bestimmte ZKKs nicht überein, stimmen auch zwangsläufig deren übergeordnete BGKs nicht überein.

d^{MSTL} ist eine konservative Metrik für die Abweichung zweier MSTLs, da zwei BGKs als nichtübereinstimmend angesehen werden, auch wenn sie sich z. B. lediglich in genau einer untergeordneten ZKK unterscheiden. Sie hat jedoch den Vorteil, dass sie effizient berechnet werden kann und damit für die im Rahmen der Demonstration durchgeführte Experimentreihe auf synthetischen MSTLs geeignet ist.

A8.4 Zeitstudien zu Schritt 1 der Methode 2

Schritt 1 der Methode 2 basiert auf einer Baumsuche, die für große Problemstellungen sehr umfangreich und damit rechenintensiv sein kann (siehe auch Kapitel 5.2.2.2). Im Rahmen der vorliegenden Arbeit wurden verschiedene Funktionen entwickelt um die Recheneffizienz der Baumsuche zu steigern. Dadurch soll der Einsatz von Methode 2 für möglichst viele Anwendungsfälle ermöglicht werden. Durch die im Folgenden vorgestellten Zeitstudie wird quantifiziert, in welchem Umfang diese Funktionen zu einer Verringerung der Rechenzeit für Schritt 1 der Methode 2 beitragen. Die entwickelten Funktionen sind:

- die Festlegung einer Betrachtungsreihenfolge der Zukaufkomponenten (ZK, siehe Kapitel 4.2.1.1.4),
- die systematische Auswahl von Aktionen in den Entscheidungsknoten (siehe Kapitel 4.2.1.2.2),
- Pruning auf Basis einer unteren Schranke (siehe Kapitel 4.2.1.2.3),
- und Pruning auf Basis der Zulässigkeit von Teillösungen (siehe Kapitel 4.2.1.2.3).

Alle vier Funktionen folgen dem Prinzip, möglichst nur wenige Lösungen des Lösungsraums betrachten zu müssen.

Um den Einfluss dieser Funktionen auf die Rechenzeit zu quantifizieren, wird Schritt 1 der Methode 2 mehrfach für eine der in Kapitel 5.2.1 beschriebenen Experimentreihen angewandt und dabei jeweils eine der Funktionen deaktiviert. Die Funktion der systematischen Betrachtungsreihenfolge wird deaktiviert indem eine gleichmäßig zufällige Betrachtungsreihenfolge gewählt wird. Die Funktion der systematischen Auswahl von

Aktionen wird deaktiviert indem gleichmäßig zufällig zulässige Aktionen ausgewählt werden. Eine neue Klassennummer wird jedoch weiterhin nur dann hinzugefügt, wenn keine Zuordnung zu einer bestehenden Klassennummer möglich ist. Funktionen des Prunings werden deaktiviert, indem das entsprechende Pruning schlicht nicht durchgeführt wird, d. h. die entsprechenden Teilbäume werden nicht von der Betrachtung ausgeschlossen.

Die beispielhafte Experimentreihe besteht aus 10 Wiederholungen mit den Parameterausprägungen $r^{ZKMult} = 0$, $r^{AbhZKK} = 0{,}1$, $n^{STO} = 2$ und $r^{RadSTO} = 0$. n^{ZKK} wird innerhalb von $\{10, 20, 50, 100\}$ und n^{VSTL} innerhalb von $\{10, 20, 50, 100, 200\}$ variiert (zu den Parametern siehe Kapitel 5.2.1). Wie die Ergebnisse in Kapitel 5.2.2.2 bestätigen, weisen Fälle mit Strukturoptionen (STO) einen hohen Rechenaufwand auf. Deshalb wurde eine Experimentreihe mit STO gewählt. Im Gegensatz zu der in Kapitel 5.2.1 beschriebenen Experimentreihe wird die Rechenzeit auf 600 Sekunden je Experiment beschränkt. Außerdem werden die Berechnungen in Schritt 1 abgebrochen, sobald eine Maximalstückliste (MSTL) gefunden wurde, die nicht mehr Zukaufkomponentenklassen (ZKK) enthält, als die Referenz-MSTL. Damit ergibt sich, wie viel Rechenzeit jeweils benötigt wurde um eine solche MSTL zu ermitteln.

Abbildung A8.1 zeigt die Rechenzeiten für Schritt 1, jeweils gemittelt über alle Experimente und alle Wiederholungen der Experimentreihe. Es zeigt sich, dass die systematische Betrachtungsreihenfolge und das Pruning auf Basis von Zulässigkeitsprüfungen mit Abstand den größten Einfluss auf die Rechenzeit haben. Wenn eine der beiden

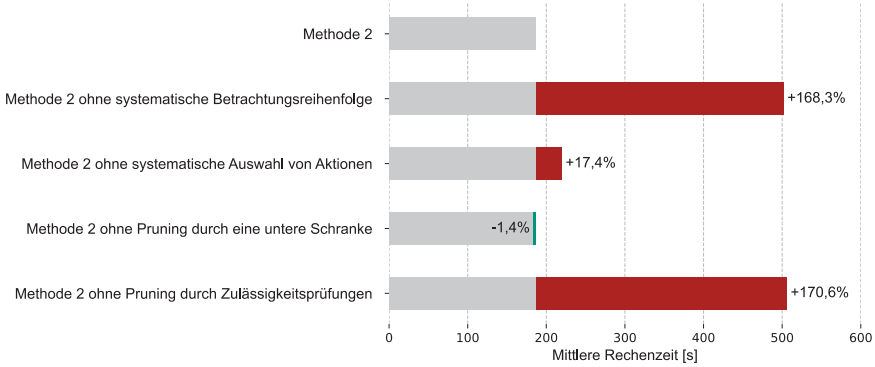


Abbildung A8.1: Ergebnisse der Zeitstudien zu Schritt 1 der Methode 2

Funktionen deaktiviert wird, wird ein Großteil der Experimente durch die Zeitbeschränkung von 600 Sekunden beendet. Die systematische Auswahl von Aktionen verringert die Rechenzeit ebenfalls, dieser Einfluss ist jedoch gering. Pruning auf Basis einer unteren Schranke hat keinen statistisch signifikanten Einfluss auf die Rechenzeit. Möglicherweise ist dies auch darauf zurückzuführen, dass die systematische Auswahl von Aktionen nur einen geringen Effekt hat. Somit ist es nicht möglich schnell gute Lösungen zu finden. Ein Pruning auf Basis einer unteren Schranke ist damit nur begrenzt möglich.

Insgesamt zeigt sich, dass es möglich ist, durch die im Rahmen der vorliegenden Arbeit entwickelten Funktionen, die Rechenzeit für die datenbasierte Erstellung von Maximalstücklisten zu reduzieren. Eine genauere Analyse der Effekte könnte zukünftig dazu beitragen, die Recheneffizienz von Methode 2 weiter zu erhöhen und damit ihren Einsatz für weitere Anwendungsfälle zu ermöglichen (siehe auch Ausblick in Kapitel 6.2.3).

A9 Anhang zu Kapitel 5.3

A9.1 Metrik für die Demonstration von Methode 3

Um bewerten zu können, wie effektiv Methode 3 für die Lösung von Problem 3 ist, wird im Folgenden eine Metrik definiert. Diese quantifiziert die Abweichung eines Ergebnis-Maximalarbeitsplans (Ergebnis-MAPL), der mittels Methode 3 erstellt wurde, von einem Referenz-Maximalarbeitsplan (Referenz-MAPL). Für den Vergleich von MAPLs besteht hinsichtlich Strukturoptionen (STOs) dieselbe Herausforderung wie für den Vergleich von MSTLs: Sowohl der Ergebnis-MAPL als auch der Referenz-MAPL können STOs aufweisen, deren Bezeichnungen beliebig sind (siehe Anhang A8.3). Ebenso wie in Anhang A8.3 beschrieben, werden die beiden MAPLs in ihre STOs zerlegt. Diese werden einander dergestalt paarweise zugeordnet, dass die Summe der Abweichungen zwischen einander zugeordneten STOs minimal wird. Es verbleibt die Definition eines Maßes für die Distanz zweier STOs in MAPLs. Dies entspricht einem Maß für die Distanz zweier gerichteter Graphen, wobei die Arbeitsvorgangsklassen (AVKs) jeweils durch ihre Bezeichnung identifiziert werden. Hierfür wird in Anlehnung an Malmi et al. (2015) das folgende Maß verwendet, das dem Anteil übereinstimmender Vorrangbeziehungen der beiden STOs entspricht[84]. Sei $S_k^{E1,Bez}$ die Menge der

[84] Die Bestimmung der Graph-Edit-Distanz auf allgemeinen annotierten Graphen ist ein NP-vollständiges Problem Bougleux et al. (2017, S. 38). Im Rahmen der vorliegenden Arbeit hat sich gezeigt, dass es nicht ausreichend

Vorrangbeziehungen (AVO_i^{Bez}, AVO_j^{Bez}) jeweils bezogen auf die Bezeichnungen zweier AVOs in einer STO k. Da in einer STO aufgrund von Multipositionen mehrere AVOs mit derselben Bezeichnung vorliegen können, sind $S_k^{E1,Bez}$ im Allgemeinen Multimengen. Für die Distanz zweier STOs gilt

$$d_{k,l}^{STO} = \left|S_k^{E1,Bez}\right| + \left|S_l^{E1,Bez}\right| - 2\left|S_k^{E1,Bez} \cap S_l^{E1,Bez}\right|, \quad \text{A9.1}$$

wobei das Symbol ∩ in diesem Fall den Schnittmengenoperator für Multimengen bezeichnet.

Sei S^{STO} die Menge aller STOs über beide MAPLs, d^{STO} die absolute Differenz der Anzahl von STOs in den beiden MAPLs und n^{AKMin} die minimale Anzahl von AVKs in einer STO derjenigen Seite mit der geringeren Anzahl von STOs (siehe Anhang A8.3). Dann gilt für die maximal mögliche kumulierte Distanz zwischen den paarweise zugeordneten STOs analog

$$\hat{d}^{MAPL} = \sum_{k \in S^{STO}} \left|S_k^{E1,Bez}\right| + d^{STO} * n^{AKMin}. \quad \text{A9.2}$$

Sei analog $d^{MAPL,abs}$ die Summe über die Distanzen der einander zugeordneten STOs, dann gilt für die relative Distanz der beiden MAPLs

$$d^{MAPL} = \frac{d^{MAPL,abs}}{\hat{d}^{MAPL}}. \quad \text{A9.3}$$

Die Distanz d^{MAPL} berücksichtigt unmittelbar nur die Abweichungen in den Strukturen der beiden MAPLs. Abweichungen in den AVKs werden jedoch mittelbar berücksichtigt, da Vorrangbeziehungen nur zwischen existierenden AVKs bestehen können.

A9.2 Gleichmäßig zufällige Generierung von synthetischen Maximalarbeitsplänen

Im Folgenden wird erläutert, wie unter Berücksichtigung der in Kapitel 5.3.1 eingeführten Parameter gleichmäßig zufällige Maximalarbeitspläne (MAPLs) erstellt werden. Es werden die folgenden von den Parametern abgeleiteten Größen benötigt:

- $n^{Mult} = runden(n^{AK} * r^{Mult})$, die Anzahl von Multipositionen je Strukturoption (STO)
- $n^{Sing} = n^{AK} - n^{Mult}$, die Anzahl von singulären Positionen je STO

effizient gelöst werden kann um die Graph-Edit-Distanz für die in Kapitel 5.2.1 beschriebene Experimentreihe anzuwenden. Approximationen der Graph-Edit-Distanzen wie z. B. diejenige von Bougleux et al. (2017) lassen die Genauigkeit der Approximation nicht erkennen.

- $n^{AK,MAPL} = n^{AK} + n^{AK} * (n^{STO} - 1)$, die Anzahl von Arbeitsvorgangsklassen (AVKs) im MAPL
- $n^{VBZ} = runden(n^{AK,MAPL} * r^{VBZ})$, die Anzahl von Vorrangbeziehungen im MAPL
- und $n^{Abh} = runden(n^{AK} * r^{Abh})$, die Anzahl von AVKs einer STO, die von der gültigen STO abhängen.

MAPLs sind gerichtete azyklische Graphen. Die Knoten gerichteter azyklischer Graphen können topologisch sortiert werden (Jungnickel 2013, S. 49–50) weshalb sich ihre Adjazenzmatrix als obere Dreiecksmatrix darstellen lässt. Die Struktur eines MAPL kann somit bestimmt werden, indem eine obere Dreiecksmatrix der Größe $n^{AK,MAPL}$ festgelegt wird. Dabei kommen nur solche Matrizen infrage, die für genau n^{VBZ} Einträge den Wert 1 aufweisen. Um die Struktur des MAPL gleichmäßig zufällig zu erstellen, können somit aus allen $n^{AK,MAPL} * (n^{AK,MAPL} - 1)$ Einträgen der oberen Dreiecksmatrix, die den Wert 1 aufweisen können, n^{VBZ} gleichmäßig zufällig ausgewählt und auf 1 gesetzt werden. Alle anderen Einträge werden auf 0 gesetzt. Damit liegt ein gleichmäßig zufällig ausgewählter gerichteter azyklischer Graph vor. Die Zuordnung von Bezeichnungen und STOs zu Knoten erfolgt analog zu Anhang A8.1. Zunächst werden n^{Sing} der Knoten zufällige Bezeichnungen zugeordnet. Anschließend wird n^{Mult} zufällig partitioniert und es werden den verbleibenden unbezeichneten Knoten mehrfach auftretende Bezeichnungen entsprechend der Summanden der Partition zugeordnet. Zuletzt werden n^{Abh} AVKs zufällig ausgewählt, kopiert und jede Kopie einer STO zugeordnet. Auch hier werden MAPLs, für die einzelne AVKs mehr als einer, aber nicht allen STOs zugeordnet sind, von der Betrachtung ausgeschlossen.

A9.3 Zeitstudien zu Schritt 1 der Methode 3

Die Zeitstudien für Schritt 1 der Methode 3 verfolgen denselben Zweck und folgen demselben Vorgehen wie die Zeitstudien für Schritt 1 der Methode 2 (siehe Anhang A8.4). Es werden dieselben vier Funktionen zur Verringerung des Rechenaufwands betrachtet. Die gewählte Experimentreihe weist die Parametrierung $r^{AKMult} = 0$, $r^{AbhAK} = 0{,}1$, $n^{STO} = 2$ und r^{RadSTO} = 0 auf. n^{ZKK} wird innerhalb von $\{10, 20, 50, 100\}$ und n^{VSTL} innerhalb von $\{10, 20, 50, 100, 200\}$ variiert (zu den Parametern siehe Kapitel 5.3.2).

Abbildung A9.1 zeigt die Rechenzeiten für Schritt 1, jeweils gemittelt über alle Experimente und alle Wiederholungen der Experimentreihe. Es zeigt sich, dass auch für Methode 3 die systematische Betrachtungsreihenfolge und das Pruning auf Basis von Zulässigkeitsprüfungen den größten Einfluss auf die Rechenzeit haben. Die Effekte der

systematischen Auswahl von Aktionen und des Prunings durch eine untere Schranke sind deutlich geringer. Sie sind dennoch jeweils stärker ausgeprägt als für Methode 2.

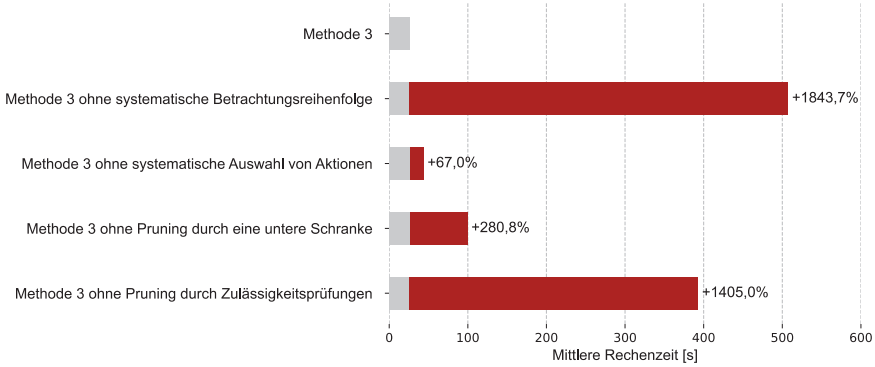


Abbildung A9.1: Ergebnisse der Zeitstudien zu Schritt 1 der Methode 3

Evtl. lassen sich die zu beobachtbaren Effekte zum Teil dadurch erklären, dass die Durchsuchung des Suchbaums für Methode 3 tendenziell zielgerichteter erfolgt als für Methode 2. Dadurch treten zum einen evtl. unzulässige Teillösungen tendenziell erst auf tiefen Ebenen des Suchbaums auf. Dadurch schließt das Pruning durch Zulässigkeitsprüfungen evtl. weniger folgende Entscheidungsknoten aus als für Methode 2. Zum anderen sind dadurch tendenziell nach weniger Iterationen bereits gute untere Schranken bekannt, so dass ein Pruning auf Basis einer unteren Schranke effektiver eingesetzt werden kann. Wenn jedoch das Durchsuchen des Suchbaums zielgerichteter erfolgt als für Methode 2 ist das nicht alleine auf eine bessere Auswahl von Aktionen zurückzuführen. Ansonsten müsste der Effekt der systematischen Auswahl von Aktionen stärker ausgeprägt sein.

Auch wenn Methode 3 für viele Anwendungsfälle bereits ausreichend recheneffizient ist (siehe Kapitel 5.3.2.2), können evtl. durch eine zukünftige genauere Analyse der Effekte Hinweise auf eine Erhöhung der Recheneffizienz für Methode 2 gewonnen werden.

A10 Anhang zu Kapitel 5.4

A10.1 Metriken für die Demonstration von Methode 4

Problem 4 entspricht einem Problem des überwachten Lernens (SL), weshalb einerseits die Genauigkeit auf einem Testdatensatz, als gängige Metrik des SL, zur Bewertung verwendet wird. Dafür wird zu Beginn jedes Durchlaufs ein Testdatensatz mit Größe $n^{Test} = 100$ in gleicher Weise wie der Trainingsdatensatz erstellt und mittels des Binary-Relevance-Ansatzes in einen Testdatensatz je Label aufgeteilt. Die Genauigkeit einer durch Methode 4 erstellten Regel entspricht dem Anteil der Datenpunkte des zugehörigen Testdatensatzes, deren Label durch das erstellte Modell korrekt wiedergegeben wird. Gemittelt über alle Labels ergibt sich die Metrik r^{GenIn} mit

$$r^{GenIn} = \frac{\sum_{i \in \{1,\dots,n^L\}} \sum_{j \in \{1,\dots,n^{Test}\}} b_{i,j}^{korrL}}{n^L n^{Test}} \quad \text{A10.1}$$

wobei $b_{i,j}^{korrL}$ angibt, ob das i-te Label des j-ten Datenpunkts des Testdatensatzes korrekt wiedergegeben wird und n^L der Anzahl von Labels im Testdatensatz entspricht.

Für Standardpositionen mit trivialen Regeln (siehe Kapitel 5.1) nehmen die Labels aller Datenpunkte immer den Wert wahr an. Auf Basis dessen wird sich durch Anwendung von Methode 4 immer ein Modell ergeben, das dem Wahrheitswert wahr entspricht und damit diese Positionen mit einer Genauigkeit von 100% korrekt vorhersagt. Je mehr Standardpositionen vorliegen, desto genauer sind die mit Methode 4 erstellten Regeln im Durchschnitt. Um dies zu berücksichtigen, wird eine weitere Metrik, r^{GenEx}, eingeführt, die sich ebenso wie r^{GenIn} berechnet, jedoch Genauigkeiten für triviale Regeln nicht berücksichtigt. r^{GenEx} ermöglicht damit die Betrachtung eines Worst-Case-Szenarios, in dem keine Standardpositionen in der Maximalstückliste und den Maximalarbeitsplänen des betrachteten Produkts vorliegen.

Da im Gegensatz zu typischen Problemen des SL die tatsächlichen Regeln, d. h. Modelle, je Position bekannt sind, besteht zum anderen die Möglichkeit, vorhergesagte Modelle mit den tatsächlichen Modellen zu vergleichen. r^{ModIn} gibt den Anteil der durch Methode 4 erstellten Modelle über alle Labels und alle Durchläufe hinweg an, die mit den tatsächlichen Modellen logisch übereinstimmen, d. h. derselben booleschen Funktion entsprechen.

Es gilt

$$r^{ModIn} = \frac{\sum_{i \in \{1,\dots,n^L\}} b_i^{korrM}}{n^L} \quad \text{A10.2}$$

wobei b_i^{korrM} angibt, ob das zum i-ten Label gehörige Modell korrekt erkannt wurde und n^L der Anzahl von Labels im Testdatensatz entspricht. Für r^{ModEx} werden im Gegensatz zu r^{ModIn} die oben genannten Standardpositionen nicht berücksichtigt. Die Maße r^{ModIn} und r^{ModEx} unterschätzen die Generalisierungsfähigkeiten der mittels Methode 4 erstellten Modelle tendenziell, da Unterschiede zu den tatsächlichen Modellen nicht zwangsläufig Fehlern im Konfigurationsmodell entsprechen. Die Modelle werden mit Datenpunkten trainiert, die zulässigen Varianten entsprechen. Ihre Vorhersagen für unzulässige Varianten sind somit nicht Gegenstand ihrer Optimierung. Ebenso berücksichtigen die tatsächlichen Regeln keine unzulässigen Varianten. Es ist somit möglich, dass eine Nichtübereinstimmung zwischen datenbasiert erstellter und tatsächlicher Regel nur unzulässige Varianten betrifft. Die Maße r^{ModIn} und r^{ModEx} dienen deshalb zur unteren Abschätzung der Effektivität der Methode 4.

A10.2 Ergebnisse der Demonstration der Methode 4 an Produkt A

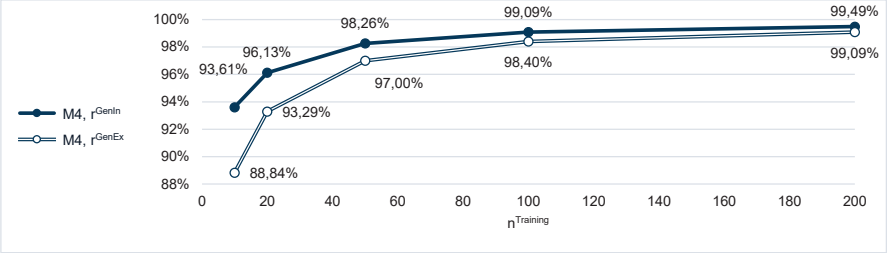


Abbildung A10.1: Ergebnisse der Demonstration an Produkt A hinsichtlich Testgenauigkeit

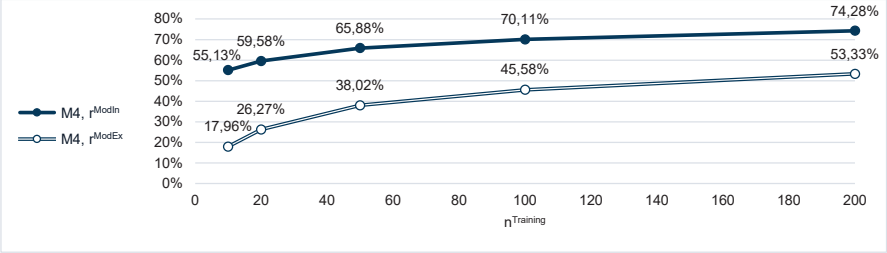


Abbildung A10.2: Ergebnisse der Demonstration an Produkt A hinsichtlich Modellübereinstimmung

A10.3 Benchmarking: Vergleich von Methode 4 mit Algorithmus Two Stage nach Ignatiev et al. (2021) hinsichtlich Recheneffizienz

Wie in Kapitel 3.4.2 beschrieben, verfolgt der Ansatz von Ignatiev et al. (2021) dasselbe Ziel wie Methode 4 der vorliegenden Arbeit: die Erstellung eines komplexitätsminimalen booleschen Ausdrucks, der eine perfekte Trainingsgenauigkeit für einen binären Trainingsdatensatz mit einem Label gewährleistet. Der Ansatz von Ignatiev et al. (2021) sieht ebenfalls eine mathematische Optimierung zur Auswahl der Monome des Ausdrucks vor. Im Gegensatz zur Methode 4 werden jedoch alle infrage kommenden Monome vorab berechnet, d. h. das Optimierungsproblem wird vollständig explizit aufgestellt. Dies lässt gegenüber der Methode 4, die die Monome nach Bedarf auf Basis von Spaltengenerierung (CG) erstellt, einen Nachteil hinsichtlich der Recheneffizienz erwarten. Dieser Nachteil wird im Folgenden quantifiziert, indem die als Two Stage bezeichnete Methode von Ignatiev et al. (2021) mit der Methode 4 hinsichtlich Recheneffizienz verglichen wird.

Metrik

Als Metrik für das Benchmarking wird die Rechenzeit verwendet, die für die Erstellung von booleschen Ausdrücken für gegebene Trainingsdatensätze benötigt wird. Um die Dauer des Benchmarkings in einem vertretbaren Rahmen zu halten, wird die Berechnung nach 300 Sekunden abgebrochen und die Rechenzeit auf 300 Sekunden festgelegt. Da die Verfahren beide eine minimale Komplexität und eine perfekte Trainingsgenauigkeit der erstellten booleschen Ausdrücke garantieren, werden Metriken hinsichtlich ihrer Generalisierungsfähigkeit nicht betrachtet; Unterschiede diesbezüglich wären zwangsläufig zufällig.

Experimentbeschreibung

Je Durchlauf werden Trainingsdatensätze aus dem Konfigurationsmodell des Produkts C generiert. Dafür werden gleichmäßig zufällig Varianten und somit Datenpunkte aus dem Konfigurationsraum ausgewählt. Diese werden über das Low-Level-Konfigurationsmodell (LLKM) mit Labels versehen. Der daraus resultierende annotierte Datensatz wird nach dem Binary-Relevance-Ansatz in Single-Lable-Datensätze aufgeteilt. Da die Methode Two Stage sehr lange Rechenzeiten aufweist werden je Durchlauf lediglich 10 gleichmäßig zufällig ausgewählte Labels betrachtet, wobei Labels von Standardpositionen nicht berücksichtigt werden. Der Algorithmus Two Stage wurde durch die von

Ignatiev et al. (2021) verwendete Bibliothek minds[85] implementiert um einen unverfälschten Vergleich der Rechenzeiten zu ermöglichen.

Ergebnisse

Abbildung A10.3 zeigt die Rechenzeiten für 10 zufällig ausgewählte Labels gemittelt über jeweils 10 Durchläufe für unterschiedliche Größen von $n^{Training}$. Sowohl für die Methode 4 als auch für Two Stage nehmen die Rechenzeiten mit zunehmender Größe des Trainingsdatensatzes zu. Die Zunahme ist jedoch für Two Stage deutlich stärker ausgeprägt als für Methode 4. Ab einer Größe des Trainingsdatensatzes von 50 äußert sich dies merklich. Die Rechenzeit beträgt hier ca. 76 Sekunden für Two Stage und ca. 3 Sekunden für Methode 4. Für $n^{Training} = 150$ liegt mit ca. 866 Sekunden gegenüber ca. 66 Sekunden ein Faktor von ca. 13 vor. Dabei müssen 28 % der Berechnungen für Two Stage vorzeitig abgebrochen werden, hingegen nur 1 % der Berechnungen für Methode 4. Auf Grund dieser deutlichen Diskrepanz werden keine weiteren Ausprägungen für $n^{Training}$ betrachtet. Es kann somit festgehalten werden, dass sich der in Kapitel 3.4.2 beschriebene Nachteil von Two Stage hinsichtlich der Recheneffizienz im Experiment bestätigt hat. Da sich dieser Nachteil bereits für praxisrelevante Problemstellungen mit $n^{Training} = 50$ auswirkt, ist Two Stage für die datenbasierte Erstellung von LLKMs nicht geeignet. Die im Rahmen der vorliegenden Arbeit entwickelte Methode 4 zeigt hingegen keinen vergleichbaren Nachteil hinsichtlich der Rechenzeit und kann deshalb für die datenbasierte Erstellung von LLKMs verwendet werden.

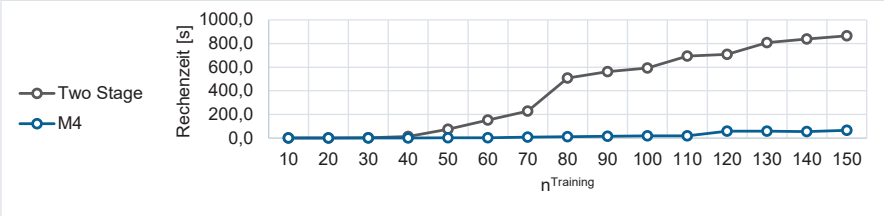


Abbildung A10.3: Benötigte Rechenzeit für die Erzeugung von 10 Modellen für die Methoden Two Stage und Methode 4 in Abhängigkeit der Größe des Trainingsdatensatzes, gemittelt über jeweils 10 Durchläufe

[85] https://github.com/alexeyignatiev/minds (zuletzt überprüft am 04.09.2024)

A10.4 Benchmarking: Vergleich von Methode 4 mit Algorithmus DK-XTSD nach Costamagna & Micheli (2023) hinsichtlich Generalisierungsfähigkeit

DK-XTSD ist ein Algorithmus von Costamagna & Micheli (2023) zur Bestimmung von booleschen Ausdrücken auf Basis eines annotierten Trainingsdatensatzes. Der Algorithmus erstellt gegenüber Algorithmen nach Stand der Forschung Ausdrücke mit überlegener Generalisierungsfähigkeit (Costamagna & Micheli 2023, S. 257–258). Er garantiert eine perfekte Trainingsgenauigkeit der Ausdrücke, jedoch im Gegensatz zur Methode 4 keine minimale Komplexität. Es wird untersucht, wie sich die Generalisierungsfähigkeit der durch Methode 4 erstellten Ausdrücke zu den durch DK-XTSD erstellten Ausdrücken verhält. Damit wird auch untersucht, ob komplexitätsminimale boolesche Ausdrücke über ihre gute Interpretierbarkeit hinaus auch Vorteile hinsichtlich Generalisierungsfähigkeit bieten. Dies ist für das Erzeugen von variantenbezogenen Stücklisten (VSTLs) und variantenbezogenen Arbeitsplänen (VAPLs) zu Varianten, die nicht Teil des Trainingsdatensatzes sind, relevant.

Metrik

Sowohl DK-XTSD als auch Methode 4 sind Methoden des überwachten Lernens (SL) auf Datensätzen mit einem Label. Die Generalisierungsfähigkeit dieser Methoden wird mit der in Kapitel 5.4.1 vorgestellten Metrik r^{GenEx} bewertet. Es wird ebenso wie für die Demonstration der Methode 4 ein Testdatensatz der Größe $n^{Test} = 100$ verwendet.

Experimentbeschreibung

Das Experiment entspricht der in Kapitel 5.4.1 beschriebenen Demonstration. Es werden Trainings- und Testdaten generiert wie in Kapitel 5.4.1 beschrieben und die Genauigkeit der beiden zu vergleichenden Methoden auf den Testdaten ermittelt. Der Algorithmus DK-XTSD erstellt boolesche Ausdrücke, indem er den Trainingsdatensatz schrittweise in ausgewählten Features teilt[86]. Die durch die Teilung entstehenden untergeordneten Trainingsdatensätze enthalten weniger Features als ihre übergeordneten Trainingsdatensätze. Unterschreitet die Anzahl von Features einen gewissen Grenzwert $n^{MinFeatures}$ wird der Datensatz nicht weiter geteilt, sondern mit dem von Chatterjee (2018) vorgestellten Algorithmus in einen booleschen Ausdruck überführt.

[86] Siehe Vorgehen zur Erstellung eines Entscheidungsbaums in Kapitel 4.4.1. Für eine vollständige Beschreibung des Algorithmus sei auf Costamagna & Micheli (2023) verwiesen.

Der finale boolesche Ausdruck ergibt sich als Komposition der booleschen Ausdrücke der Teiltrainingsdatensätze. Der Algorithmus von Chatterjee (2018) bildet den Trainingsdatensatz durch Wahrheitstabellen ab, wobei nur zufällig ausgewählte Features betrachtet werden. Die so erstellten Wahrheitstabellen werden Verknüpft und erneut durch Wahrheitstabellen abgebildet, sodass sich mehrere Ebenen von Wahrheitstabellen ergeben. Die Einstellparameter des Algorithmus sind die Anzahl der Wahrheitstabellen je Ebene und die Anzahl der Ebenen. Diese Parameter werden auf Basis der Ergebnisse von Chatterjee (2018) mit 1024 bzw. 6 gewählt. Der Parameter $n^{MinFeatures}$ wird durch die im Folgenden vorgestellte Parameterstudie ermittelt.

Für die Algorithmen von und Chatterjee (2018) und Costamagna & Micheli (2023) existiert kein öffentlich zugänglicher bzw. kein ausreichend dokumentierter, öffentlich zugänglicher Quelltext. Deshalb wurden beide Algorithmen im Rahmen der vorliegenden Arbeit auf Basis ihrer Darstellung in den Arbeiten der Autoren implementiert. Für die Parameterstudie und das Benchmarking wird die Rechenzeit für die Erstellung einer Regel auf $t^{Reg} = 100$ Sekunden begrenzt. Diese Zeit wurde jedoch von DK-XTSD in keinem Experiment erreicht, weshalb sie keinen Einfluss auf die Ergebnisse von DK-XTSD hat.

Parameterstudie

Die Parameterstudie folgt dem in Kapitel 5.4.1 beschriebenen Vorgehen, wobei jedoch ausschließlich der Algorithmus DK-XTSD mit unterschiedlichen Ausprägungen für $n^{MinFeatures}$ entsprechend einer Rastersuche eingesetzt wird. $n^{Training}$ wird mit 10, 20, 50, 100 und 200 gewählt und es werden jeweils 10 Durchläufe durchgeführt. Um eine Verzerrung der Ergebnisse des folgenden Benchmarkings zu vermeiden, wird die Parameterstudie nicht für die Konfigurationsmodelle (KMs) der Produkte A oder B,

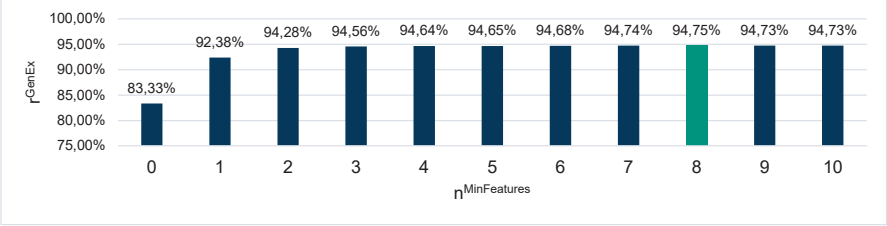


Abbildung A10.4: Ergebnisse für die Parameterstudie zu $n^{MinFeatures}$ des Algorithmus DK-XTSD

sondern für das KM des Produkts C durchgeführt. Abbildung A10.4 zeigt die Ergebnisse als r^{GenEx} gemittelt über alle Werte für $n^{Training}$ und alle Durchläufe. Auf Basis der Parameterstudie wird für das Benchmarking $n^{MinFeatures} = 8$ gewählt.

Ergebnisse des Benchmarkings

Tabelle A10.1 zeigt die Ergebnisse des Benchmarkings von Methode 4 und DK-XTSD nach der Metrik r^{GenEx} jeweils gemittelt über 10 Durchläufe. Für alle $n^{Training} \geq 20$ ergab sich aus den Experimenten ein höheres r^{GenEx} für Methode 4 gegenüber DK-XTSD. Wie der angegebene p-Wert eines zweiseitigen Welch-Tests zeigt, kann jedoch nur für Produkt A und $n^{Training} \geq 50$ die Nullhypothese einer identischen Verteilung zum Signifikanzniveau 5 % verworfen werden. Nur in diesem Fällen kann mit ausreichender statistischer Signifikanz davon ausgegangen werden, dass die durch Methode 4 erstellten Modelle eine höhere Generalisierungsfähigkeit aufweisen als die durch DK-XTSD erstellten. Für alle anderen Fälle kann keine verlässliche Aussage darüber getroffen werden, welche der beiden Methoden Modelle mit einer höheren Generalisierungsfähigkeit erstellt. Insgesamt kann somit geschlossen werden, dass die Methode 4 hinsichtlich der Generalisierungsfähigkeit ihrer erstellten Modelle mit Algorithmen nach Stand der Technik vergleichbar ist und in einigen Fällen sogar zu besseren Ergebnissen führt. Darüber hinaus garantiert Methode 4 im Gegensatz zu DK-XTSD minimale boolesche Ausdrücke und stellt damit eine gute Interpretierbarkeit sicher. Damit kann die Methode als überlegene Methode für die datenbasierte Erstellung von Regeln für Low-Level-Konfigurationsmodelle (LLKMs) angesehen werden.

Tabelle A10.1: Ergebnisse des Benchmarkings von Methode 4 und DK-XTSD nach der Metrik r^{GenEx}.

r^{GenEx}		$n^{Training}$				
		10	**20**	**50**	**100**	**200**
Produkt A	**Methode 4**	88,84 %	93,29 %	97,00 %	98,40 %	99,09 %
	DK-XTSD	88,95 %	93,18 %	96,79 %	98,15 %	98,92 %
	p-Wert	47,51 %	33,10 %	1,97 %	0,00 %	1,89 %
Produkt B	**Methode 4**	94,66 %	96,46 %	98,70 %	99,21 %	99,45 %
	DK-XTSD	94,69 %	96,44 %	98,47 %	99,02 %	99,18 %
	p-Wert	92,01 %	94,37 %	19,33 %	11,87 %	6,27 %

A11 Anhang zu Kapitel 5.5

A11.1 Parameterstudie zu Parameter w^{MS} der Methode 5

Parameter w^{MS} gibt die Gewichtung des Separationskriteriums bei der Auswahl einer Variante durch Methode 5 an. Die Gewichtung des Diversitätskriteriums ergibt sich daraus als $1 - w^{MS}$. Für die Parameterstudie wird eine Rastersuche mit Schrittweite $0{,}1$ verwendet. Je Parameterausprägung wird dasselbe Vorgehen gewählt wie für die in Kapitel 5.5.1 beschriebene Demonstration mit $\hat{n}^{Training} = 30$ und jeweils 10 Durchläufen. Die Rechenzeit je zu erstellender Regel wird auf $t^{Reg} = 10$ Sekunden begrenzt. Für die Auswahl der besten Parameterausprägung wird die in Kapitel 5.4.1 beschriebene Metrik r^{GenIn} verwendet. Um die Ergebnisse der Demonstration nicht zu verfälschen, wird die Parameterstudie auf dem Konfigurationsmodell (KM) des Produkts C durchgeführt.

Abbildung A11.1 zeigt die Ergebnisse der Parameterstudie unter Verwendung von 30 Datenpunkten, wobei r^{GenIn} je Parameterausprägung über 10 Durchläufe gemittelt wurde. Es zeigt sich, dass mit der Gewichtung des Separationskriteriums die Generalisierungsfähigkeit der gelernten Modelle zunächst zunimmt und für Werte größer als 0,5 abnimmt. Aus dem Vergleich der beiden Extrempunkte $w^{MS} = 0$ und $w^{MS} = 1$ zeigt sich, dass für den unikriteriellen Fall das Separationskriterium dem Diversitätskriterium geringfügig überlegen ist. Der Verlauf über alle Parameterausprägungen hinweg und das Optimum für $w^{MS} = 0{,}5$ zeigen jedoch, dass eine Komposition der beiden Kriterien einer unikriteriellen Bewertung vorzuziehen ist. Für die in Kapitel 5.5.1 beschriebene Demonstration wird $w^{MS} = 0{,}5$ gewählt.

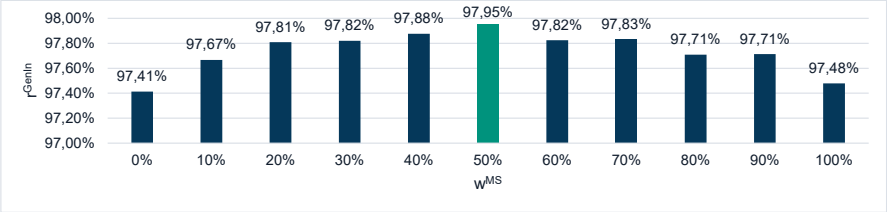


Abbildung A11.1: Ergebnisse der Parameterstudie für Parameter w^{MS} der Methode 5

A11.2 Ergebnisse der Demonstration der Methode 5 an Produkt A

Im Folgenden sind die Ergebnisse der Demonstration der Methode 5 an Produkt A dargestellt. Diese werden in Kapitel 5.5.2 referenziert.

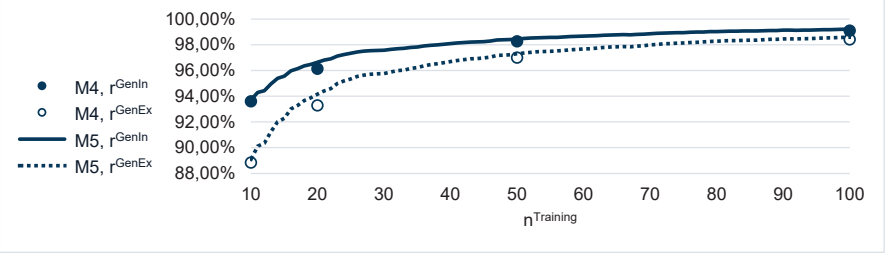


Abbildung A11.2: Ergebnisse der Demonstration der Methode 5 an Produkt A hinsichtlich Testgenauigkeit

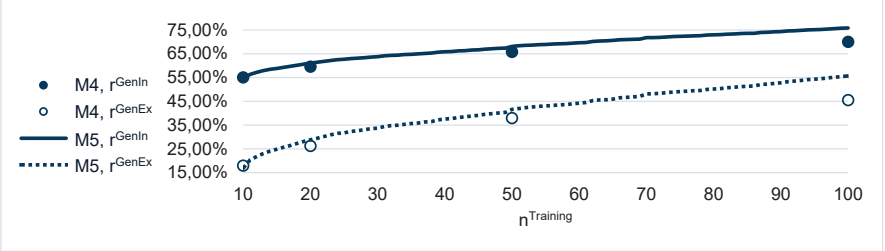


Abbildung A11.3: Ergebnisse der Demonstration der Methode 5 an Produkt A hinsichtlich Modellübereinstimmung

Tabelle A11.1: Ergebnisse der Demonstration der Methode 4 und Methode 5 an Produkt A im Detailvergleich

r^{GenEx}	$n^{Training}$				r^{ModEx}	$n^{Training}$			
	10	**20**	**50**	**100**		**10**	**20**	**50**	**100**
Methode 4	88,84 %	93,29 %	97,00 %	98,40 %	**Methode 4**	17,96 %	26,27 %	38,02 %	45,58 %
Methode 5	88,73 %	94,17 %	97,38 %	98,63 %	**Methode 5**	16,94 %	29,38 %	40,91 %	54,92 %
p-Wert	80,89 %	3,96 %	0,03 %	0,11 %	**p-Wert**	9,42 %	0,00 %	0,00 %	0,00 %

A12 Anhang zu Kapitel 5.6

A12.1 Erzeugung der Literal- und Termtabelle sowie Einbringung von Fehlern in diese Tabellen für die Demonstration der Methode 6

Im Folgenden wird erläutert, wie die Tabellen, die der Industriepartner verwendet um Regeln in seinen Konfigurationsmodellen (KMs) zu pflegen – im Folgenden als Regeltabellen bezeichnet – in Literal- und Termtabellen überführt werden. Darüber hinaus wird gezeigt, wie zum Zweck der Demonstration Fehler in diese Tabellen eingebracht werden.

Wie in Kapitel 5.1 beschrieben, sind die abhängigen Parameter die Aktivitätsparameter der Zukaufkomponentenklassen (ZKKs) und Arbeitsvorgangsklassen (AVKs) in der Maximalstückliste (MSTL) bzw. im Maximalarbeitsplan (MAPL). Diese hängen jeweils direkt von den Produktmerkmalen ab. Abbildung A12.1 zeigt schematisch den Aufbau einer Regeltabelle. Je abhängigem Parameter y liegen eine oder mehrere Zeilen in der Regeltabelle vor. Jede Zeile beschreibt einen Term. Ein abhängiger Parameter ergibt sich als disjunktive Verknüpfung seiner Terme. Jeder Term stellt eine konjunktive Verknüpfung von Aussagen bzgl. der Produktmerkmale dar, wie in Abbildung A12.1 unten beispielhaft dargestellt. Einige der kategorischen Merkmale sind positiv definiert, d. h., damit der zugehörige Term wahr ist, muss eine der angegebenen Ausprägungen angenommen werden. Einige der kategorischen Merkmale sind hingegen negativ definiert, d. h., damit der zugehörige Term wahr ist, darf keine der angegebenen Ausprägungen angenommen werden.

Abhängige Parameter	**Term #**	**Initiale Produktmerkmale**			
		Merkmal 1 (x_1^{init})	**Merkmal 2 (x_2^{init})**	**Merkmal 3 (x_3^{init})**	**…**
		Mehrwertig	**Kategorisch**	**Kategorisch**	**…**
y_1	1.1	$v_{1,1}$, $\neg v_{1,2}$	$\neg v_{2,1}$, $\neg v_{2,2}$	$v_{3,1}$, $v_{3,2}$	…
y_1	1.2	$v_{1,1}$, $v_{1,3}$	$\neg v_{2,1}$, $\neg v_{2,3}$	$v_{3,2}$	…
y_2	2.1	…	…	…	…
…	…	…	…	…	…

Bedeutung von Term 1.1: $x_1^{init} = (v_{1,1}, \neg v_{1,2}) \wedge x_2^{init} \notin \{v_{2,1}, v_{2,2}\} \wedge x_3^{init} \in \{v_{3,1}, v_{3,2}\} \wedge \cdots$

Abbildung A12.1: Schematische Abbildung einer Regeltabelle des Industriepartners

Um diese Tabelle mit Methode 6 verarbeiten zu können, wird sie in eine Literal- und eine Termtabelle überführt, wie in Abbildung A12.2 zu sehen. Jede Zeile der Literaltabelle – d. h. jeder Datenpunkt – entspricht einem Term der Regeltabelle, wobei identische Terme nur einmal auftreten und die Terme über alle Terme hinweg fortlaufend nummeriert werden. Die Spalten – d. h. Features – der Literaltabelle entsprechen booleschen Variablen, die die Produktmerkmale in One-Hot-Codierung darstellen. Die Einträge der Tabelle geben an, ob die entsprechende Variable in dem entsprechenden Term positiv (1), negiert (0) oder überhaupt nicht (○) auftritt. Boolesche Variablen, die demselben kategorischen Merkmal zugeordnet sind, treten entweder alle positiv oder alle negiert auf. Jeder Datenpunkt der Termtabelle entspricht einem abhängigen Parameter, d. h. einer Position der MSTL oder des MAPL. Die Features der Termtabelle geben an, welche Terme der Literaltabelle in der Regel des zugehörigen abhängigen Parameters auftreten. Grundsätzlich ist es möglich, dass verschiedene abhängige Parameter von denselben Termen abhängen und damit dieselben Ausprägungen der Features aufweisen. Diese werden in einer Zeile zusammengefasst.

Literaltabelle

Term #	Transformierte Produktmerkmale (Features)								
	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{3,1}$	$x_{3,2}$	…
	Mehrwertig			Kategorisch			Kategorisch		…
1	1	0	○	0	0	○	1	1	…
2	1	○	1	0	○	0	○	1	…
3	…	…	…	…	…		…	…	…
4	…	…	…	…	…		…	…	…
…	…	…	…	…	…		…	…	…

Bedeutung von Term 1: $(x_{11} \wedge \neg x_{1,2}) \wedge (\neg x_{2,1} \wedge \neg x_{2,2}) \wedge (x_{3,1} \vee x_{3,2}) \wedge \ldots$

Termtabelle

Abhängige Parameter	Term #				
	1	2	3	4	…
y_1	1	1	○	○	…
y_2	…	…	…	…	…
…	…	…	…	…	…

Bedeutung der Abhängigkeit von y_1: $Term\ 1 \vee Term\ 2 \vee \cdots \rightarrow y_1$

Abbildung A12.2: Schematische Darstellung der Literal- und Termtabelle

Durch Einsetzen des booleschen Ausdrucks aus der Literaltabelle für die Terme der Termtabelle ergibt sich ein boolescher Ausdruck je abhängigem Parameter. Dieser stellt dieselbe Information wie die Beschreibung der Regel in der Regeltabelle dar. Im

Gegensatz zu dem in Kapitel 4.6 verwendeten Beispiel, entspricht der boolesche Ausdruck keiner Normalform. Dies stellt jedoch kein Hindernis für die Anwendung der Methode 6 dar. Grundsätzlich ist es möglich, die Literal- und die Termtabelle dergestalt zu transformieren, dass sich Ausdrücke in Normalformen ergeben. Dabei würde jedoch die Beziehung zwischen den Einträgen der Tabellen und der Regeltabelle verloren gehen. Damit könnten Fehlerhinweise aus der Literal- oder der Termtabelle nicht unmittelbar Einträgen der Regeltabelle zugeordnet werden. Das würde die Auswertung für einen Experten erschweren. Um somit die Praxisrelevanz der Demonstration zu gewährleisten werden die Literal- und Termtabellen verwendet, die sich unmittelbar aus der Regeltabelle ergeben. Außerdem können auf diese Weise Fehler in die Literal- und die Termtabelle eingebracht werden, die unmittelbar Fehlern in der Regeltabelle entsprechen.

Für die Demonstration werden Fehler in die Literal- und die Termtabelle eingebracht. Für die Literaltabelle werden die Fehlerarten Negationsfehler, zusätzliche Variablen und fehlende Variablen betrachtet, wie in Kapitel 5.6.1 beschrieben (siehe Abbildung A12.6). Zum Einfügen von Negationsfehlern in die Literaltabelle wird ein gleichmäßig zufällig ausgewählter Eintrag 0 oder 1 für ein mehrwertiges oder boolesches Merkmal in 1 bzw. 0 geändert. Dies entspricht einem falschen Vorzeichen für ein Merkmal oder eine Merkmalausprägung in der Regeltabelle. Negationsfehler für kategorische Merkmale werden nicht betrachtet. Zusätzliche Variablen werden generiert, indem ein zufälliger Eintrag ○ zufällig auf 0 oder 1 gesetzt wird. Dies entspricht einem unzutreffenden Merkmal oder einer unzutreffenden Merkmalausprägung in der Regeltabelle. Fehlende Variablen werden generiert, indem ein zufälliger Eintrag 0 oder 1 in der Literaltabelle auf ○ gesetzt wird. Dies entspricht einem fehlenden Merkmal oder einer fehlenden

Term #	Transformierte Produktmerkmale (Features)								
	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{3,1}$	$x_{3,2}$	…
	Mehrwertig			Kategorisch			Kategorisch		…
1	1	~~0~~ 1	○	0	0	○	1	~~1~~ ○	…
2	1	○	1	0	~~0~~ 1	0	○	1	…
3	…	…	…	…	…		…	…	…
4	…	…	…	…	…		…	…	…
…	…	…	…	…	…		…	…	…

Negationsfehler — Zusätzliche Variable — Fehlende Variable

Abbildung A12.3: Einfügen von Fehlern in die Literaltabelle

Merkmalausprägung in der Regeltabelle. In der Termtabelle können fehlende und zusätzliche Monome auftreten. Diese werden analog zu fehlenden und zusätzlichen Variablen in der Literaltabelle generiert. Sie entsprechen fehlenden bzw. fälschlicherweise eingefügten Termen in der Regeltabelle.

A12.2 Parameterstudie zu dem in Methode 6 verwendeten Random-Forest-Algorithmus

In Methode 6 wird ein Random-Forest-Algorithmus eingesetzt, um Anomalien in der Literal- und der Termtabelle zu ermitteln. Ziel der Parameterstudie ist es, die Ausprägungen der Parameter des Random-Forest-Algorithmus zu ermitteln, die zu den besten Ergebnissen führen. Dabei wird die in Kapitel 5.6.1 eingeführte Metrik zugrunde gelegt. Die Studie erfolgt für die Literaltabelle und die Termtabelle separat.

Für die folgenden vier Parameter wird von einer großen Auswirkung auf die Ergebnisse ausgegangen. Sie werden deshalb für die Parameterstudie berücksichtigt.

- Anzahl n^{DT} der Entscheidungsbäume: Dieser Parameter hat einen Einfluss darauf, wie genau der Random-Forest die Daten des gegebenen Datensatzes abbildet. Bildet er die Daten zu genau ab, sind die Abweichungen gegenüber dem Modell, aus denen auf Anomalien geschlossen werden kann, gering und damit u. U. nicht ausreichend differenziert. Bildet er die Daten zu ungenau ab, lässt sich aus Abweichungen vom Modell nicht auf relevante Anomalien schließen. Entsprechend Sluban et al. (2014) wird $n^{DT} = 100$ betrachtet. Außerdem wird für die Parameterstudie eine deutlich größere Anzahl $n^{DT} = 200$ und eine deutlich kleinere Anzahl $n^{DT} = 50$ betrachtet.
- Auswahlstrategie $k^{Features}$ für Feature: Um Heterogenität bei der Erstellung von Entscheidungsbäumen zu gewährleisten, werden zur Erstellung eines Entscheidungsbaums nur eine gewisse Anzahl zufällig ausgewählter Features berücksichtigt. Diese Anzahl ist typischerweise die Wurzel ($k^{Feature} = sqrt$) oder der duale Logarithmus ($k^{Features} = log2$) der Anzahl aller Features. $k^{Features} = log2$ wählt i. d. R. weniger Features aus als $k^{Features} = sqrt$ und sorgt damit für eine höhere Heterogenität der Entscheidungsbäume. Das kann analog zur Anzahl n^{DT} für den vorliegenden Zweck vor- oder nachteilhaft sein.
- Auswahlstrategie $k^{Datenpunkte}$ für Datenpunkte: Um Heterogenität bei der Erstellung von Entscheidungsbäumen zu gewährleisten, werden zur Erstellung eines Entscheidungsbaums u. U. nur eine gewisse Anzahl zufällig ausgewählter

Datenpunkte berücksichtigt. Dies kann jedoch den Nachteil haben, dass Muster im Datensatz verloren gehen. Es wird deshalb der Fall untersucht, dass die Anzahl verwendeter Datenpunkte der Anzahl aller Datenpunkte ($k^{Datenpunkte} = 1$)[87] oder der Hälfte dieser Anzahl ($k^{Datenpunkte} = 0{,}5$) entspricht.

- Strategie $k^{Gewichtung}$ für die Gewichtung der Klassen: Die Einträge 0, 1 und ○, entsprechen in Schritt 3 der Methode 6 Klassen im Sinne des überwachten Lernens (SL). Sie treten nicht mit derselben Häufigkeit auf. Im Sinne des SL liegen somit unausgewogene Klassen vor. Um auch Minderheitsklassen beim Erstellen der Entscheidungsbäume ausreichend zu berücksichtigen, können Datenpunkte dieser Klassen mit höherer Wahrscheinlichkeit gewählt werden. Inwieweit dies für den vorliegenden Zweck sinnvoll ist, ist zu untersuchen. Es wird einerseits die Strategie $k^{Gewichtung} = Nicht\ gesetzt$ betrachtet, bei der keine Gewichtung vorgenommen wird, d. h. Mehrheitsklassen bei der Erstellung der Entscheidungsbäume auch stärker repräsentiert sind. Andererseits wird die Strategie $k^{Gewichtung} = Ausgeglichen$ betrachtet, bei der Datenpunkte aus Minderheitsklassen höheres Gewicht erhalten und damit ebenso stark repräsentiert sind wie Datenpunkte der Mehrheitsklasse.

Für die Parameterstudie wird eine Rastersuche mit den oben beschriebenen Ausprägungen der Einstellparameter verwendet. Je Kombination der Parameterausprägungen werden 10 Experimente durchgeführt, d. h. 10 manipulierte Tabellen erstellt und Fehler ermittelt. Um eine Verzerrung der Ergebnisse der späteren Demonstration zu vermeiden, wird die Parameterstudie nicht auf den Konfigurationsmodellen der Produkte A und B, sondern auf dem Konfigurationsmodell des Produkts C durchgeführt. Um einen vertretbaren Rechenaufwand zu gewährleisten, wird ausschließlich ein repräsentativer Fehlerfall mit $n^{Fehler} = 100$ und $k^{Fehlerart} = Gleichverteilt$, d. h. eine Gleichverteilung über alle möglichen Fehlerarten, betrachtet. Tabelle A12.1 und Tabelle A12.2 zeigen die Anzahlen der benötigten Überprüfungen zum Finden eines gewissen Anteils an Fehlern, gemittelt über 10 Durchläufe für die Literal- bzw. die Termtabelle. Für die Literaltabelle zeigt sich eine große Spannweite der Ergebnisse in Abhängigkeit der Parameterausprägungen. Für die Anteile 10 % bis einschließlich 90 % ist ein Muster in den Ergebnissen erkennbar. Beim Finden von 100 % der eingebrachten Fehler wird dieses

[87] Einzelne Datenpunkte können zufallsbedingt mehrfach ausgewählt werden, so dass nicht notwendigerweise alle Datenpunkte verwendet werden. Damit bleibt die Stochastik zur Erzeugung von heterogenen Entscheidungsbäumen erhalten.

Muster jedoch unterbrochen und es ist eine unverhältnismäßig große Anzahl von Überprüfungen notwendig[88]. Damit sind die Ergebnisse für den Anteil 100 % ohne praktische Relevanz und werden für den Vergleich der Parameterausprägungen nicht berücksichtigt. Je Anteil gefundener Fehler wird der Rang jeder Kombination an Parameterausprägungen berechnet, wobei Rang 1 der geringsten Anzahl von benötigten Überprüfungen entspricht. Die Ränge der Parameterausprägungen werden über alle Anteile gefundener Fehler gemittelt. Es zeigt sich, dass die Strategie $k^{Datenpunkte} = 0{,}5$ eine

Tabelle A12.1: Ergebnisse der Parameterstudie für den in Methode 6 eingesetzten Random-Forest-Algorithmus für das Finden von Fehlern in der Literaltabelle

Parameterausprägungen				Benötigte Überprüfungen für Anteil gefundener Fehler										Mittlerer Rang
n^{DT}	$k^{Features}$	$k^{Datenp.}$	$k^{Gewichtung}$	10 %	20 %	30 %	40 %	50 %	60 %	70 %	80 %	90 %	100 %	
100	sqrt	0,5	Ausgeglichen	82	149	211	262	330	398	474	586	1.091	69.644	2,7
200	sqrt	0,5	Ausgeglichen	99	155	228	270	327	401	475	603	1.061	79.131	3,4
200	sqrt	0,5	Nicht gesetzt	191	234	272	313	354	409	464	563	828	65.809	5,3
100	sqrt	0,5	Nicht gesetzt	167	220	267	312	363	427	479	583	803	72.181	5,3
200	log2	0,5	Ausgeglichen	104	169	235	286	336	413	504	789	1.750	85.513	6,3
100	log2	0,5	Ausgeglichen	81	167	238	297	362	435	502	691	1.251	78.669	6,4
50	sqrt	0,5	Ausgeglichen	72	143	221	281	362	458	591	850	2.014	76.837	7,0
50	sqrt	0,5	Nicht gesetzt	157	221	275	321	370	444	513	628	873	82.079	7,7
50	log2	0,5	Ausgeglichen	76	139	219	286	373	478	559	825	2.070	71.156	7,8
200	log2	0,5	Nicht gesetzt	200	244	287	324	368	428	496	620	906	73.465	8,0
100	log2	0,5	Nicht gesetzt	178	234	288	327	374	436	511	629	927	73.437	9,0
50	log2	0,5	Nicht gesetzt	177	235	287	337	399	475	546	672	1.099	80.364	10,2
200	sqrt	1	Nicht gesetzt	369	413	448	485	536	608	699	811	1.239	83.179	13,2
50	sqrt	1	Nicht gesetzt	314	382	436	483	548	649	763	978	1.926	100.146	14,1
100	log2	1	Nicht gesetzt	343	411	475	521	570	649	753	962	2.275	76.424	15,3
100	sqrt	1	Nicht gesetzt	398	469	519	562	612	673	730	880	1.665	58.350	16,0
200	log2	1	Nicht gesetzt	403	445	488	522	575	650	741	911	2.708	85.897	16,3
50	log2	1	Nicht gesetzt	326	401	479	541	620	709	869	1.215	3.296	62.086	16,8
200	log2	1	Ausgeglichen	458	525	583	637	706	799	1.035	4.801	16.517	79.443	20,0
200	sqrt	1	Ausgeglichen	497	552	610	665	720	790	900	2.682	17.943	72.410	20,7
100	log2	1	Ausgeglichen	439	519	592	652	771	1.107	1.928	5.759	18.445	61.725	21,2
100	sqrt	1	Ausgeglichen	513	583	650	713	810	1.007	1.315	3.839	14.083	70.326	21,7
50	sqrt	1	Ausgeglichen	431	513	628	809	1.107	1.906	2.989	6.442	17.789	59.926	21,8
50	log2	1	Ausgeglichen	469	587	740	870	1.274	1.995	3.375	7.746	18.198	76.046	23,7

[88] Dieses Phänomen wird in Kapitel 5.6.2 näher betrachtet.

Tabelle A12.2: Ergebnisse der Parameterstudie für den in Methode 6 eingesetzten Random-Forest-Algorithmus für das Finden von Fehlern in der Termtabelle

Parameterausprägungen				Benötigte Überprüfungen für Anteil gefundener Fehler										Mittlerer Rang
n^{DT}	$k^{Features}$	$k^{Datenp.}$	$k^{Gewichtung}$	10 %	20 %	30 %	40 %	50 %	60 %	70 %	80 %	90 %	100 %	
50	log2	0,5	Nicht gesetzt	295	341	382	416	453	107.166	118.657	131.561	145.216	160.279	3
100	sqrt	0,5	Nicht gesetzt	299	345	390	420	449	109.725	122.507	134.018	147.162	159.668	3,8
100	log2	0,5	Nicht gesetzt	307	344	383	419	450	109.326	121.172	132.341	144.719	159.831	4
200	log2	0,5	Nicht gesetzt	311	355	395	420	448	106.627	122.080	133.901	145.244	159.844	5,2
50	sqrt	0,5	Nicht gesetzt	301	349	391	420	455	110.373	122.062	132.260	146.354	160.037	5,6
200	sqrt	0,5	Nicht gesetzt	312	358	399	426	450	109.113	122.655	135.280	146.246	160.414	6,6
200	log2	1	Nicht gesetzt	323	373	413	442	476	109.399	121.686	133.618	148.300	159.654	9
50	sqrt	1	Nicht gesetzt	322	368	412	446	502	109.840	122.119	133.838	147.661	160.688	9,4
100	log2	1	Nicht gesetzt	328	376	414	452	490	107.298	121.672	132.863	145.911	159.673	11,2
100	sqrt	1	Nicht gesetzt	337	377	417	448	487	109.144	120.344	132.265	144.407	160.130	11,2
50	log2	1	Nicht gesetzt	323	375	418	452	499	106.215	119.377	131.269	146.903	160.283	11,4
200	sqrt	0,5	Ausgeglichen	262	434	1.133	3.714	8.960	110.880	122.036	133.624	146.305	159.623	11,6
200	log2	0,5	Ausgeglichen	248	505	1.479	3.907	8.716	108.893	119.017	132.179	146.382	159.888	11,8
200	sqrt	1	Nicht gesetzt	345	387	421	451	485	108.393	121.189	134.734	147.156	160.079	12
100	log2	0,5	Ausgeglichen	256	493	1.528	3.993	15.040	107.885	122.246	136.038	148.518	159.775	12,8
100	sqrt	0,5	Ausgeglichen	277	494	1.903	4.475	13.251	107.163	120.015	133.842	145.829	159.976	13,6
50	log2	0,5	Ausgeglichen	249	599	2.356	5.453	21.408	109.182	121.370	132.902	145.759	158.805	14,4
50	sqrt	0,5	Ausgeglichen	235	671	2.381	5.344	22.799	109.557	120.939	133.838	146.017	160.095	14,4
200	log2	1	Ausgeglichen	8.671	9.517	45.763	73.163	81.220	107.907	121.286	132.381	144.049	159.746	20,8
200	sqrt	1	Ausgeglichen	8.045	9.022	58.188	71.636	82.966	107.329	118.761	131.172	144.974	159.238	21
100	log2	1	Ausgeglichen	8.536	11.111	48.217	71.610	83.313	108.420	118.680	132.087	145.682	159.525	21
100	sqrt	1	Ausgeglichen	8.104	12.941	53.883	71.600	83.527	112.064	122.622	133.029	146.947	160.154	21,6
50	sqrt	1	Ausgeglichen	8.703	17.276	48.417	67.506	84.815	106.127	119.461	131.978	144.913	160.140	21,8
50	log2	1	Ausgeglichen	8.760	22.027	52.505	68.725	84.909	108.367	121.726	133.336	144.495	160.155	22,8

überlegene Strategie ist und einen großen Einfluss auf die Ergebnisse hat. Das lässt darauf schließen, dass für den Random-Forest, angewandt auf die Literaltabelle, eine Gefahr von Überanpassung besteht. Es scheint also Muster zu geben, die im Falle zu vieler berücksichtigter Datenpunkte von allen Entscheidungsbäumen abgebildet werden. Aufgrund der im Produkt vorliegenden technisch bedingten Gesetzmäßigkeiten erscheint das plausibel. Außerdem ist $k^{Features} = sqrt$ und $k^{Gewichtung} = Ausgeglichen$ zu bevorzugen. Dass mit $k^{Features} = sqrt$ eher mehr Features verwendet werden, kann darauf zurückzuführen sein, dass die Muster im Datensatz sich über mehrere Features erstrecken. Das ist ebenfalls naheliegend, da z. T. bestimmte abhängige Parameter

von vielen Ausprägungen bestimmter kategorischer Produktmerkmale abhängen. Der Unterschied zwischen 100 und 200 Entscheidungsbäumen ist gering. Lediglich weniger Entscheidungsbäume scheinen zu schlechteren Ergebnissen zu führen. Für die Demonstration werden die Parameterausprägungen $n^{DT} = 100$, $k^{Features} = sqrt$, $k^{Datenpunkte} = 0{,}5$ und $k^{Gewichtung} = Ausgeglichen$ gewählt.

Für die Termtabelle zeigt sich bereits für das Finden von 60 % der Fehler gegenüber 50 % der Fehler ein sprunghafter Anstieg[89]. Deshalb sind nur die Fehleranteile bis einschließlich 50 % von praktischer Bedeutung und werden für die Rangbildung berücksichtigt. Für die Termtabelle ergeben sich andere überlegene Parameterausprägungen als für die Literaltabelle. Insbesondere ist die Strategie $k^{Gewichtung} = Nicht\ gesetzt$ der Strategie $k^{Gewichtung} = Ausgeglichen$ überlegen. Da in vielen Fällen eine weitgehend exklusive Beziehung zwischen abhängigen Parametern und Monomen in der Regeltabelle besteht, enthält die Termtabelle überwiegen die Einträge 0. Die Unausgeglichenheit der Klassen im Datensatz ist deutlich stärker ausgeprägt als für die Literaltabelle. Diese Unausgeglichenheit der Daten durch eine ausgeglichene Gewichtung auszugleichen sorgt u. U. für eine zu starke Verzerrung der im Datensatz vorliegenden Muster, so dass relevante Abweichungen schlechter erkannt werden können. Auffällig ist außerdem, dass für die höchstrangige Kombination an Parameterausprägungen nur wenige Features und Datenpunkte genutzt werden und nur wenige Entscheidungsbäume trainiert werden. Sie weist somit einen großen Einfluss von Stochastik auf und neigt damit eher zu Unter- als zu Überanpassung. Das deutet darauf hin, dass die Herausforderung für die Termtabelle darin besteht, heterogene Entscheidungsbäume zu erstellen und damit überhaupt ausgeprägte Anomaliehinweise zu erhalten. Dass jedoch die Kombinationen der folgenden Ränge sich in ihren Ergebnissen nur unwesentlich von der Kombination mit dem höchsten Rang unterscheiden und mehr Entscheidungsbäume und z. T. auch mehr Features nutzen, relativiert die Bedeutung dieses Phänomens jedoch. Für die Demonstration werden die Parameterausprägungen $n^{DT} = 50$, $k^{Features} = log2$, $k^{Datenpunkte} = 0{,}5$ und $k^{Gewichtung} = Nicht\ gesetzt$ gewählt.

[89] Dieses Phänomen wird in Kapitel 5.6.2 näher betrachtet.

A12.3 Ergebnisse der Demonstration der Methode 6 an Produkt A

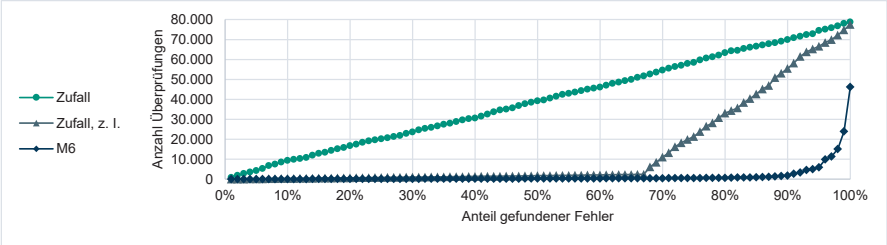


Abbildung A12.4: Demonstration der Methode 6 für die Literaltabelle von Produkt A mit $n^{Fehler} = 100$ und $k^{Fehlerart} = Gleichverteilt$; Gesamtansicht

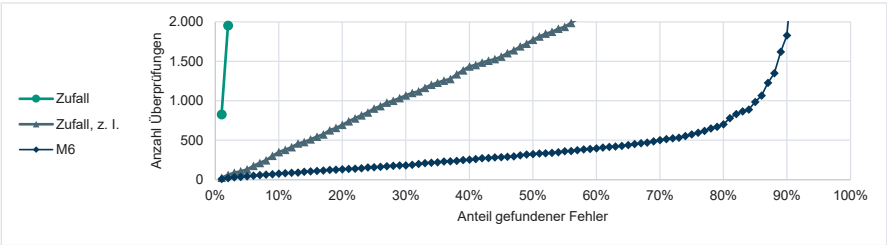


Abbildung A12.5: Demonstration der Methode 6 für die Literaltabelle von Produkt A mit $n^{Fehler} = 100$ und $k^{Fehlerart} = Gleichverteilt$; Detailansicht

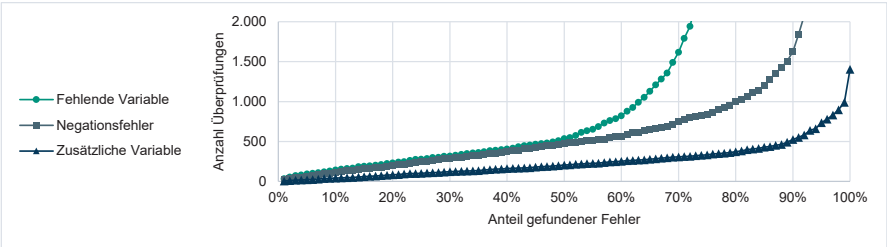


Abbildung A12.6: Demonstration der Methode 6 für die Literaltabelle von Produkt A mit $n^{Fehler} = 100$ und variierten Fehlerarten

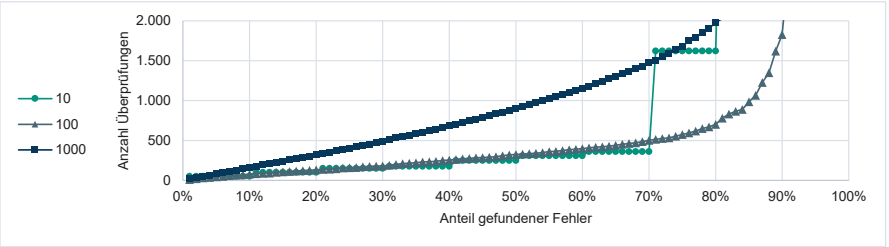


Abbildung A12.7: Demonstration der Methode 6 für die Literaltabelle von Produkt A mit $k^{Fehlerart} = Gleichverteilt$ und variierten Fehleranzahlen

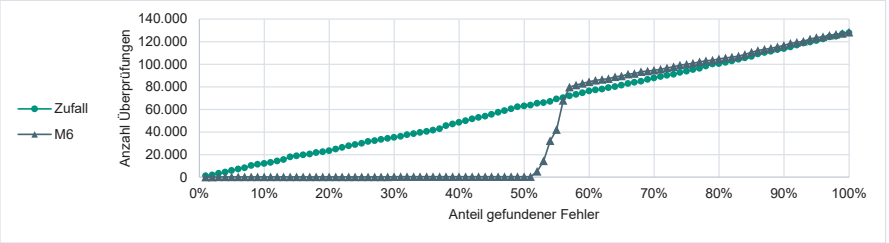


Abbildung A12.8: Demonstration der Methode 6 für die Termtabelle von Produkt A mit $n^{Fehler} = 100$ und $k^{Fehlerart} = Gleichverteilt$; Gesamtansicht

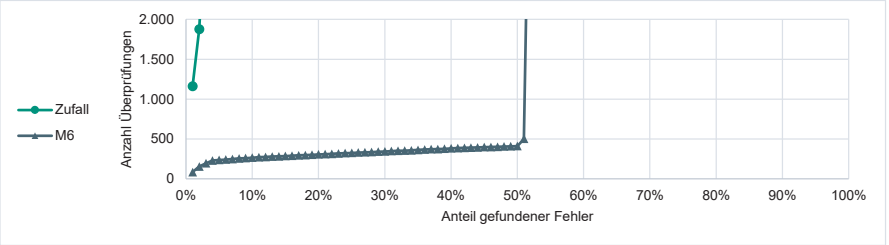


Abbildung A12.9: Demonstration der Methode 6 für die Termtabelle von Produkt A mit $n^{Fehler} = 100$ und $k^{Fehlerart} = Gleichverteilt$; Detailansicht

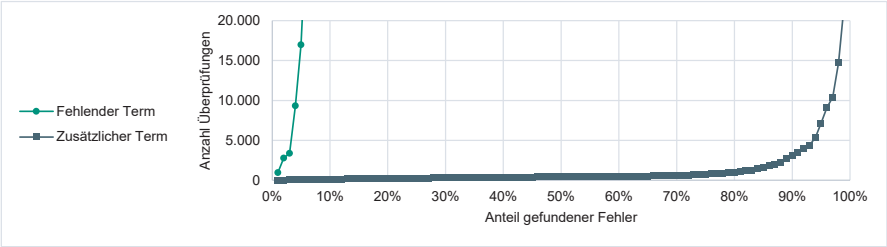


Abbildung A12.10: Demonstration der Methode 6 für die Termtabelle von Produkt A mit $n^{Fehler} = 100$ und variierten Fehlerarten

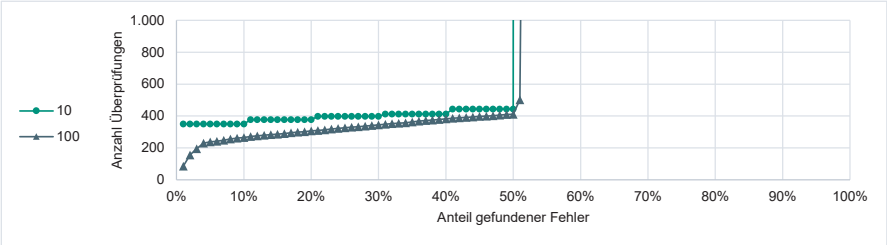


Abbildung A12.11: Demonstration der Methode 6 für die Termtabelle von Produkt A mit $k^{Fehlerart} = Gleichverteilt$ und variierten Fehleranzahlen

1000 Fehler werden in Abbildung A12.11 nicht betrachtet, da nicht ausreichend viele Terme in der Termtabelle vorliegen um 500 Fehler der Art fehlender Term einzubringen.

A12.4 Methode zur Ermittlung des Übergangs von einem linearen in einen nicht-linearen Abschnitt einer Kurve

Der Wert für den eine Kurve von einem linearen in einen nicht-linearen Verlauf übergeht wird in der vorliegenden Arbeit wie folgt ermittelt. Es wird die Nullhypothese aufgestellt, dass die Steigung zwischen zwei aufeinanderfolgenden Punkten derselben Normalverteilung entstammt wie vorangegangene Steigungen. Dabei werden die Mittelwerte vorangegangener Steigungen und deren Stichprobenstandardabweichungen als Schätzer verwendet. Beträgt die Konfidenz der Nullhypothese für zwei aufeinanderfolgende Steigungen weniger als 1 % wird davon ausgegangen, dass kein lineares Verhalten mehr vorliegt. Dieses Verfahren kann auch als Abbruchkriterium für den Einsatz von Methode 6 in der Praxis genutzt werden.