Machine-Learning-Verfahren in der Produktlinienoptimierung – Simulationsrechnungen und Robustheit

Von der Fakultät für Wirtschaft, Recht und Gesellschaft der Brandenburgischen Technischen Universität Cottbus–Senftenberg zur Erlangung des akademischen Grades eines Doktors der Wirtschaftswissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Wirtschaftsmathematiker
Sascha Vökler

geboren am 19.06.1984 in Cottbus.

Vorsitzender: Prof. Dr.-Ing. Uwe Meinberg
Gutachterin: Prof. Dr. Magdalena Mißler-Behr
Gutachter: Prof. Dr. Daniel Baier
Tag der mündlichen Prüfung: 28.07.2021

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACO	Ant-Colony-Optimization
API	Application Programming Interface
bspw.	beispielsweise
BTL	Bradley-Terry-Luce
bzgl.	bezüglich
bzw.	beziehungsweise
C	Programmiersprache
CBC	Choice-Based Conjointanalyse
d. h.	das heißt
GA	Genetischer Algorithmus
GAC	Genetischer Algorithmus mit Clusteransatz
ggf.	gegebenenfalls
ggü.	gegenüber
HB-Schätzung	Hierarchische Bayes-Schätzung
IIA	independence of irrelevant alternatives
i. d. R.	in der Regel

max	maximiere
MCS	Monte-Carlo-Simulation
MMAS	Min-Max Ant-System
MMAS ls	Min-Max Ant-System mit lokaler Suche
PSO	Particle-Swarm-Optimization
RLH-Wert	Root-Likelihood-Wert
SA	Simulated Annealing
sog.	sogenannt
u. a.	unter anderem
u. d. N.	unter den Nebenbedingungen
u. v. m.	und viele mehr
Wkt.	Wahrscheinlichkeit
z. B.	zum Beispiel

1 Einleitung

1.1 Problemstellung

1.1.1 Ausgangssituation

Ausgehend von der Arbeit von Zufryden (1977) entwickelte sich die Produktlinienoptimierung auf Basis der Conjointanalyse in den folgenden Jahren und Jahrzehnten rasant. Die Conjointanalyse (vgl. bspw. Debreu 1960, Luce/Tuckey 1964, Green/Rao 1969) bildet in der Marketingforschung den Goldstandard der Präferenzmessung von Konsumenten. Zur Durchführung einer Conjointanalyse werden im Vorhinein durch die Marketingforscher Eigenschaften (z.B. Farbe, Preis, etc.) und deren Eigenschaftsausprägungen (z.B. grün, blau, 5 EUR, 10 EUR, etc.) des interessierenden Produkts oder der interessierenden Dienstleistung definiert. Hierfür stehen verschiedene Arten der Conjointanalyse zur Verfügung. Die bedeutendsten Arten sind die traditionelle Conjointanalyse, die adaptive Conjointanalyse sowie vor allem die Choice-based Conjointanalyse mit einem Durchführungsanteil von 94% (vgl. Selka/Baier 2014, S. 59). Mit der Einführung der Choice-based Conjointanalyse (vgl. Louviere/Woodworth 1983) konnten Konsumenten vor realistische Kaufentscheidungen gestellt werden, indem vollständige Produktprofile abgefragt wurden. Durch eine multivariate Analyse war die dekompositionelle Schätzung der Konsumentenpräferenzen zunächst lediglich auf aggregierter Ebene möglich. Der Durchbruch der Choice-based Conjointanalyse als dominierende Art kam in den 1990er Jahren mit der Hierarchischen Bayes-Analyse (vgl. Allenby et al. 1995, Allenby/Ginter 1995, Lenk et al. 1996), welche es ermöglichte die Konsumentenpräferenzen auf individueller Ebene zu schätzen, d.h. es existiert ein Präferenzwert für jede Eigenschaftsausprägung einer Eigenschaft eines Produkts oder einer Dienstleistung, Teilnutzenwert genannt, für jeden Konsumenten. Auf Basis dieser Präferenzwerte, auch auf aggregierter Ebene oder von anderen Arten der Conjointanalyse, können Produkte und Dienstleistungen optimiert sowie optimale Produktlinien unter gewissen Nebenbedingungen bestimmt werden. Für die Bestimmung optimaler Produkte, Dienstleistungen und Produktlinien werden Modelle benötigt, die definieren, welche Ziele erreicht werden sollen unter welchen Bedingungen. So kann das Ziel eines Unternehmens sein, seinen Marktanteil für eine bestimmte Produktgruppe unter Berücksichtigung der Konsumentenpräfe-

renzen und eines Status-quo-Markts zu maximieren. Der Status-quo-Markt besteht dabei aus Produkten des eigenen Unternehmens sowie Konkurrenzunternehmen. Des Weiteren können die Gewinne einer Produktlinie unter Berücksichtigung von Deckungsbeiträgen, Kosten und der Konkurrenzsituation maximiert werden. Die Maximierung bzw. die Optimierung von Produktlinien geschieht also quantitativ auf Basis der mathematischen Optimierung.

Diese Produktoptimierungs- bzw. Produktlinienoptimierungsmodelle aus der Literatur verfolgen grundsätzlich zwei Ziele: Den Marktanteil oder den Gewinn eines Produkts oder einer Produktlinie unter Berücksichtigung der Konsumentenpräferenzen und des Status-quo-Markts zu maximieren. Zu den Marktanteilsmaximierungsmodellen können das *Buyers'-Problem* und das *Share-of-Choice-Problem* gezählt werden (vgl. Green/Krieger 1985, S. 3-5; Kohli/Sukumar 1990, S. 1467-1468). Das *Buyers'-Problem* maximiert zwar lediglich die Summe der Teilnutzenwerte der Produkte in der Produktlinie über alle Konsumenten, kann aber zu einem äquivalenten Marktanteilsmaximierungsmodell umgewandelt werden (vgl. Tsafarakis et al. 2011, S. 16). Der Beweis dieser Äquivalenz wird in Abschnitt 4.2.2 erbracht. Das *Share-of-Choice-Problem* zählt für die Produkte in einer Produktlinie diejenigen Konsumenten, die aufgrund eines höheren Nutzenwerts von ihrem bisherigen Status-quo-Produkt zu einem Produkt aus der Produktlinie wechseln würden. Diejenige Produktlinie mit der höchsten Anzahl an wechselnden Konsumenten ist die optimale Produktlinie. Betrachtet man die wechselnden Konsumenten als Anteil an der Gesamtzahl der Konsumenten, lässt sich ein prozentualer Marktanteil der optimalen Produktlinie bestimmen. Die Gewinnmaximierungsmodelle sind im Wesentlichen das *Seller's-Problem* (vgl. Green/Krieger 1985, S. 4-5; Kohli/Sukumar 1990, S. 1468) und das Modell von Gaul/Baier (2009). Das *Seller's-Problem* berücksichtigt dabei, je nach konkreter Problemformulierung, individuelle Teilstückdeckungsbeiträge für die Eigenschaftsausprägungen eines Produkts oder einen bestimmten Preis für ein spezielles Produkt (siehe z.B. Belloni et al. 2008b). Das Gewinnmaximierungsmodell von Gaul/Baier (2009) berücksichtigt neben individuellen Teilstückdeckungsbeiträgen auch anfallende Fixkosten für die Eigenschaftsausprägungen.

Die angesprochenen Modelle gliedern sich innerhalb der Marktanteils- und Gewinnmaximierung durch ein entscheidendes Merkmal auf: die Entscheidungsregeln für einen Konsumenten über die Vorziehenswürdigkeit eines Produkts. Die Aufgliederung erfolgt in deterministische und probabilistische Entscheidungsregeln (siehe 2.2 für Details). Aus jeder Gruppe der Optimierungsmodelle nutzt ein Modell eine deterministische und eine probabilistische Entscheidungsregel. Aus diesem Grund wird in dieser Arbeit von deterministischer und probabilistischer Marktanteilsmaximierung bzw. Gewinnmaximierung gesprochen. Die Aufteilung der Modelle erfolgt dabei, wie folgt:

- Deterministische Marktanteilsmaximierung: *Share-of-Choice-Problem*,

- deterministische Gewinnmaximierung: *Seller's-Problem*,
- probabilistische Marktanteilsmaximierung: *Buyers'-Problem*,
- probabilistische Gewinnmaximierung: Modell von Gaul/Baier (2009).

Nachfolgend wird in den Rahmenbedingungen erläutert, welche Herausforderungen sich für das Finden der optimalen Produktlinie für die vorgestellten Optimierungsmodelle ergeben und dabei die Brücke zu Lösungsverfahren mithilfe von Machine Learning geschlagen.

1.1.2 Rahmenbedingungen

Die Produktlinienoptimierung zählt zur Kombinatorischen Optimierung, da ihre Entscheidungsvariablen diskret, bzw. genauer binär, sind. D. h., es existiert eine endliche Anzahl von Produktlinien, die durch die Anzahl der Eigenschaften und deren Eigenschaftsausprägungen des betreffenden Produkts sowie durch die Anzahl der aufzunehmenden Produkte in die Produktlinie determiniert ist. Der Lösungsraum, also die Menge aller zulässigen Produktlinien, steigt sehr stark an, wenn es sich um Produkte mit vielen Eigenschaften und Eigenschaftsausprägungen handelt und sich die Anzahl der Produkte in der Produktlinie erhöht. Vor allem die Auswahl einer bestimmten Anzahl von Produkten für die Produktlinie aus der Menge der zulässigen Produktlinien lässt die Anzahl von Kombinationsmöglichkeiten rasant steigen. Diese Anzahl von Kombinationsmöglichkeiten macht es für viele Probleminstanzen unmöglich, die optimale Produktlinie in angemessener, sprich polynomieller, Zeit zu finden. Diese Eigenschaft von Kombinatorischen Optimierungsproblemen nennt sich NP-hard. Hierzu zählen klassische Optimierungsprobleme wie das Problem des Handlungsreisenden und das Rucksackproblem, aber auch das Produktlinienoptimierungsproblem (vgl. Kohli/Krishnamurti 1989). Aufgrund der endlichen Anzahl von zulässigen Produktlinien ist es zumindest theoretisch möglich, mittels kompletter Enumeration die Zielfunktionswerte aller zulässigen Produktlinien zu bestimmen und diejenige auszuwählen, die den besten Zielfunktionswert liefert (höchster Marktanteil oder Gewinn). Des Weiteren können Lösungsstrategien aus der Ganzzahligen Optimierung, wie Lagrange-Relaxation, Branch-&-Bound-Verfahren und Branch-&-Cut-Verfahren bemüht werden, um das Optimum zu bestimmen. Die Komplexität dieser Verfahren steigt allerdings stark mit der Problemgröße, wodurch es schon für moderate Problemgrößen unmöglich wird, die optimale Produktlinie zu finden (vgl. Green/Krieger 1985, S. 1). Aus diesem Grund wurden diverse, hochspezialisierte Lösungsalgorithmen, sog. Heuristiken, für die Produktlinienoptimierung entwickelt, um gute Näherungen des Optimums erzielen zu können. Genannt seien hier z.B. die Greedy-Heuristik und Interchange-Heuristik (vgl. Green/Krieger 1985, S 8/9) sowie die Dynamic-Programming-Heuristik (vgl. Kohli/Krishnamurti 1987, S. 1526). Im Jahr 1996 gelang es Balakrishnan/Jacob

(1996, S. 1112) für das Share-of-Choice-Problem und das Buyers'-Problem zu zeigen, dass ein Verfahren aus dem Machine-Learning, der Genetische Algorithmus, der bis zu diesem Zeitpunkt besten Heuristik, der Dynamic-Programming-Heuristik überlegen ist. Damit begann der Einzug weiterer Verfahren aus dem Machine-Learning in die Produktlinienoptimierung. An diesem Punkt setzt die vorliegende Arbeit an.

Ein Problem, das sich aus den Veröffentlichungen, die Verfahren aus dem Bereich des Machine-Learnings miteinander zu vergleichen, ergibt, ist, dass zwar die Performance der Verfahren berichtet wird, häufig allerdings kein Verfahren sicher benannt werden kann, das die besten Zielfunktionswerte liefert. Hierfür fehlen statistische Tests, um ausschließen zu können, dass es sich bei der Überlegenheit einiger Verfahren ggü. anderen Verfahren nicht um zufällige Effekte handelt. Des Weiteren werden des häufig Operatoren aus Publikationen verwendet, die zwar bei einem anderen Optimierungsproblem, z.B. beim Problem des Handlungsreisenden angewendet wurden, es allerdings nicht sichergestellt ist, dass diese auch für das Produktlinienoptimierungsproblem geeignet sind. So werden z.B. für den Genetischen Algorithmus Sensitivitätsanalysen für die Höhe einzelner Parameter durchgeführt, es fehlt aber ein grundsätzlicher Vergleich, welche Operatoren für die Selektion, den Crossover oder die Mutation besser geeignet sein könnten. Ähnlich sieht es bei Simulated Annealing (Belloni et al. 2008b) aus, wo für die Simulationsstudie und den realen Datensatz händisch Cooling-Schedules gesucht und anschließend verwendet wurden, die gute Zielfunktionswerte finden. Aber eben nur für dieses eine spezifische Problem, ohne einen allgemeingültigeren Cooling-Schedule zu entwerfen. In diesem umfangreichsten und bedeutendsten Verfahrensvergleich fehlt ebenso ein statistischer Vergleich, um zufällige Effekte ausschließen zu können. Ähnliche Probleme zeigen sich in den Veröffentlichungen zur Ant-Colony-Optimization von Albritton/McMullen (2007) und der Particle-Swarm-Optimization von Tsafarakis et al. (2011), deren Implementierungen und Ergebnisse die Gleichwertigkeit zu etablierten Verfahren zeigen, was sich allerdings in umfangreicheren Simulationen als unhaltbar erweisen wird. In den meisten Publikationen zum vorliegenden Thema wird meist nur eine Zielfunktion betrachtet. Dies macht es nicht notwendigerweise möglich, auf die Performance eines Verfahrens auf andere Zielfunktionen der Produktlinienoptimierung zu schließen.

Ein weiteres, nicht produktlinienoptimierungsspezifisches, Problem ist die große Anzahl von Machine-Learning-Verfahren zur näherungsweisen Lösung von Kombinatorischen Optimierungsproblemen. In den vergangenen 20 Jahren gab es in diesem Gebiet eine beschleunigte Entwicklung, die nicht unbedingt positiv zu bewerten ist. So handelt es sich bei vielen neuen Verfahren nicht um neue Verfahren im eigentlichen Sinn. Es gibt kaum konzeptionell neue Lösungsverfahren, sondern hauptsächlich welche, die sich aus Elementen erfolgreicher Verfahren zusammensetzen und lediglich eine unterschiedliche Aneinanderreihung eingebettet in Metaphern, vor allem aus der Tierwelt und Physik, bieten (vgl. Soerensen 2013). Dementsprechend sind es nach wie vor die etablierten und gut untersuchten Verfahren, wie Genetische Algorith-

men (vgl. Holland 1975, Goldberg 1989), Simulated Annealing (vgl. Kirkpatrick et al. 1983), das Min-Max Ant-System (vgl. Stützle/Hoos 1997) und weitere ältere Verfahren, die für viele verschiedene Optimierungsprobleme geeignet sind.

1.2 Relevanz des Forschungsthemas

Die Relevanz des Forschungsthemas ergibt sich zum einen aus der Sicht der Marketingforschung, die durch das schwierige Auffinden von optimalen Produktlinien weiterhin auf selbstlernende Verfahren angewiesen ist, um optimale Produktlinien anzunähern. Hierfür ist das dezidierte Untersuchen, Weiterentwickeln und Vergleichen der infrage kommenden Verfahren unerlässlich. Die Marketingforschung profitiert dabei von Algorithmen, die in der Lage sind, in kurzer Zeit Entscheidungshilfen für z.B. die Produktoptimierung, die Produktneuentwicklung, Produktlinienerweiterungen oder gänzlich neuen Produktlinien mit bestimmten Zielen zur Verfügung zu stellen. Die Produktoptimierung ist bspw. mit 52,41 % aller Anwendungen der häufigste Anwendungsfall nach der Durchführung einer Conjointanalyse (Selka 2013, S. 72). Die Aktualität von Produktlinien und deren Ausgestaltung ist nach wie vor ein wichtiger Bestandteil der Marketingforschung (siehe z.B. Zhang et al. 2018, Bertsimas/Misic 2019, Shen et al. 2020, Tsafarakis et al. 2020, Zou et al. 2020).

Zum anderen zeigt sich die Relevanz des Forschungsthemas aus der Notwendigkeit der Einführung statistischer Stringenz bzgl. des Vergleichs der Machine-Learning-Verfahren im Sinne der Lösungsqualität, um valide Aussagen über die Überlegenheit eines Verfahrens über ein anderes treffen zu können. Diese Arbeit schließt diese Forschungslücke.

Des Weiteren können praxisrelevante Aspekte die Wichtigkeit der Produktlinienoptimierung hervorheben. So ist z.B. die Produktneuentwicklung der Schlüssel für die Profitabilität eines Unternehmens. Ein wichtiger Aspekt der Produktneuentwicklung ist die Produktlinie, deren ideale Zusammensetzung ein äußerst kompliziertes Problem ist (Hauser 2011, S. 26). Die optimale Zusammensetzung einer Produktlinie gehört zu den wichtigsten Problemen in der Marketingforschung (Green/Krieger 1989, S. 127; Kohli/Krishnamurti 1989, S. 186; Nair et al. 1995, S. 767; Balakrishnan/Jacob 1996, S. 1105; Chen/Hausman 2000, S. 327; Steiner/Hruschka 2003, S. 229; Schön 2010, S. 896). Damit Produktlinien optimal zusammengestellt werden können, bedarf es einer Analyse, ob Produkte zur Produktlinie hinzugefügt oder herausgenommen werden. Des Weiteren spielen bei dieser Entscheidung Wechselbeziehungen eine wichtige Rolle. Zu nennen sind hier bspw., ob sich Produkte gegenseitig ersetzen (Substitution) oder eine Ergänzung (Komplement) darstellen (Green/Krieger 1985, S. 1). Jedes Unternehmen, das eine Palette von Produkten anbietet, die aus für die Konsumenten bzw. Kunden relevanten verschiedenen Eigenschaften und Eigenschaftsausprägungen aufgebaut sind, ist von Produktlinienent-

scheidungen betroffen. Diese Tatsache zieht sich durch das gesamte Spektrum der produzierenden Wirtschaft, wie Lebensmittelindustrie, Automobilindustrie, Telekommunikation oder auch die Gesundheitswirtschaft (Balakrishnan et al. 2004, S. 1). Aufgrund des immer größer werdenden Drucks für Unternehmen durch Globalisierung, immer kürzer werdende Produktlebenszyklen und die schnelle Entwicklung im Technologiebereich hat die Wahl der optimalen Produktlinie vor allem in hochkompetitiven Märkten und Branchen großen Einfluss auf die Umsätze und die Marktanteile (Balakrishnan et al. 2004, S. 1; Tsafarakis et al. 2011, S. 13). Eine beeindruckende Zahl, die die Wichtigkeit der Durchführung einer Produktlinienoptimierung unterstreicht, ist, dass 65-79% aller Produktneueinführungen floppen (Hermann 2006, Tacke et al. 2014).

Weitere wichtige Aspekte sind die Anwendung und Weiterentwicklung der Machine-Learning-Verfahren auf das Produktlinienoptimierungsproblem. Hierdurch profitiert dieses allgemeinere Forschungsfeld durch das Untersuchen der Verfahren für vier weitere Zielfunktionen. Umso mehr Optimierungsprobleme mit derartigen selbstlernenden Verfahren untersucht werden, desto mehr lässt sich über deren Eignung und Funktionsweise lernen.

1.3 Zielstellung der Arbeit

Aus der bisherigen Forschung ergeben sich einige weitere Fragestellungen, die in dieser Arbeit geklärt werden sollen. Ein Aspekt ist eine Literaturübersicht der eingesetzten Machine-Learning-Verfahren seit 1995. Dieser Zeitpunkt wird so gewählt, da hier zum einen der Startpunkt für moderne Machine-Learning-Verfahren in der Produktlinienoptimierung durch die Einführung Genetischer Algorithmen liegt (Balakrishnan/Jacob 1995; 1996), und zum anderen eine Literaturübersicht von Baier/Gaul (1999, S. 372) endet, an welcher angesetzt werden soll.

Des Weiteren sollen diejenigen Verfahren miteinander verglichen werden, die bis zum Zeitpunkt dieser Arbeit in der Produktlinienoptimierung verwendet wurden und nicht bereits miteinander verglichen wurden. Dabei handelt es sich um Genetische Algorithmen (siehe z.B. Balakrishnan/Jacob (1995; 1996), Steiner/Hruschka (2003)), Ant-Colony-Optimization (Albritton/McMullen 2007), Particle-Swarm-Optimization (Tsafarakis et al. 2011) und Simulated Annealing (Belloni et al. 2008b). Diese Verfahren werden auf vier verschiedene Zielfunktionswerte angewendet, denen unterschiedliche Entscheidungsregeln zugrunde liegen. Dabei wird unterschieden in probabilistische und deterministische Gewinn- und Marktanteilsmaximierung. Ein weiteres Ziel ist es hierbei, adaptive und funktionierende Parameter der Verfahren zu identifizieren, die unabhängig von der Problemgröße und der Zielfunktion gute, approximative Zielfunktionswerte liefern. Auf diesem Gebiet besteht eine Forschungslücke, da die Parameter in den

bisherigen Veröffentlichungen entweder händisch und problemspezifisch durch die Forscher festgelegt wurden oder aufwändige Sensitivitätsanalysen zum Einstellen der Parameter – lediglich für bestimmte Probleminstanzen – durchgeführt wurden. Die verwendeten Parameter jedes Verfahrens werden des Weiteren in eigenständigen Literaturtabellen aufgeführt. Die Überlegenheit der adaptiven Parametereinstellung wird demonstriert.

Die Verbesserung der Performance der bisher verwendeten Verfahren steht ebenso im Fokus. So wird in einer Monte-Carlo-Simulation untersucht, welche Operatoren beim Genetischen Algorithmus für jede Zielfunktion die viel versprechendsten für die Produktlinienoptimierung sind. Weiterhin wird überprüft, ob Particle-Swarm-Optimization für die Produktlinienoptimierung grundsätzlich geeignet ist. Für die Ant-Colony-Optimization wird gezeigt, dass die Implementierung von Albritton/McMullen (2007), vor allem für mittlere und größere Probleminstanzen, ineffizient ist und durch das MIN-MAX Ant-System von Stützle/Hoos (1997) ersetzt werden sollte. Für Simulated Annealing wird ein adaptiver Cooling-Schedule implementiert, der sich aus der statistischen Mechanik herleitet (Aarts/Van Laarhoven 1985, Aarts/Korst 1989).

In der Literatur zu anderen Kombinatorischen Optimierungsproblemen hat sich gezeigt, dass es sich vorteilhaft auf die Performance der Algorithmen auswirken kann, wenn eine lokale Suche in den Lernprozess integriert wird. Es wird gezeigt, dass eine lokale Suche beim MIN-MAX Ant-System zu besseren Zielfunktionswerten führt. Außerdem wird versucht über die Ausnutzung der Datenstruktur der individuellen Teilnutzenwerte der Konsumenten durch ein Clusteringverfahren im Genetischen Algorithmus die Zielfunktionswerte zu verbessern.

Nach der Verbesserung der einzelnen Verfahren werden diese einer Monte-Carlo-Simulation mit simulierten Datensätzen zugeführt, um herauszufinden, welche Verfahren sich besonders für die Produktlinienoptimierung eignen. Mit „besonderer Eignung“ ist die Höhe der Zielfunktionswerte gemeint. Da es sich ausschließlich um Maximierungsmodelle handelt, bedeutet ein höherer Zielfunktionswert einen besseren Zielfunktionswert. Generell ist es schwierig feste Rahmenbedingungen für die Verfahren vorzugeben, so dass ein Vergleich zwischen den Verfahren in jeglicher Hinsicht unter gleichen Voraussetzungen stattfinden kann. Dies ist der Tatsache geschuldet, dass jedes Verfahren grundsätzlich eine andere Funktionsweise besitzt. Setzt man z.B. eine feste Laufzeit für jedes Verfahren, wird diese für einige Verfahren bedeuten, dass sie bereits vor Laufzeitende gegen ein lokales Optimum konvergieren. Andere Verfahren hingegen haben ggf. noch nicht konvergiert. Zudem ist die Laufzeit der Verfahren nicht nur von den Verfahren selbst abhängig, sondern auch von Implementierungsdetails (z.B. binäre vs. multinomiale Modellierung der Entscheidungsvariablen) und der verwendeten Programmiersprache. Eine andere Idee wäre, die Verfahren über eine vorher festgelegte Anzahl von Zielfunktionswertevaluationen zu vergleichen und am Ende zu beurteilen, welches Verfahren die höheren Zielfunktionswerte liefert. Hierbei kommt es zu ähnlichen Problemen der unterschiedlichen Konvergenzgeschwindigkeiten der Verfahren. Aus diesen Gründen wird die bessere Performance der Verfahren nur

über die Qualität, sprich, die Höhe des Zielfunktionswerts beurteilt. Aus praktischen Gründen wird in der Monte-Carlo-Simulation zwar die Laufzeit auf ein Maximum begrenzt, aber nicht um eine bessere Vergleichbarkeit erzielen zu können, sondern um einen Vergleich erst möglich zu machen. In diesem Sinne bedeutet, „ein Verfahren ist besser als ein anderes“ in diesem Kontext, dass mit einem Verfahren innerhalb der festgelegten Parameter bessere Zielfunktionswerte erzielt wurden.

Zur Beurteilung der gemittelten Zielfunktionswerte werden statistische Tests paarweise durchgeführt und somit eine weitere Forschungslücke geschlossen. In der Literatur seit 1995 fehlen diese, so dass eine Bewertung der Eignung von bestimmten Verfahren schwerfällt, wenn diese mit der Höhe der Zielfunktionswerte nah beieinanderliegen.

Eine weitere Neuerung ist die Entwicklung eines Algorithmus, der die probabilistische Marktanteilsmaximierung in sehr kurzer Laufzeit, auch für größere Probleminstanzen, exakt lösen kann. Ein Beweis der Optimalität dieses Algorithmus wird gegeben. Hierdurch kann bestimmt werden wie nah (in Prozent) die durch die Verfahren erzeugten Lösungen dem tatsächlichen Optimum kommen.

Schlussendlich wird die Robustheit der hierarchischen Bayes-Schätzung bzgl. Messfehlern mittels der Verfahren untersucht. Dies führten schon Belloni et al. (2008b) und Camm et al. (2006) bereits durch, jedoch nicht für alle in dieser Arbeit verwendeten Zielfunktionen. In der vorliegenden Arbeit wird die Robustheit der Verfahren durch einen Datensatz einer realen Conjointstudie über ihre durch eine hierarchische Bayes-Schätzung geschätzten Teilnutzenwerte untersucht. Dabei werden die Ziehungen aus der zugrunde liegenden Markov-Kette verwendet.

Aus diesen Ausführungen ergeben sich die folgenden Forschungsfragen:

1. Lässt sich das probabilistische Marktanteilsmaximierungsproblem durch einen Algorithmus in angemessener Zeit exakt lösen?

2. Welche Operatoren für den Genetischen Algorithmus liefern die besten Zielfunktionswerte?

3. Lassen sich bestimmte Parameter der Verfahren adaptiv auf die zugrunde liegenden Problemgrößen und Zielfunktionswerte anpassen, um dadurch bessere Zielfunktionswerte als ihre Pendants aus der Literatur zu erzeugen?

4. Welches angewendete Verfahren erzielt die besten mittleren Zielfunktionswerte im Sinne der Zielfunktionswertqualität?

5. Ist die hierarchische Bayes-Schätzung bezogen auf Messfehler robust für alle Zielfunktionen?

1.4 Aufbau der Arbeit

In diesem Abschnitt wird der weitere Aufbau der Arbeit beschrieben.

In **Kapitel 2** wird ein allgemeiner Überblick über die Produktliniengestaltung gegeben. Anschließend werden theoretische und normative Ansätze der Produktliniengestaltung unterschieden. Nach dieser kurzen Übersicht wird in den Hauptteil der Arbeit eingestiegen, indem die Produktlinienoptimierung auf Basis der Conjointanalyse vorgestellt wird. Als Grundlage für die verwendeten Zielfunktionen werden deterministische und probabilistische Entscheidungsregeln beschrieben und deren Vor- und Nachteile beleuchtet. Weiter werden die Optimierungsmodelle für die probabilistische sowie deterministische Gewinn- und Marktanteilsmaximierung erläutert. Diese Modelle bilden die Grundlage für die Untersuchung mittels der Machine-Learning-Verfahren.

Kapitel 3 legt die benötigten theoretischen Grundlagen der Kombinatorischen Optimierung. Folgend wird eine Auswahl von Verfahren des Machine-Learnings seit 1995 gegeben und kritisch diskutiert. Im Anschluss werden diejenigen Verfahren detailliert beschrieben, die in den späteren Simulationsrechnungen Verwendung finden, namentlich Genetische Algorithmen, Particle-Swarm-Optimzation, Ant-Colony-Optimization und das MIN-MAX Ant-System, sowie Simulated Annealing.

Das **4. Kapitel** beginnt mit einer Literaturanalyse der Produktlinienoptimierung seit 1995. Außerdem werden die Veröffentlichungen in diesem Bereich zu jedem Verfahren zusammengefasst und die wichtigsten Parameter aufgeführt. Bevor die Details der Implementierungen der Verfahren gegeben werden, wird der exakte Algorithmus zur Lösung des probabilistischen Marktanteilsmaximierungsmodells sowie der zugehörige Beweis präsentiert. In diesem Kapitel befinden sich die Monte-Carlo-Simulationen der einzelnen Verfahren. So werden 18 verschiedene Genetische Algorithmen mit unterschiedlichen Operatoren untersucht. Danach wird ein Clusteransatz präsentiert, der für die besten drei Genetischen Algorithmen die Performance weiter verbessert, und von diesen drei Algorithmen wird derjenige für den späteren Vergleich mit den anderen Verfahren ausgewählt, der unter den gegebenen Parametern die besten Zielfunktionswerte liefert. Im Folgenden werden die Implementierungsdetails der Particle-Swarm-Optimization und der Ant-Colony-Optimzation dargelegt. Zusätzlich wird das MIN-MAX Ant-System für die Produktlinienoptimierung eingeführt, mit einer lokalen Suche versehen und gegen die Ant-Colony-Optimization gebenchmarkt. Am Ende des Kapitels wird ein ad-

aptiver Cooling-Schedule für Simulated Annealing erarbeitet und mit dem bisherigen Cooling-Schedule verglichen.

Der große Vergleich der verschiedenen Verfahren wird in **Kapitel 5** durchgeführt. Hierzu werden die Parameter definiert und die Probleminstanzen aus einem Datensatz mit 2089 aufgelisteten Conjointstudien zwischen 1996 und 2011 mittels linearer Regression hergeleitet. Für den Vergleich werden sechs Verfahren herangezogen: ein Genetischer Algorithmus, ein Genetischer Algorithmus mit Clusteransatz, Particle-Swarm-Optimization, MIN-MAX Ant-System, MIN-MAX Ant-System mit lokaler Suche und Simulated Annealing.

Kapitel 6 untersucht anhand zweier realer Conjointdatensätze, ob die Verfahren robuste Produktlinien generieren, wenn über die Draws der Markov-Kette der hierarchischen Bayes-Schätzung sowie über die Punktschätzer optimiert wird.

Abschließend wird in **Kapitel 7** die Schlussbetrachtung formuliert und ein Ausblick auf weitere Forschungsfragen, die sich aus dieser Arbeit ergeben, gegeben.

Anmerkung: In der gesamten Arbeit werden Dezimalzahlen wie in der englischen Schreibweise mit einem Punkt von ihren Nachkommastellen getrennt. Der Grund liegt in der sonst schlechten Übersichtlichkeit in den Ergebnistabellen der Kapitel 4 und 5.

2 Produktliniengestaltung

Zu Beginn dieses Kapitels soll zunächst geklärt werden, worum es sich bei einer Produktlinie handelt. Eine auf Mussa/Rosen (1978) zurückgehende Definition beschreibt eine Produktlinie als Spektrum von Waren desselben Typs, das in der Qualität der einzelnen Waren Unterschiede aufweist (Mussa/Rosen 1978, S. 301). Bezugnehmend auf diese Definition sehen Aydin/Ryan (2000) in einer Produktlinie ein Bündel von Produkten, das im Prinzip die gleichen Kundenbedürfnisse adressiert (Aydin/Ryan 2000, S. 1). Diese Produkte unterscheiden sich hinsichtlich der Existenz oder Nichtexistenz bestimmter Eigenschaften. Diese Definition soll als Grundlage für diese Arbeit dienen, wobei gleiche Kundenbedürfnisse nicht mit einer homogenen Präferenzstruktur der Kunden verwechselt werden dürfen. Im Gegenteil, gerade weil die Kunden heterogene Präferenzen aufweisen, ist es von entscheidender Bedeutung die Produktlinie auszudifferenzieren und somit einer möglichst breiten Masse an Kunden gerecht zu werden. Eine Studie von Kekre/Srinivasan (1990) legt diesen Zusammenhang offen, in der die Produktpolitik von Fortune 500 Unternehmen analysiert und dabei festgestellt wurde, dass umso breiter eine Produktlinie ist, desto höher ist der Marktanteil im betreffenden Segment. Einzuwenden ist hierbei allerdings, dass die Unternehmen mit heterogenen Märkten konfrontiert sind. Das bedeutet, dass für jedes Käufersegment ein anderes ideales Produkt existiert. Ein anderer Grund, aus dem nicht immer alle Käufersegmente bedient werden können, ist, dass die Einführung neuer Produkte in die Produktlinie mit einer Erhöhung der fixen Kosten einhergeht.

Es wird folgende Begriffsvereinbarung vorgenommen. *Produktliniengestaltung* wird verwendet, wenn es sich explizit nicht um Optimierungsmodelle handelt, sondern um die allgemeine Beschreibung von Produktdesign. Im Gegensatz dazu wird von *Produktlinienoptimierung* gesprochen, wenn das Hauptaugenmerk auf der Optimierung von Produktlinien im mathematischen Sinne liegt. Die Konfusion der Begrifflichkeiten kommt durch die Tatsache zustande, dass in der englischsprachigen Literatur häufig von *product(-line) design* und *product-line optimization* gesprochen wird.

2.1 Theoretische und normative Ansätze

Ein zentrales Problem des Marketing ist, wie eine Produktlinie bzw. Produktlinien am Markt zu positionieren und welche Verkaufspreise diesen Produktlinien zuzuordnen sind, um den Gewinn zu maximieren (vgl. Dobsen/Kalish 1988; 1993). Mit der Preisfindung von Produktlinien befasste sich bereits Dean (1950) Anfang der 1950er Jahre. Unternehmen stehen vor dem Problem, Preise für Produkte zu bestimmen, die einander ähneln, sich jedoch in verschiedenen Punkten, wie Größe, Qualität, Aussehen oder Funktionen unterscheiden. Neue Produkte sollen somit optimal am Markt platziert werden. Optimal bedeutet in diesem Fall, dass die Ausprägungen der Produkteigenschaften so gewählt werden, dass die vom Unternehmen vorgegebenen Ziele erreicht werden (vgl. Albers 1979, S. 222). Hierzu zählen bspw. das Minimieren von Kosten sowie das Maximieren von Marktanteilen, Gewinnen oder Verkaufszahlen (vgl. Green et al. 1981, Gaul et al. 1995).

Um Produktlinienentscheidungen zu treffen, bedarf es nach Green/Krieger (1985) einer Analyse des Hinzufügens und des Entfernens von Produkten aus einer Produktlinie. Hierzu gehört ebenfalls das Untersuchen von Wechselbeziehungen zwischen den Produkten innerhalb der Produktlinie, wie bspw. Substitutionalität und Komplementarität. Weiterhin ist zu berücksichtigen, wie die eigene und die Produktlinie konkurrierender Unternehmen in Beziehung zueinander stehen. Typische Fragen, die sich bei Produktlinienentscheidungen für Unternehmen ergeben, sind z. B. wie viele verschiedene Geschmacksrichtungen soll ein Hersteller von Suppen anbieten. Sobald diese Entscheidung getroffen ist, muss entschieden werden, um welche Geschmacksrichtungen es sich dabei handeln soll. Oder welche Zusatzangebote, die zusätzliche Kosten für den Verbraucher verursachen, soll ein Elektronikhersteller für seine Kühlschränke anbieten? Welche Tarife sollte ein Kommunikationsunternehmen für seine Handy- oder Smartphoneverträge anbieten? Diese Fragen zeigen, dass Unternehmen für ihre Produktlinienentscheidungen u.a. folgende Dinge berücksichtigen müssen: die Größe und die Gestaltung der Produktlinie, die Gestaltung alternativer Produktzusammensetzungen und die Wahlmöglichkeit bei Zusatzangeboten für die verschiedenen Produkte.

In Anlehnung an Dobsen/Kalish (1988) wird eine Einteilung in die qualitativ orientierte und die quantitativ orientierte Produktliniengestaltung unternommen. Als eine der ersten Vertreter der qualitativen Produktliniengestaltung beschäftigten sich Murray/Wolfe (1970) mit dem Problem, aus wie vielen Produkten eine Produktlinie bestehen sollte. Die Breite einer Produktlinie hat Auswirkungen auf die Verkaufszahlen der einzelnen Produkte, die Produktionskosten und die Rendite der Unternehmen. Die Schwierigkeit bei der Anzahl der aufzunehmenden Produkte in eine Produktlinie liegt vor allem in der Komplexität des miteinander zu verknüpfenden Wissens. Hierzu zählen die Marktkenntnis, das Wissen über den Produktionsprozess, die Kosten für die Herstellung der Produkte sowie deren Vertrieb und die Charakteristika der alternativen Produk-

te. Das entwickelte Modell wird dazu genutzt, folgende Effekte zu untersuchen: Reduzierung einer Produktlinie durch das Entfernen von marginalen, zu speziellen und schlecht gehenden Produkten; Erweiterung der Produktlinie durch das Hinzufügen neuer Produkte; Ersetzen von Produkten einer existierenden Produktlinie sowie die Kombination aller drei vorhergehenden Möglichkeiten.

Weitere typische Vertreter des qualitativ orientierten Ansatzes sind Wind/Claycamp (1976), die in der Entwicklung eines Strategieplans für die existierende Produktlinie eines Unternehmens das kritischste Element bei der Planung der Marketingaktivitäten des Unternehmens sehen. Zum Erstellen solcher Pläne benötigt das Unternehmen möglichst genaue Informationen über die aktuelle Akzeptanz seiner Produkte am Markt. Diese Informationen sollten durch folgende zwei Punkte beschafft werden: zum einen durch die Evaluierung der Kundenakzeptanz bezüglich der eigenen Produkte, vor allem durch das Ausfindigmachen der Stärken und Schwächen seiner eigenen Produkte im Vergleich zur Konkurrenz. Dies könnte z. B. durch Produktpositionierung durch Marktsegmentierung erreicht werden. Zum anderen sollten objektive Informationen über wirkliche und erwartete Produktperformances von relevanten Kriterien wie Verkaufszahlen, Gewinne und Marktanteile herangezogen werden. Die Autoren entwickelten einen Ansatz zur Planung von Produktlinien basierend auf den Verkaufszahlen, dem Marktanteil und der Profitabilität mithilfe einer Produktevaluationsmatrix.

Als Übersicht werden im Folgenden die zur qualitativ orientierten Produktliniengestaltung zählenden Vorgehensweisen genannt und kurz beschrieben:

- Das Strecken,
- das Ausfüllen,
- die Modernisierung und
- das Bereinigen

einer Produktlinie sind qualitative Merkmale einer Produktlinie (vgl. Kotler/Bliemel 1999, Kotler/Armstrong 2010). Im Allgemeinen gilt, dass der Umfang einer Produktlinie zu klein ist, wenn der Gewinn durch Hinzunahme neuer Artikel gesteigert werden kann. Der Umfang ist hingegen zu groß, wenn der Gewinn durch Eliminierung von Artikeln aus der Produktlinie erhöht werden kann.

Das Strecken einer Produktlinie bedeutet, dass Unternehmen ihre Produktlinie in andere Bereiche des Marktes ausdehnen. Möglich sind hierbei drei Szenarien: das Abwärts- und Aufwärtsstrecken, sowie das Strecken der Produktlinie in beide Richtungen. Von Abwärtsstrecken wird gesprochen, wenn Unternehmen ihre Produktlinie nach unten erweitern. Meist werden dabei niedrigklassigere Preissegmente adressiert, um den Umsatz zu erhöhen oder Kunden an die eigene Marke heranzuführen, damit diese später die höherpreisigen Produkte kaufen. Das

Aufwärtsstrecken bezeichnet das genaue Gegenteil. Unternehmen, die bisher nierdrigpreisigere Segmente bedienen, strecken ihr Produktangebot nach oben, um bspw. von höheren Gewinnmargen zu profitieren oder ebenfalls den Umsatz zu steigern. Unternimmt ein Anbieter einen Vorstoß in beide Richtungen, so spricht man von zweiseitigem Strecken, wovon er sich die Vorteile beider Möglichkeiten verspricht (vgl. Kotler/Bliemel 1999).

Wird eine Produktlinie durch Hinzunahme neuer Artikel innerhalb des schon vorhandenen Produktspektrums vergrößert, so liegt das Ausfüllen einer Produktlinie vor. Gründe hierfür können in der Nachfragebefriedigung von Händlern oder Konsumenten liegen oder auch, um Marktlücken zu schließen, in die die Konkurrenz dann nicht mehr vordringen kann (vgl. Kotler/Bliemel 1999).

Sowohl beim Strecken als auch beim Ausfüllen einer Produktlinie besteht die Gefahr, dass der sog. Kannibalisierungseffekt auftritt. Zum einen können die unterschiedlichen Bedürfnisse der Kunden durch Einführung neuer Produkte in die einzelnen Kundensegmente gedeckt werden, aber zum anderen kann dies dazu führen, dass durch die Fixkosten, die beim Hinzufügen eines Produktes in die Produktlinie anfallen, und den kontraproduktiven Effekt, dass die Nachfrage nach anderen Produkten der Produktlinie sinkt, unter Umständen den Gewinn des Unternehmens zurückgehen lassen können (vgl. Dobsen/Kalish 1988). Green/Krieger (1987) erarbeiteten einen Ansatz zur optimalen Produktlinienerweiterung, der dem Problem der Kannibalisierung von bestehenden Produkten der eigenen Produktlinie gerecht wird und die negativen Auswirkungen des Kannibalisierungseffekts auf vereinzelte Produkte minimiert.

Im Laufe der Zeit kommt der Punkt, an dem Produktlinien erneuert werden müssen. Eine Modernisierung wird z. B. dann nötig, wenn die Funktionen oder das Erscheinungsbild der Produkte in der Produktlinie veraltet sind. Je nach Bedarf kann die Produktlinie ganz oder nur teilweise erneuert werden. Ist eine Modernisierung einzelner Produkte aus der Produktlinie nicht mehr möglich oder vermindern bestimmte Produkte gar den Gewinn der gesamten Produktlinie, so ist das Unternehmen gezwungen, eine Bereinigung der Produktlinie durch Entfernen einzelner, unprofitabler Artikel durchzuführen (vgl. Kotler/Armstrong 2010).

Die Vorgehensweisen Stecken und Ausfüllen einer Produktlinie bedürfen einer kurzen, kritischen Betrachtung. Zum einen wird bei diesem Vorgehen unterstellt, dass es für die Qualität der Produkte innerhalb einer Produktlinie eine objektive Rangfolge gibt, die sogar noch auf individueller Kundenebene allgemeingültig besitzen müsste. Die subjektive Wahrnehmung des einzelnen Kunden über die Einschätzung der Qualität der Produkte wird so nur ungenügend Rechnung getragen. Zum anderen ist es oft überhaupt nicht möglich, Qualitätsunterschiede in den meisten Produktklassen zu identifizieren. Häufig unterscheiden sich die Produkte lediglich durch viele verschiedene Eigenschaftsausprägungskombinationen, die i. d. R. alle ein einheitliches Qualitätsniveau aufweisen. Das Problem der Produktliniengestaltung kann somit in den

meisten Fällen nur unbefriedigend gelöst werden, weshalb eine Entscheidungsunterstützung für das Management eines Unternehmens mittels qualitativer Methoden als eher fraglich anzusehen ist (vgl. Aust 1995).

Die vorliegende Arbeit wird sich ausschließlich mit quantitativ orientierten Ansätzen der Produktliniengestaltung beschäftigen. Gekennzeichnet sind diese vor allem durch die formale Festlegung des zugrunde liegenden Optimierungsproblems. In Anlehnung an Aust (1995) wird die Unterteilung in theoretische und normative Ansätze vorgenommen.

2.1.1 Theoretische Ansätze

Die theoretischen Ansätze wurden früh, vor allem durch die Arbeit von Hotelling (1929), geprägt. Hotelling (1929) beschäftigte sich mit der Stabilität von Angebot und Nachfrage in Märkten, die von mehreren, konkurrierenden Marktteilnehmern bestimmt sind. Er befasste sich mit der Frage, unter welchen Voraussetzungen ein Gleichgewicht am Markt existieren kann. Einem Duopol liegt bspw. eine inhärente Instabilität zugrunde, hervorgerufen durch die Tatsache, dass Verkäufer von Produkten den Preis festlegen und die Käufer dieser Produkte die Menge, die sie abnehmen, dementsprechend anpassen müssen.

Auf Grundlage dieser Arbeit modifizierte Moorthy (1993) die Voraussetzungen leicht. Als erste Voraussetzung wird postuliert, dass sich Produkte, mit Ausnahme des Preises, lediglich in einer Eigenschaft unterscheiden dürfen. Des Weiteren gliedert sich die Nachfrage in Segmente, welche bzgl. der optimalen Eigenschaft gleichverteilt über einem bestimmten Intervall angenommen wird. Für jedes Segment wird der Preis eines Produkts von der Nachfrageseite über die maximale Zahlungsbereitschaft definiert und nimmt quadratisch mit der Distanz zur optimalen Eigenschaftsausprägung der deskriptiven Eigenschaft ab. Vereinfachend wird weiterhin vorausgesetzt, dass keine Fixkosten für die zu produzierenden Produkte existieren und die variablen Kosten des Produkts von der Eigenschaftsausprägung der deskriptiven Eigenschaft nicht abhängig sind. In diesen Modellen wird die Differenz zwischen Nachfrage- und Angebotspreis, also die Konsumentenrente, ohne Beachtung stochastischer Variablen und für jedes einzelne Segment maximiert. Die Konkurrenzsituation wird durch ein Duopol beschrieben, in dem jeder der beiden Anbieter jeweils nur ein Produkt im Sortiment hat. Außerdem wird die Eigenschaftsausprägung der deskriptiven Eigenschaft für die Produkte simultan ausgewählt. Die Konkurrenten können nun nach dieser Festlegung lediglich noch den Preis variieren.

Legt man diese Annahmen zugrunde, so kommt es bei einer Gewinnmaximierung zu einem Marktanteil im Gleichgewicht von 50 %. Hierbei sind die Gewinne beider Konkurrenten gleich, vorausgesetzt die segmentspezifischen Nachfragepreise unterschreiten eine bestimmte Schranke nicht. Die auf Hotelling (1929) und Moorthy (1993) basierenden Modelle weisen die große

Schwäche auf, dass sie sehr restriktiv sind. So stellte Carpenter (1989) fest, dass die Aussagekraft der Modelle bereits dann eingeschränkt ist, wenn anstatt einer deskriptiven Eigenschaft, zwei Eigenschaften zur Gewinnmaximierung herangezogen werden. Es handelt sich bei diesen Annahmen immer um den Ein-Produkt-Fall, d.h., ganze Produktlinien werden hier nicht betrachtet.

Weitere theoretische Modelle (vgl. Mussa/Rosen 1978, Brander/Eaton 1984, Moorthy 1984; 1987; 1988; 1993) sollen hier nicht weiter erläutert werden, da der Fokus dieser Arbeit auf Modellen beruht, die die Conjointanalyse als Basis zur Simulation oder Optimierung für die Produktliniengestaltung verwenden.

2.1.2 Normative Ansätze

Einen ersten Ansatz zur normativen Produktliniengestaltung lieferte Urban (1969). Er stellte fest, dass die Wechselbeziehungen zwischen den Marken eines Unternehmens in einer Produktlinie bei der Marketingstrategie berücksichtigt werden sollten. Diesen Wechselbeziehungen kommt eine Schlüsselrolle bei der Auswahl der optimalen Produktzusammenstellung zu, da die Einführung neuer Produkte die Verkaufszahlen anderer Produkte aus der Produktlinie beeinflussen kann. Urban (1969) entwickelte ein mathematisches Modell, das Interaktionen zwischen verschiedenen Produkten für normative Strategieempfehlungen erfasst. In den 1960er Jahren gewannen die Produktlinien der Unternehmen an Breite, da sie sich zunehmend dem Wettbewerb mit anderen Firmen stellen mussten, um so weitere, ausdifferenziertere Marktsegmente adressieren zu können. Weiterhin stieg in diesem Zeitraum die Bedeutung der Unternehmen, die viele unterschiedliche Produkte in ihrem Sortiment hatten (vgl. Urban 1969).

Ein mit der normativen Produktliniengestaltung verwandtes Problem ist die optimale Produktpositionierung. Als eine der Grundlagen der normativen Produktliniengestaltung soll das Produktpositionierungsproblem kurz beschrieben werden. Durch die Anwendung der multidimensionalen Skalierung, auf der die normative Produktliniengestaltung u.a. beruht, bietet sich dies als einführendes Beispiel an. Optimale Produktpositionierung bedeutet, dass diejenigen Eigenschaftsausprägungen eines neu in einen Markt einzuführenden Produkts bestimmt werden müssen, die gewisse Zielvorgaben eines Unternehmens am besten erfüllen (vgl. Albers 1979, S. 222). Das Problem der Produktpositionierung würde nicht auftreten, wenn alle Kunden jede Produkteigenschaft rational bewerten würden. Hiermit ist z. B. gemeint, dass für bestimmte Eigenschaften gilt: je mehr (oder weniger), desto besser. Ein Beispiel hierfür ist, umso fruchtiger ein Saft, desto besser ist das Produkt. In diesem Fall wäre das Vorgehen bei der Produktentwicklung für das Unternehmen offensichtlich. Es sollte diejenigen Produkte neu entwickeln, die eine möglichst große Menge der einen Eigenschaft aufweisen, ohne dabei auf die anderen Eigenschaften zu verzichten. Das Produktpositionierungsproblem tritt dort auf, wo Kunden

ideale Eigenschaftsausprägungen bevorzugen (vgl. Lehmann 1971). Dies gilt z. B. für die Süße von Wein, man kann hier nicht uneingeschränkt feststellen, dass je süßer (oder saurer) ein Wein, desto besser ist er. Sowohl positive als auch negative Abweichungen vom Idealen werden als weniger favorisierbar eingeschätzt (vgl. Albers 1979, S. 222). Für derartige Probleme wäre man bestrebt Nischen in Märkten zu finden, in denen man das Konzept der idealen Menge einer oder mehrerer Eigenschaftsausprägungen anwenden kann. Selbst in der Politikwissenschaft lässt sich dieses Konzept wiederfinden. Speziell bei der Aufstellung eines optimalen Kandidaten für Wahlen. Diese Kandidaten müssen so positioniert werden, dass sie mit ihren persönlichen Eigenschaften den Großteil der Wähler anziehen (vgl. Albers 1979, S. 222). Zur Lösung des Produktpositionierungsproblems benutzen Albers/Brockhoff (1977) Kundenpräferenzen auf Basis der mehrdimensionalen Skalierung (vgl. Green/Rao 1972) sowie ihren Lösungsalgorithmus für nichtlineare gemischt-ganzzahlige Optimierungsprobleme, PROPOSAS.

Die normativen Ansätze der Produktliniengestaltung werden in der Literatur meist durch die Einteilung in multidimensional skalierte und conjointanalytische Verfahren unterschieden (vgl. z. B. Green/Krieger 1989, Aust 1995, Kaul/Rao 1995, Baier/Gaul 1999). Aust (1995) kritisiert bei dieser Gliederung zurecht die Probleme, die mit dieser Konzeption einhergehen. Zum einen lassen sich die Einteilungen in multidimensional-skalierte und conjointanalytische Verfahren nicht immer eindeutig klassifizieren, da es allgemeine Ansätze zur normativen Produktliniengestaltung gibt, die sich keiner dieser Verfahren zur Präferenzmodellierung bedienen und überdies existiert eine Reihe von Ansätzen, die sowohl die Vorteile der multidimensionalen Skalierung als auch die der Conjointanalyse verbinden (für Beispiele vgl. Aust 1995, S. 39). Zum anderen weist der Autor darauf hin, dass diese Gliederung vor allem deshalb problematisch sei, da es sich bei diesen beiden Verfahren nicht um unterschiedliche Verfahren im eigentlichen Sinne handelt, sondern die Conjointanalyse von ihrer Methodik her vielmehr einen Sonderfall der multidimensionalen Skalierung darstellt (vgl. Böckenholt 1989).

Shocker/Srinivasan (1974) entwickelten ein analytisches Modell, das die Neigungen der Konsumenten zum Kaufen verschiedener Marken mit einem Suchprozess zur Identifikation optimaler neuer Produktideen verbindet. Im Marketing wird meist davon ausgegangen, dass es Produkte und Dienstleistungen gibt, die die Bedürfnisse der Kunden befriedigen. Das erfolgreichere Unternehmen wird dasjenige sein, welches Produkte mit überlegenen Eigenschaften anbietet. In der Praxis der Produktentwicklung sieht es jedoch so aus, dass die meisten Unternehmen üblicherweise nicht damit beginnen, ihre Kunden zu analysieren oder herauszufinden, welche existierenden Produkte ihren Bedürfnissen am ehesten entsprechen (vgl. Shocker/Srinivasan 1974, S. 921 ff.). Normalerweise kommt den Bedürfnissen der Kunden erst zu einem späteren Zeitpunkt im Produktentwicklungsprozess eine Bedeutung zu, wenn abzusehen ist, wie sie durch ihre Meinung oder ihr Verhalten am Markt auf die eingeführten Produkte reagieren (vgl. Day 1968, S. 24). Deshalb stellten Shocker/Srinivasan (1979) in ihrer Arbeit ein fünfschrittiges Ablaufschema zur normativen Produktliniengestaltung vor. Mithilfe dieses Schemas arbeitete

Aust (1995, S. 39 ff.) die wesentlichen Unterschiede zwischen multidimensional skalierten und conjointanalytischen Verfahren heraus.

Die zentralen Fragen aus Sicht der normativen Ansätze gestalten sich wie folgt (Green et al. 1981, S. 17):

- Welches neue oder modifizierte Produkt ist unter Berücksichtigung der aktuellen Produktlinie nach der Herstellung am profitabelsten und in welchem Markt sollte das neue oder modifizierte Produkt am besten positioniert werden?

- Welche Marktsegmente sind diejenigen, die den Gewinn der aktuellen Produktlinie maximieren und welches Produkt ist das profitabelste für ein vorher definiertes Segment?

- Welche Strategie ist die beste, um auf die Einführung eines Konkurrenzproduktes vom Produkt- bzw. Marktstandpunkt zu reagieren?

2.2 Produktlinienoptimierung auf Basis der Conjointanalyse

Bereits in den 70er Jahren des 20. Jahrhunderts, kurz nach der Einführung der Conjointanalyse in das Marketing (vgl. Green/Rao 1969), stammend aus der mathematischen Psychologie (vgl. Luce/Tuckey 1964), entstanden erste Simulations- und Optimierungsmodelle, die sich dieser, zur damaligen Zeit, neuen Methode bedienten (vgl. z. B. Shugan/Balachandran 1977, Zufryden 1977). Die Produktliniengestaltung avancierte nach und nach zum wichtigsten Anwendungsbereich der Conjointanalyse (vgl. Cattin/Wittink 1982, Wittink/Cattin 1989, Wittink et al. 1994).

Aufgrund der hervorragenden Literaturübersichten im Bereich der Produktliniengestaltung und -optimierung von Aust (1995), Kaul/Rao (1995) und Baier/Gaul (1999) werden in der vorliegenden Arbeit vor allem die Veröffentlichungen ab dem Jahr 1995 berücksichtigt. Des Weiteren werden in diesem Abschnitt nur Veröffentlichungen der Produktliniengestaltung betrachtet, die den Fokus nicht auf die Lösungsheuristiken für Produktlinienoptimierungsprobleme legen. Diese werden in Abschnitt 4.1 im Detail vorgestellt.

In der Arbeit von Kaul/Rao (1995) differenzieren die Autoren zunächst das Produktpositionierungsproblem sowie die Produktliniengestaltung und geben einen entsprechenden Überblick über die existierende Literatur (Kaul/Rao 1995, S. 297; S. 305; S. 313). Weiterhin entwickeln sie ein Modell, dass beide Gebiete zusammenführt und somit die Produktpositionierung und

die Produktgestaltung unter dem Gesichtspunkt der Gewinnmaximierung gemeinsam betrachtet (vgl. Kaul/Rao 1995, S. 303).

Eine Einteilung der Produktlinienoptimierung mittels Conjointanalyse in Simulations- und Optimierungsansätzen, wobei die Optimierungsansätze weiter in simultane und sequentielle Ansätze unterteilt werden, kann aus Aust (1995) entnommen werden. Bei den Simulationsansätzen werden aus den Teilnutzenwerten der einzelnen Eigenschaftsausprägungen, die mithilfe der Conjointanalyse gewonnen wurden, verschiedene Marktsituationen nachgeahmt. Ein Unternehmen möchte bspw. ein neues Produkt einführen, das es so vorher noch nicht auf dem Markt gab. Der Gesamtnutzen des neu einzuführenden Produkts ergibt sich aus den Teilnutzen seiner einzelnen Bestandteile. Es kann nun berechnet werden, welcher Marktanteil, Gewinn oder Absatz für das neue Produkt zu erwarten ist, da die Status-quo-Produkte der Wettbewerber mit in die Simulation einbezogen werden können. Mithilfe der Simulationsrechnungen können also die Anbieterseite mit den Produkten und ihren Eigenschaften bzw. Eigenschaftsausprägungen sowie die Nachfrageseite abgebildet werden.

Die Optimierungsansätze ermöglichen eine Erweiterung der Simulationsansätze. Wenn ein Unternehmen nicht nur ein, sondern mehrere Produkte neu einführen möchte bzw. bereits bestehende Produkte verbessern möchte, so kann es einzelne Produkte oder ganze Produktlinien anhand bestimmter Kriterien optimieren. Diese Kriterien können wiederum verschiedener Art sein, je nach dem worauf es dem Unternehmen zu diesem Zeitpunkt ankommt. Angestrebt werden meist Gewinn- oder Marktanteilsmaximierung. Das Ziel kann es aber auch sein, Kosten zu minimieren. Diese Optimierungsansätze gehören zu den kombinatorischen Optimierungsproblemen. Sie sind daher sehr rechenintensiv, weshalb es nicht immer möglich ist, alle Produktkombinationen miteinander zu vergleichen. Deshalb wurden verschiedene Heuristiken entwickelt, um die optimalen Lösungen der Optimierungsprobleme näherungsweise zu bestimmen. Auf einige dieser Heuristiken wird im Verlauf dieser Arbeit genauer eingegangen.

Aufgrund der größeren Flexibilität und der Möglichkeit der Betrachtung von komplexen Marktsituationen werden im Weiteren nur Optimierungsansätze näher erläutert. Sie ermöglichen nicht nur die Bestimmung optimaler Produktlinienerweiterungen, sondern auch die Bewertung beliebiger Produktlinien (vgl. Aust 1995, S. 84). In Abhängigkeit davon, ob die Produktprofile in den Optimierungsdurchläufen nacheinander oder parallel bestimmt werden, unterscheidet man in simultane und sequentielle Optimierungsansätze (vgl. Green/Krieger 1989). Wegen der großen Menge an verschiedenen Produktkombinationen und der noch größeren Menge aus daraus resultierenden Produktlinien, kann man die simultanen Optimierungsansätze noch nach partiellen oder totalen Ansätzen unterscheiden. Partielle Ansätze betrachten lediglich eine Teilmenge der möglichen Produktkombinationen, wohingegen die totalen Ansätze alle Möglichkeiten berücksichtigen, was zu deutlich besseren Zielfunktionswerten führen kann (vgl. Aust 1995, S. 84).

Baier/Gaul (1999) geben in ihrem Artikel einen tabellarischen Überblick über die bis 1999 erschienenen Veröffentlichungen im Bereich der Produktlinienoptimierung auf Basis der multidimensionalen Skalierung und der Conjointanalyse. Die Autoren präsentieren außerdem einen neuen Ansatz, mit dessen Hilfe Produkte optimal auf einem Markt positioniert werden können. Als Grundlage dienen Daten, die aus Paarvergleichen stammen. In Baier/Gaul (1999) werden zwar keine neuen Heuristiken zur Lösung des Produktlinienoptimierungsproblems entwickelt, jedoch stellt die Arbeit wegen der umfassenden Literaturübersicht einen wichtigen Beitrag zu diesem Thema dar. Nichtsdestoweniger konnte zusätzlich durch eine Monte-Carlo-Simulation und ein Anwendungsbeispiel aus dem Kaffeemarkt gezeigt werden, dass Idealpunktmodelle in Kombination mit Produktlinienoptimierungsmodellen sehr hilfreich sein können (vgl. Baier/-Gaul 1999, S. 386).

In den folgenden Betrachtungen werden die Grundlagen für die Optimierungsmodelle gelegt. Die Bezeichnungsweise der Variablen und Parameter wird von Gaul/Baier (2009) übernommen. Zur Erhöhung der Übersichtlichkeit werden die genannten Variablenvereinbarungen für den weiteren Verlauf auf einen Blick dargestellt:

I	Menge der Konsumenten/Segmente mit $i = 1,2,...,I$,
K	Menge der Eigenschaften eines Produkts mit $k = 1,2,...,K$,
L_k	Menge der Eigenschaftsausprägungen mit $l = 1,2,...,L_k, k = 1,2,...,K$,
J	Menge der aller zulässigen Produkte mit $j = 1,2,...,J$,
R	Menge der in die Produktlinie aufzunehmenden Produkte mit $j = 1,2,...,R$,
E	Menge der unternehmenseigenen Produkte auf dem Status-quo-Markt mit $j = 1,2,...,E$,
F	Menge der Produkte anderer Anbieter auf dem Status-quo-Markt mit $j = 1,2,...,F$,
β_{ikl}	Teilnutzenwert des i-ten Konsumenten/Segment mit der l-ten Eigenschaftsausprägung der k-ten Eigenschaft,
u_{ij}	Nutzen eines Produkts j für ein Individuum/Segment i,
p_{ij}	Auswahlwahrscheinlichkeit des j-ten Produkts für das i-te Individuum/Segment,
d^r_{ikl}	relative Stückdeckungsbeiträge von Konsument i für die l-ten Eigenschaftsausprägung der k-ten Eigenschaft,
f_{kl}	Fixkosten der l-ten Eigenschaftsausprägung der k-ten Eigenschaft,
ω_i	Gewicht/Anzahl der Produkte, die ein Konsument i kauft,
x_{jkl}	binäre Entscheidungsvariable des j-ten Produkts der l-ten Eigenschaftsausprägung der k-ten Eigenschaft (= 1, wenn das j-te Produkt die l-te Eigenschaftsausprägung der k-ten Eigenschaft hat; = 0, sonst),
x_{ijkl}	binäre Entscheidungsvariable des i-ten Konsumenten für das j-te Produkt der l-ten Eigenschaftsausprägung der k-ten Eigenschaft (= 1, wenn der dem Konsumenten i zugeordnete Produktkandidat j die l-te Eigenschaftsausprägung der k-ten Eigenschaft hat; = 0, sonst),
x_{ij}	= 1, wenn ein Produkt j aus der Produktlinie dem Konsumenten i einen gleich hohen oder niedrigeren Nutzen stiftet als sein Status-quo-Produkt; = 0, sonst,

x_i $= 0$, wenn keines der Produkte in der Produktlinie dem Konsumenten i einen höheren Nutzen stiften als sein Status-quo-Produkt; $= 1$, sonst,

y_i $= 1$, wenn Konsument i ein Produkt aus der Produktlinie kauft; $= 0$, sonst.

Es sei angemerkt, dass die Mächtigkeit der Mengen hier dem letzten Wert dieser geordneten Mengen entspricht. Deshalb wird die Mächtigkeit $|\cdot|$ einer Menge im Folgenden mit den Großbuchstaben für die entsprechende gleichgesetzt (z. B.: $|J| \widehat{=} J$).

Produktlinienoptimierungsmodelle auf Basis der Conjointanalyse sind NP-hard. Dieser Umstand resultiert aus der Tatsache, dass es sich um ein kombinatorisches Optimierungsproblem, speziell ein binäres Optimierungsproblem, handelt (vgl. Kohli/Sukumar 1990, S. 1466). Angenommen man möchte lediglich ein einzelnes optimales Produkt berechnen. Dann gäbe es hierfür

$$J = \prod_{k=1}^{K} L_k$$

mögliche Produkte, also Lösungen des Optimierungsproblems, die für eine Optimierung infrage kämen. Ein willkürlich ausgesuchtes Produkt besitze bspw. zehn Eigenschaften mit jeweils acht Eigenschaftsausprägungen. Dies hätte zur Folge, dass bereits bei dieser Problemgröße insgesamt 1.073.741.824 Zielfunktionswerte berechnet werden müssten, um das gewinnmaximale Produkt zu identifizieren. Betrachtet man nun nicht nur einzelne Produkte, sondern ganze Produktlinien wird das Problem noch deutlicher. Sei R die Anzahl von Produkten, die einer Produktlinie hinzugefügt werden sollen. So wäre die zu berechnende Anzahl von Produktkombinationen:

$$\binom{J}{R} = \binom{\prod_{k=1}^{K} L_k}{R}.$$

Die Grundlage für die Produktlinienoptimierung auf Basis der Conjointanalyse bilden das additive Teilnutzenwertmodell, sowie die darauf aufbauenden Entscheidungsregeln, wann ein Konsument ein Produkt kauft bzw. wie groß die Kaufwahrscheinlichkeit eines Konsumenten für ein bestimmtes Produkt ist. Anschließend lassen sich Optimierungsmodelle erstellen, die bspw. Marktanteile und Gewinne maximieren können. Die Entscheidungsregeln und vier mögliche Optimierungsmodelle für Produktlinien, wobei in deterministische und probabilistische Optimierungsmodelle unterschieden wird, abhängig von der Entscheidungsregel, werden in den folgenden Abschnitten detailliert beschrieben.

2.2.1 Entscheidungsregeln

Zunächst sollen Vereinbarungen über Indizes getroffen werden, die im Folgenden zu jedem Zeitpunkt für dieselben Indexmengen gelten. Die Indexmenge I steht entweder für einzelne

Individuen oder für einzelne Segmente $i \in I, i = 1,2,...,I$. Mit $k \in K, k = 1,2,...,K$, werden die Eigenschaften eines Produkts bezeichnet. Jede Eigenschaft besitzt $l \in L_k, l = 1,2...,L_k; k = 1,2,...,K$ Eigenschaftsausprägungen. Nach erfolgreicher Durchführung der Conjointanalyse stehen dem Forscher die geschätzten Teilnutzenwerte, also die Präferenzen, des i-ten Individuums/Segments der k-ten Eigenschaft der l-ten Eigenschaftsausprägung zur Verfügung. Diese werden mit $\beta_{ikl}, i \in I, k \in K, l \in L_k$ bezeichnet.

Der Nutzen eines Produkts $j \in J, j = 1,2,...J$, für ein Individuum/Segment $i \in I$ wird mit $u_{ij}, i \in I, j \in J$, bezeichnet. Dieser ergibt sich aus dem Produkt der Teilnutzenwerte β_{ikl} mit der binären Variable x_{jkl} über die Summe aller Eigenschaften K und aller Eigenschaftsausprägungen L_k, wobei $x_{jkl} = 1$, wenn das Produkt $j \in J$ die Eigenschaftsausprägung $l \in L_k$ der Eigenschaft $k \in K$ besitzt. Ist dies nicht der Fall, so ist $x_{jkl} = 0$. Der Gesamtnutzen eines Produkts j für einen Konsumenten i ergibt sich zu:

$$u_{ij} = \sum_{k=1}^{K} \sum_{l=1}^{L_k} \beta_{ikl} x_{jkl} \ . \tag{2.1}$$

Hat man die aggregierten bzw. individuellen Nutzenwerte der Produkte u_{ij} berechnet, lassen sich daraus die Auswahlwahrscheinlichkeiten der interessierenden Produkte errechnen. Hierzu existieren grundsätzlich zwei Konzepte: deterministische und probabilistische Entscheidungsregeln (vgl. Green/Krieger 1988; 1992, S. 121), auf die folgend eingegangen wird.

2.2.1.1 Deterministische Entscheidungsregel

Die First-Choice-Regel ist eine deterministische Auswahlregel. Der potentielle Kunde wählt diejenige Produktalternative, die ihm den höchsten Nutzen bringt, da zugrunde gelegt wird, dass sich die Kunden nutzenmaximierend verhalten.

$$p_{ij} = \begin{cases} 1, & \text{wenn } u_{ij} \geq u_{ij'} \qquad \forall j', \\ 0, & \text{sonst.} \end{cases} \tag{2.2}$$

Mit p_{ij} wird die Auswahlwahrscheinlichkeit eines Individuums/Segments $i \in I$ verstanden, ein bestimmtes Produkt $j \in J$ auszuwählen. Sollte der Fall eintreten, dass es mehr als ein Produkt mit derselben Auswahlwahrscheinlichkeit gibt, wird mit Hilfe einer Gleichverteilung eines der Produkte selektiert.

2.2.1.2 Probabilistische Entscheidungsregeln

Für die probabilistischen Auswahlregeln gibt es zwei gängige Alternativen. Zum einen die Bradley-Terry-Luce-Auswahlregel (BTL)

$$p_{ij} = \frac{u_{ij}^{\alpha}}{\sum_{j'=1}^{J} u_{ij'}^{\alpha}}, \quad \alpha \geq 0, \tag{2.3}$$

und zum anderen die Logit-Auswahlregel

$$p_{ij} = \frac{\exp(\alpha u_{ij})}{\sum_{j'=1}^{J} \exp(\alpha u_{ij'})}, \quad \alpha \geq 0. \tag{2.4}$$

Die Auswahlwahrscheinlichkeit eines Produkts $j \in J$ für ein Individuum/Segment $i \in I$ ergibt sich also als Quotient aus dem Nutzenwert des betreffenden Produkts und der Summe über alle Produkte für den entsprechenden Probanden. Bei der Logit-Auswahlregel werden die Nutzenwerte zusätzlich exponenziert. Mit Hilfe des Parameters α, das typischerweise einen Wert von mindestens $\alpha = 1$ besitzt (vgl. Green/Krieger 1992, S. 121), lassen sich vom Marktforscher gewonnene Information über die Beschaffenheit des Marktes in die Auswahlwahrscheinlichkeiten integrieren, wodurch eine höhere Realitätsnähe des Modells erreicht werden kann. Weiterhin erhalten diese beiden probabilistischen Auswahlregeln die First-Choice-Auswahlregel als Spezialfall. Betrachtet man den Grenzwert der Quotienten, wenn $\alpha \to \infty$ läuft, so wird dasjenige Produkt mit dem höchsten Nutzenwert ausgewählt. Setzt man hingegen $\alpha = 0$, so bekommt man eine Gleichverteilung der Auswahlwahrscheinlichkeiten, die von den Nutzenwerten nicht mehr abhängig sind.

Aus diesen Entscheidungsregeln ergeben sich verschiedene Vor- und Nachteile. Der First-Choice-Regel liegt die Annahme zugrunde, dass Konsumenten immer das Produkt mit dem höchsten Nutzen wählen. Das Problem bei dieser Annahme ist, dass versucht wird, aus geschätzten Teilnutzenwerten, die einen statistischen Fehler besitzen, weil sie aus stochastischen Modellen resultieren (z. B. Lineare Regression, Maximum-Likelihood-Schätzung und Hierarchische Bayes-Schätzung), jedoch mit einer deterministischen Entscheidungsregel versucht wird, das Auswahlverhalten der Konsumenten vorherzusagen (vgl. Louviere 1988, S. 97). Ein weiteres Problem bei diesem Vorgehen ist, dass die Konsumenten auf Basis des linear-additiven Modells immer transitive und konsistente Auswahlentscheidungen treffen, sowie vollständige Informationen über die Produkteigenschaften und deren Ausprägungen und über alle verfügbaren Pro-

duktalternativen besitzen (vgl. Louviere 1988, S. 96). Dies sind sehr strenge Annahmen, die sich in der Realität nicht vollständig widerspiegeln. Elrod/Kumar (1989, S. 264) untersuchen systematisch unter welchen Bedingungen die First-Choice-Regel zu einer Über- bzw. Unterschätzung der Marktanteile führt (z. B., wenn ein kleiner Fehler im Auswahlexperiment besteht, aber die Teilnutzenwerte nicht akkurat geschätzt wurden oder ein großer Fehler im Auswahlexperiment festgestellt wurde, aber die Teilnutzenwerte sehr akkurat gemessen wurden). Louviere (1988, S. 101) argumentiert, dass der Einsatz von probabilistischen Entscheidungsregeln aus diesen Gründen bestimmte Vorteile ggü. der First-Choice-Regel besitzen, die allerdings ebenfalls einen großen Nachteil mit sich bringen, der folgend kurz beschrieben wird.

Basierend auf dem Choice-Behavior-Axiom von Luce (1959), das dem Auswahlverhalten der Konsumenten in der Choice-based Conjointanalyse zugrunde liegt, zeigte Debreu (1960, S. 188) eine Eigenschaft von probabilistischen Entscheidungsregeln auf, die in der Literatur unter der Bezeichnung *independence of irrelevant alternatives (IIA)*[1] bekannt ist. Zur Beschreibung dieser Eigenschaft wird das häufig in der Literatur verwendete „roter Bus/blauer Bus Paradoxon“ bemüht. Angenommen eine Person entscheidet sich für die Durchführung einer Reise für den Bus oder für das Auto. Die Auswahlwahrscheinlichkeit beider Möglichkeiten läge bei jeweils $\frac{1}{2}$, bzw. in Wahrscheinlichkeiten $P(\cdot)$ ausgedrückt: $P('Bus') = P('Auto') = \frac{1}{2}$. Ändert man die Auswahlmöglichkeiten in einen blauen und einen roten Bus, sowie ein Auto, würde man folgende Wahrscheinlichkeiten erwarten: $P('roterBus') = P('blauerBus') = \frac{1}{4}$ und $P('Auto') = \frac{1}{2}$, wenn angenommen wird, dass die Farbe des Busses für die Person irrelevant ist. Probabilistische Auswahlregeln würden jedoch jeder Alternative eine Wahrscheinlichkeit von $P('roterBus') = P('blauerBus') = P('Auto') = \frac{1}{3}$. Diese Eigenschaft widerspricht somit der Intuition (vgl. Chen/Hausman 2000, S. 330). Auf der anderen Seite lassen sich durch den Parameter α flexibel vorhandene Marktinformationen berücksichtigen. Des Weiteren argumentieren Kaul/Rao (1995, S. 313), dass probabilistische Entscheidungsregeln einen realistischeren Ansatz zur Modellierung der Auswahlentscheidungen der Konsumenten bieten.

Da in der praktischen Anwendung alle drei Auswahlregeln Beachtung finden (vgl. Gaul/Baier 2009, S. 166; siehe Tabelle 4) und somit eine legitime Daseinsberechtigung besitzen, soll an dieser Stelle dem Manager bzw. dem Forscher die Wahl der passenden Entscheidungsregel überlassen werden, da domänenspezifisches Expertenwissen zur Beurteilung der Situation vorliegt. In dieser Arbeit werden für die Produktlinienoptimierungsmodelle daher sowohl probabilistische als auch deterministische Entscheidungsregeln berücksichtigt. Des Weiteren werden meist Marktanteile bzw. Share-of-Choices und der Gewinn einer Produktlinie maximiert (siehe Tabelle 4). Deshalb werden im Folgenden je ein Vertreter der Marktanteils- bzw. Share-of-Choices- und Gewinnmaximierung mit probabilistischen und deterministischen Entscheidungsregeln ausgewählt. Diese Optimierungsmodelle werden im späteren Verlauf in den

[1] Im Deutschen unter der Bezeichnung Unabhängigkeitsaxiom bekannt. Aufgrund der fast ausschließlich englischen Literatur im Bereich der Conjointanalyse wird hier der englische Begriff verwendet.

Simulationsrechnungen berücksichtigt.

2.2.2 Deterministische Modelle

In diesem Abschnitt werden ein Share-of-Choices- sowie ein Gewinnmaximierungsmodell vorgestellt, die in der Literatur die größte Verbreitung finden. Beide Modelle nutzen die First-Choice-Regel für die Modellierung der Auswahlentscheidungen der Konsumenten für die Produkte.

2.2.2.1 Deterministisches Marktanteilsmaximierungsmodell

Das deterministische Marktanteilsmaximierungsmodell [2] auf Basis der First-Choice-Regel wurde z. B. von Shocker/Srinivasan (1974), Albers/Brockhoff (1977), Kohli/Krishnamurti (1987), Kohli/Sukumar (1990), Shi et al. (2001), Balakrishnan et al. (2004), Camm et al. (2006), Albritton/McMullen (2007), Voekler et al. (2013) und vielen anderen mehr (vgl. Baier/Gaul 1999, S. 372/373 und Tabelle 2.2 für weitere Beispiele) beschrieben. Dieses Modell beschreibt die Anzahl derjenigen Konsumenten, die von ihrem Status-quo-Produkt zu einem Produkt aus der Produktlinie wechseln. Wird dann die Anzahl der vom Status-quo-Produkt wechselnden Konsumenten ins Verhältnis zu allen Konsumenten gesetzt, ergibt sich der Share-of-Choices. Wobei nur diejenigen Konsumenten berücksichtigt werden, die nicht bereits ein unternehmenseigenes Produkt kaufen, um einer Kannibalisierung der eigenen Produkte vorzubeugen (Kohli/Krishnamurti 1987, S. 1526). Dieses Verhältnis wurde von Balakrishnan et al. (2004, S. 3) direkt in das Modell eingeführt, basiert jedoch auf dem gleichen Optimierungsmodell wie von Kohli/-Sukumar (1990). Um direkt den Share-of-Choices an der Zielfunktion ablesen zu können, wird die Zielfunktion von Balakrishnan et al. (2004, S. 3) verwendet, bei sonst gleichen Nebenbedingungen. Die Menge I' bezeichne hier die Menge der Konsumenten, die nicht bereits ein unternehmenseigenes Produkt kaufen. Die Teilnutzenwerte der Konsumenten werden normalisiert, so dass ihre Summe 1 ergibt. Dann werden die Teilnutzenwerte jedes Konsumenten für sein Status-quo-Produkt von seinen Teilnutzenwerten abgezogen, womit die β_{ikl} hier relative Teilnutzenwerte darstellen. Unter diesen Voraussetzungen kann das Share-of-Choices-Problem, wie folgt, definiert werden:

$$\frac{1}{|I'|}\sum_{i=1}^{|I'|} x_i \longrightarrow \max! \tag{2.5}$$

[2] In der Literatur als Share-of-Choices-Problem bekannt.

unter den Nebenbedingungen

$$\sum_{l=1}^{L_k} x_{jkl} = 1 \qquad k \in K, j \in R, \tag{2.6}$$

$$\sum_{k=1}^{K} \sum_{l=1}^{L_k} \beta_{ikl} x_{jkl} + x_{ij} > 0 \qquad i \in I', j \in R, \tag{2.7}$$

$$x_i \leq R - \sum_{j=1}^{R} x_{ij} \qquad \forall i \in I', \tag{2.8}$$

$$x_{jkl}, x_{ij}, x_i \in \{0,1\} \qquad i \in I', l \in L_k, k \in K, j \in R. \tag{2.9}$$

Die Zielfunktion (2.5) maximiert den Anteil der Konsumenten, die von ihrem Status-quo-Produkt zu einem Produkt aus der Produktlinie wechseln. Die Nebenbedingung (2.6) stellt sicher, dass jede Eigenschaft der Produkte in der Produktlinie genau ein Level besitzt. Aus Nebenbedingung (2.7) folgt, dass $x_{ij} = 1$, wenn ein Produkt $j \in R$ aus der Produktlinie dem Konsumenten $i \in I$ einen gleich hohen oder niedrigeren Nutzen stiftet als sein Status-quo-Produkt und $= 0$, sonst. Nebenbedingung (2.8) stellt sicher, dass $x_i = 0$, wenn keines der Produkte in der Produktlinie einen höheren Nutzen für Konsument $i \in I$ stiftet als sein Status-quo-Produkt und $= 1$, sonst. Die Restriktionen in Nebenbedingung (2.9) erzwingen die Entscheidungsvariablen ganzzahlig und binär zu sein.

2.2.2.2 Deterministisches Gewinnmaximierungsmodell

Den Vertreter mit First-Choice-Regel für die Gewinnmaximierung einer Produktlinie repräsentiert das Modell von Kohli/Sukumar (1990, S. 1468). Es sei darauf verwiesen, dass die Bezeichnungen der Indizes, Konstanten und Variablen aus Gaul/Baier (2009, S. 172/173) übernommen werden. Die relativen Stückdeckungsbeiträge für jeden Konsumenten $d^r_{ikl} = d_{ikl} - d_{i00}$ werden definiert als Differenz aus den individuellen Stückdeckungsbeiträgen d_{ikl} und den individuellen Stückdeckungsbeiträgen des Status-quo-Produkts eines Konsumenten d_{i00}. Sollte das Status-quo-Produkt allerdings kein unternehmenseigenes Produkt sein, ist $d_{i00} = 0, i \in I$. Mit dieser Formulierung wird einer Kannibalisierung der eigenen Status-quo-Produkte vorgebeugt (Gaul/-Baier 2009, S. 171). Der Nutzenwert u_{i0} ist der Nutzenwert des Status-quo-Produkts eines Konsumenten $i \in I$. Die binären Entscheidungsvariablen des Modells werden durch $x_{ijkl} = 1$, wenn der dem Konsumenten $i \in I$ zugeordnete Produktkandidat $j \in J$ die l-te Eigenschaftsausprägung der k-ten Eigenschaft besitzt; $= 0$, sonst und $y_i = 1$, wenn Konsument i ein Produkt aus der Produktlinie kauft; $= 0$, sonst, repräsentiert. Somit lässt sich das deterministische Gewinnmaximierungsproblem, wie folgt, aufstellen:

$$\sum_{i=1}^{I}\sum_{j=1}^{R}\sum_{k=1}^{K}\sum_{l=1}^{L_k} d_{ikl}^{r} x_{ijkl} y_i \longrightarrow \max! \tag{2.10}$$

unter den Nebenbedingungen

$$\sum_{j=1}^{R}\sum_{l=1}^{L_k} x_{ijkl} = 1 \qquad \forall i,k, \tag{2.11}$$

$$\sum_{l=1}^{L_k} x_{ijkl} - \sum_{l=1}^{L_{k'}} x_{ijk'l} = 0 \qquad k' < k, \forall i,j,k, \tag{2.12}$$

$$x_{ijkl} + x_{i'jkl'} \leq 1 \qquad \forall i < i', l < l', j,k, \tag{2.13}$$

$$\sum_{j=1}^{R}\sum_{k=1}^{K}\sum_{l=1}^{L_k} \beta_{ikl}(x_{ijkl} - x_{i'jkl}) \geq 0 \qquad \forall i \neq i', \tag{2.14}$$

$$y_i \sum_{j=1}^{R}\sum_{k=1}^{K}\sum_{l=1}^{L_k} \beta_{ikl} x_{ijkl} \geq y_i(u_{i0} + \varepsilon) \qquad \forall i, \tag{2.15}$$

$$y_i, x_{ijkl} \in \{0,1\} \qquad \forall i,j,k,l. \tag{2.16}$$

Die Zielfunktion (**??**) maximiert den Gewinn einer Produktlinie mit R Produkten auf Eigenschaftsausprägungsebene. Die Nebenbedingung (2.11) verlangt, dass jedem Konsumenten über alle Produkte in der Produktlinie nur eine Eigenschaftsausprägung für jede Eigenschaft zugeordnet wird. Nebenbedingung (2.12) erfordert, dass über alle Eigenschaften diejenige Eigenschaftsausprägung, die Konsument $i \in I$ zugeordnet ist, vom gleichen Produkt stammt. Damit jede Eigenschaftsausprägung einer Eigenschaft für jeden Konsumenten, der einem Produkt zugeordnet ist, spezifiziert ist, wird Nebenbedingung (2.13) benötigt. Zusammen ergeben die Nebenbedingungen (2.11)-(2.13) die Forderung, dass jedem Konsumenten nur genau ein Produkt aus der Produktlinie zugeordnet wird. Die Nebenbedingung (2.14) stellt sicher, dass jeder Konsument dasjenige Produkt aus der Produktlinie wählt, das ihm den höchsten Nutzen stiftet. Dass ein Konsument ein Produkt aus der Produktlinie nur dann kauft, wenn es einen höheren Nutzen als sein Status-quo-Produkt besitzt, wird mit Nebenbedingung (2.15) ausgedrückt. Damit ein Konsument nur von seinem Status-quo-Produkt wechselt, wenn ein Produkt in der Produktlinie einen echt größeren Nutzen besitzt, wird eine Konstante ε eingeführt, deren konkreter Wert durch das Expertenwissen des Forschers festgelegt werden kann. Schließlich zeigt die Nebenbedingung (2.16) die binären Restriktionen für die Entscheidungsvariablen an.

2.2.3 Probabilistische Modelle

In diesem Abschnitt werden zwei Modelle vorgestellt, die probabilistische Entscheidungsregeln für das Auswahlverhalten der Konsumenten nutzen. Die Marktanteilsmaximierung (vgl. z. B.

Shocker/Srinivasan (1974), Albers (1979), Green et al. (1981), Green/Krieger (1992), Tsafarakis et al. (2011)) sowie die Gewinnmaximierung (vgl. z. B. Green et al. (1981), Green/Krieger (1992), Choi/DeSarbo (1993; 1994), Gaul et al. (1995), Steiner/Hruschka (2000), Chen/Hausman (2000), Steiner/Hruschka (2003), Gaul/Baier (2009)) mittels einer probabilistischen Entscheidungsregel werden im Folgenden detailliert beschrieben. Bei der Marktanteilsmaximierung dominieren die Modelle auf Basis der BTL-Entscheidungsregel und bei der Gewinnmaximierung kommen sowohl die Logit- als auch die BTL-Entscheidungsregel zum Einsatz.

2.2.3.1 Probabilistische Marktanteilsmaximierung

In diesem Abschnitt wird die probabilistische Marktanteilsmaximierung auf Basis der BTL-Entscheidungsregel beschrieben. Es wurde sich zum einen für die BTL- und nicht die Logit-Entscheidungsregel entschieden, da die BTL-Entscheidungsregel in der Literatur häufiger vorkommt und zum anderen, weil es für die Optimierung kaum einen Unterschied macht, welche Entscheidungsregel verwendet wird, da die Auswahlwahrscheinlichkeiten aus dem Logit-Modell lediglich die „Ränder“ der Verteilung der Auswahlwahrscheinlichkeiten glätten und somit kleinere bzw. größere Nutzenwerte der Konsumenten eine etwas größere bzw. eine etwas kleinere Wahrscheinlichkeit zuordnen. Mithilfe des linear-additiven Teilnutzenwertmodells aus (2.1) und der BTL-Entscheidungsregel aus (2.3), wobei sich die Anzahl der Produkte J im Nenner des Bruchs, die den Markt abbilden, aus der einzuführenden Produktlinie, sowie den sich bereits auf dem Markt befindlichen Status-quo-Produkten des eigenen Unternehmens E und der Wettbewerber F zusammensetzen (also: $J = F + E + R$), lässt sich das probabilistische Marktanteilsmaximierungsproblem wie folgt beschreiben (vgl. Tsafarakis et al. 2011, S. 16):

$$\sum_{j=1}^{R} \sum_{i=1}^{I} p_{ij} \longrightarrow \max! \tag{2.17}$$

unter den Nebenbedingungen

$$\sum_{l=1}^{L_k} x_{jkl} = 1 \qquad \forall k, j = 1,2,...,R, \tag{2.18}$$

$$x_{jkl} \neq x_{j'kl} \qquad \text{für mindestens ein } l \in L_k; j, j' \in R, j \neq j', \tag{2.19}$$

$$x_{jkl} \in \{0,1\} \qquad \forall k, l, j = 1,2,...,R. \tag{2.20}$$

Die Zielfunktion (2.17) maximiert den Marktanteil bzw. die Auswahlwahrscheinlichkeiten für die in die Produktlinie einzuführenden R Produkte. Die Nebenbedingung (2.18) stellt sicher, dass jedes Produkt in der Produktlinie für jede Eigenschaft genau eine Eigenschaftsausprägung

besitzt. Nebenbedingung (2.19) wurde dem Modell von Tsafarakis et al. (2011) ergänzt, da es sonst passieren kann, dass Produkte doppelt in der Produktlinie vorkommen, indem sie sich jeweils paarweise in mindestens einer Eigenschaftsausprägung unterscheiden (Steiner/Hruschka 2000, S. 78). Schließlich bildet die Nebenbedingung (2.20) die Binärrestriktion der Entscheidungsvariablen ab.

2.2.3.2 Probabilistische Gewinnmaximierung

Für die probabilistische Gewinnmaximierung wird das Modell von Gaul/Baier (2009, S. 174/175) verwendet, da es ein sehr flexibles Modell durch die Einführung von Fixkosten auf Eigenschaftsausprägungsebene darstellt. Der Status-quo-Markt wird in diesem Modell in Fremdprodukte mit $j = 1,2,...,F$ und Eigenprodukte mit $j = F+1,...,F+E$ untergliedert. Die neu in die Produktlinie einzuführenden Produkte werden somit mit $j = F+E+1,...,F+E+R$ indiziert. Es werden individuell vom Konsumenten abhängige Teilstückdeckungsbeiträge d_{ikl} auf Eigenschaftsausprägungsebene, individuelle bzw. segmentspezifische Teilnutzenwerte β_{ikl}, sowie individuelle Gewichtungsfaktoren ω_i für den Mengenbedarf der Konsumenten bzw. der Segmente verwendet. Des Weiteren wird ein additives Teilfixkostenmodell auf Eigenschaftsausprägungsebene benutzt, um die Fixkosten f_j eines Produktes j über $f_j = \sum_{k=1}^{K}\sum_{l=1}^{L_k} f_{kl}x_{jkl}$, wobei $x_{jkl} = 1$, falls das Produkt j die l-te Eigenschaftsausprägung der k-ten Eigenschaft besitzt; $= 0$ sonst, berechnen zu können. Ein weiterer Vorteil ergibt sich, wenn man bestimmte Eigenschaftsausprägungen für die Optimierung sperren möchte, indem man den Teilfixkosten einen entsprechend hohen Wert zuweist. Mit diesen Voraussetzungen kann das Modell, wie folgt, definiert werden:

$$\sum_{k=1}^{K}\sum_{l=1}^{L_k}\left(\sum_{j=F+E+1}^{F+E+R} x_{jkl}\left(\sum_{i=1}^{I} p_{ij}\omega_i d_{ikl} - f_{kl}\right) + \sum_{j=F+1}^{F+E} x_{jkl}\sum_{i=1}^{I} p_{ij}\omega_i d_{ikl}\right) \longrightarrow \max! \qquad (2.21)$$

unter den Nebenbedingungen

$$\sum_{l=1}^{L_k} x_{jkl} \leq 1 \qquad \forall k, j = F+E+1,...,F+E+R, \qquad (2.22)$$

$$p_{ij} = \frac{\left(\sum_{k=1}^{K}\sum_{l=1}^{L_k}\beta_{ikl}x_{jkl}\right)^{\alpha}}{\sum_{j'=1}^{F+E+R}\left(\sum_{k=1}^{K}\sum_{l=1}^{L_k}\beta_{ikl}x_{j'kl}\right)^{\alpha}} \qquad \forall i, j, \qquad (2.23)$$

$$\sum_{l=1}^{L_k} x_{jkl} = \sum_{l=1}^{L_{k+1}} x_{j(k+1)l} \qquad k = 1,...,K-1, j = F+E+1,...,F+E+R, \qquad (2.24)$$

$$x_{jkl} \in \{0,1\} \qquad \forall k,l, j = F+E+1,...,F+E+R. \qquad (2.25)$$

Die Zielfunktion (2.21) maximiert die Deckungsbeiträge aus neuen und bereits vorhandenen eigenen Produkten abzüglich der Fixkosten für neue Produkte, wobei eine Gewichtung über die eigenschaftsausprägungsspezifischen Teilstückdeckungsbeiträge mit dem Mengenbedarf und den Kaufwahrscheinlichkeiten der Konsumenten stattfindet. Die Nebenbedingung (2.23) repräsentiert die BTL-Entscheidungsregel (siehe Bildungsvorschrift 2.3), wobei p_{ij} die Wahrscheinlichkeit darstellt, dass ein Konsument i ein Produkt j kauft. Nebenbedingung (2.22) sorgt dafür, dass jede Eigenschaft der Produktkandidaten maximal eine Eigenschaftsausprägung hat, während Nebenbedingung (2.24) sichert, dass Produkte komplett mit allen Eigenschaften angeboten werden. Schließlich reflektiert Nebenbedingung (2.25) die Binärrestriktion der Entscheidungsvariablen.

3 Machine-Learning-Verfahren in der Kombinatorischen Optimierung

In diesem Kapitel werden die theoretischen Grundlagen für die Kombinatorische Optimierung gelegt, die für die Produktlinienoptimierung benötigt werden. Es werden wichtige Begriffe wie die des globalen bzw. lokalen Optimums und der Nachbarschaft definiert. Des Weiteren wird eine Übersicht über die in der Kombinatorischen Optimierung verwendeten Verfahren aus dem Machine-Learning gegeben. Außerdem werden die in dieser Arbeit eingesetzten Verfahren detailliert vorgestellt.

Zunächst sollen die Begriffe definiert werden, die für die in dieser Arbeit zugrunde liegenden Optimierungsmodelle und die spätere Verwendung und Implementierung der Verfahren wichtig sind. Aufgrund der binären Entscheidungsvariablen in der Produktlinienoptimierung, der endlichen Anzahl von Lösungen und der Notwendigkeit die Entscheidungsvariablen miteinander zu kombinieren, zählt die Produktlinienoptimierung zur Kombinatorischen Optimierung, die, wie folgt, definiert werden kann (Aarts/Lenstra 1997, S. 5–8):

Definition 1 *Ein **Kombinatorisches Optimierungsproblem** wird durch eine Menge von Probleminstanzen beschrieben und ist entweder ein Minimierungs- oder ein Maximierungsproblem. Eine **Probleminstanz** eines Kombinatorischen Optimierungsproblems ist ein Paar $(\mathscr{L}, f)$, wobei die Lösungsmenge $\mathscr{L}$ die Menge aller zulässigen Lösungen ist und die Bewertungsfunktion f ist eine Abbildung $f : \mathscr{L} \to \mathbb{R}$. Das Ziel ist es, eine **global optimale Lösung** $x^* \in \mathscr{L}$ zu finden, so dass $f(x^*) \leq f(x)$, $\forall x \in \mathscr{L}$ für **Minimierungsprobleme** bzw. $f(x^*) \geq f(x)$, $\forall x \in \mathscr{L}$ für **Maximierungsprobleme** gilt.*

Definition 2 *Sei $(\mathscr{L}, f)$ eine Probleminstanz eines Kombinatorischen Optimierungsproblems. Dann ist die **Nachbarschaftsstruktur** eine Abbildung $\mathscr{N} : \mathscr{L} \to 2^{\mathscr{L}}$, die für jede Lösung $x \in \mathscr{L}$ eine Menge $\mathscr{L}_x \subset \mathscr{L}$ von Lösungen definieren, die nach bestimmten Kriterien „nah“ an x liegen. Die Menge $\mathscr{L}_x$ heißt die **Nachbarschaft** der Lösung x, und jedes $z \in \mathscr{L}_x$ heißt **Nachbarschaftslösung** von x. Des Weiteren wird angenommen, dass gilt, wenn $z \in \mathscr{L}_x \Leftrightarrow x \in \mathscr{L}_z$.*

Definition 3 *Sei $(\mathscr{L}, f)$ eine Probleminstanz eines Kombinatorischen Optimierungsproblems und sei $\mathscr{N}$ eine Nachbarschaftsstruktur, dann heißt $\hat{x} \in \mathscr{L}$* ***lokales Optimum*** *bzgl. $\mathscr{N}$, falls $\hat{x}$ gleich oder besser als alle Nachbarschaftslösungen bzgl. des Zielfunktionswerts ist. Für ein* ***lokales Minimum*** *gilt: $f(\hat{x}) \leq f(z)$, $\forall z \in \mathscr{L}$. Für ein* ***lokales Maximum*** *gilt: $f(\hat{x}) \geq f(z)$, $\forall z \in \mathscr{L}$.*

Diese Definitionen sind essentiell für das Fundament der Produktlinienoptimierung. Es kann entschieden werden, wann eine Lösung das globale Optimum darstellt. Dies ist der Fall, wenn es keine Lösung gibt, die einen kleineren oder größeren Zielfunktionswert als eine Lösung x^* hat. Das Bestimmen dieser Lösung ist das Ziel der Kombinatorischen Optimierung, was jedoch häufig nicht möglich ist, da die Lösungsmengen zu groß für exakte Lösungsverfahren, wie Branch&Bound, Lagrange-Relaxation oder kompletter Enumeration, sind. Aus diesem Grund muss man sich oft mit lokalen Optima $\hat{x}$ begnügen, die auf einer Nachbarschaftsstruktur $\mathscr{N}$ definiert sind.

Da die in dieser Arbeit betrachteten Produktlinienoptimierungsprobleme zu den Maximierungsproblemen zählen, werden alle weiteren Ausführungen unter dieser Prämisse getätigt. In der Produktlinienoptimierung konstituiert sich die Lösungsmenge $\mathscr{L}$ aus allen zulässigen Produktlinien, die sich wiederum aus Entscheidungsvariablen zusammensetzen, die die Eigenschaften und deren Ausprägungen eines Produkts in der Produktlinie kodieren. Die Bewertungsfunktion f repräsentiert entweder den zu maximierenden Gewinn oder den zu maximierenden Marktanteil. Der Begriff der Nachbarschaftslösung wird benötigt, um ein lokales Optimum, bzw. für die Produktlinienoptimierung lokales Maximum, zu definieren. Eine Nachbarschaft ist für die Produktlinienoptimierung nicht universell definierbar, da die Nachbarschaft von den verwendeten Verfahren abhängt. Für Verfahren, bei denen während der Optimierung eine Eigenschaft verändert und evaluiert wird, entspricht die Nachbarschaft $\mathscr{L}_x$ einer Lösung x denjenigen Produktlinien z, die sich lediglich in einer Eigenschaft von x unterscheiden. Diejenige Lösung $\hat{x}$, deren Zielfunktionswert größer oder gleich der Zielfunktionswerte aller Nachbarschaftslösungen ist, wird als lokales Maximum bezeichnet. Werden in den Verfahren mehrere Eigenschaften der Produktlinie gleichzeitig verändert, entspricht das lokale Maximum derjenigen Lösung, die den höchsten Zielfunktionswert in dieser Nachbarschaftsstruktur besitzt. In diese Kategorie gehören bspw. die Implementierungen der in dieser Arbeit verwendeten Verfahren. Auch dies ist nicht allgemeingültig und hängt von der konkreten Implementierung der Verfahren ab. Des Weiteren existieren Verfahren, die nicht einzelne Eigenschaften während der Optimierung verändern und evaluieren, sondern vollständige Produkte der Produktlinie. In diesem Fall entspricht das lokale Maximum derjenigen Lösung, die durch den Austausch eines Produkts der Produktlinie den Zielfunktionswert nicht weiter verbessern kann. Beispiele hierfür sind die Greedy-Heuristik von Green/Krieger (1985) oder die Dynamic-Programming-Heuristik von Kohli/Krishnamurti (1987). Ergo, ist das lokale Optimum von der Nachbarschaftsstruktur abhängig, die wiederum vom zugrundeliegenden Problem sowie der Implementierung der Verfahren abhängt.

In den folgenden Abschnitten werden praktische Probleme der Lösung von kombinatorischen Optimierungsproblemen beschrieben, eine Literaturübersicht über verschiedene Verfahren seit 1995 gegeben und diejenigen Verfahren vorgestellt, die in dieser Arbeit zum Einsatz kommen.

3.1 Herausforderungen bei der Lösung Kombinatorischer Optimierungsprobleme

Bei dem Versuch Kombinatorische Optimierungsprobleme zu lösen, treten verschiedene Probleme auf, die in diesem Abschnitt kurz beleuchtet werden.

Das entscheidendste Problem ist die Tatsache, dass viele Kombinatorische Optimierungsprobleme NP-hard sind. Die Produktlinienoptimierung gehört zu dieser Klasse von Problemen, die NP-hard sind (Kohli/Krishnamurti 1989). NP-hard bedeutet, dass die optimale Lösung im Allgemeinen nicht in polynomieller Zeit gefunden werden kann. Selbst die im Folgenden genannten exakten Verfahren können häufig nicht angewendet werden, da sie das globale Optimum nicht in angemessener Zeit bestimmen können.

Zur Lösung Kombinatorischer Optimierungsprobleme gibt es eine Vielzahl kommerzieller Software. Zu nennen sind hier bspw. AMPL, CPLEX, Lindo, GAMS, Maple, Mathematica u. v. m. Diese High-Level-Solver nutzen meist eine Kombination aus Standardverfahren der linearen Optimierung sowie Branch-& -Bound-Techniken für den ganzzahligen Anteil. Zur Lösung Ganzzahliger Optimierungsprobleme existieren verschiedene Methoden, wie z. B. Branch-& -Bound-Verfahren, Schnittebenenverfahren, Branch-& -Cut-Verfahren (eine Kombination aus Branch-&-Bound- und Schnittebenenverfahren) und eine Kombination aus Branch-& -Bound-Verfahren mit (Lagrange-) Relaxationstechniken. Trotz dieser Vielzahl an Software ist es für die meisten Probleminstanzen nicht möglich, das globale Optimum für Produktlinienoptimierungsprobleme zu finden. Deshalb wurden Verfahren entwickelt, die gute Näherungen, z.B. lokale Optima, an das globale Optimum erreichen können.

Die Frage, die sich daraus ergibt, ist, welche Verfahren für welche Kombinatorischen Optimierungsprobleme besonders geeignet sind. Auf diese Frage kann es jedoch keine allgemeingültige Antwort geben. So zeigten Wolpert/Macready (1997) mit ihrem Satz „No free lunch theorems for optimization“, dass es kein Verfahren gibt, dass für *alle* Optimierungsprobleme gut geeignet ist und im Mittel über alle möglichen Optimierungsprobleme alle Verfahren gleich sind. Hieraus ergeben sich wichtige Konsequenzen für die Auswahl eines Verfahrens für ein bestimmtes Optimierungsproblem, da im Vorhinein nicht mit Gewissheit gesagt werden kann, welches Verfahren

tatsächlich geeignet ist. Aus diesem Grund ist es entscheidend, das zugrundeliegende Optimierungsproblem gut zu kennen, und anschließend zu entscheiden, welche Verfahren geeignet sein könnten und eine Auswahl davon gegeneinander zu benchmarken. Im Allgemeinen ist es zum Beispiel nicht möglich zu sagen, dass für die Produktlinienoptimierung genetische Algorithmen gut funktionieren werden, mit der Begründung, dass sie für das Problem des Handlungsreisenden gute Ergebnisse liefern. In der Praxis wird daher versucht, andere Optimierungsprobleme zu identifizieren, die eine ähnliche Struktur wie das zugrundeliegende aufweisen, um wenigstens einen Anhaltspunkt zu haben, ob sich ein bestimmtes Verfahren eignen könnte.

Dieses Problem der Auswahl eines geeigneten Verfahrens verstärkt sich noch durch die große Anzahl von Verfahren im Bereich der näherungsweisen Optimierung. In den letzten Jahren gab es große Kritik an diesem Forschungsbereich für neue Verfahren, die vor allem durch Soerensen (2013) vorgetragen wurde. Der Autor kritisiert, dass es in den letzten 20 Jahren eine Explosion von neuen Verfahren zur approximativen Lösung Kombinatorischer Optimierungsproblemen gibt, die in Wirklichkeit nicht neu sind, sondern sich weiteren Metaphern aus der Natur bedienen, die das Verhalten von Tieren, Menschen, biologischen, chemischen oder physikalischen Systemen imitieren, ohne konzeptionell etwas Neues zum Forschungsbereich beizutragen. So werden springende Frösche, die Lichtbrechung, das Spielen eines Orchesters, die Spiralbewegungen von Galaxien, die Ausbreitung von Königreichen etc. herangezogen, obwohl diese Mechanismen nicht im Entferntesten etwas mit mathematischer Optimierung gemein haben. Häufig werden einfach bestimmte Elemente aus bestehenden Verfahren wie genetischen Algorithmen, Simulated Annealing, Ameisenalgorithmen oder Particle-Swarm-Optimization miteinander kombiniert und so verkauft, dass ein bestimmtes „neues“ Verfahren einen Sachverhalt aus der Natur modelliert und meist besser abschneidet als die bestehenden Verfahren. Ein genauerer Blick verrät jedoch meistens, dass es sich um unpräzise Wissenschaft handelt und die bestehenden Verfahren so ausgewählt wurden, bzw. entsprechende Parameterkonfigurationen, dass diese schlechter abschneiden müssen oder die Probleminstanzen wurden so klein gewählt, dass die Verfahren nicht überraschend gut funktionierten, wie z. B. bei der Produktlinienoptimierung Particle-Swarm-Optimization von Tsafarakis et al. (2011) und Ant-Colony-Optimization von Albritton/McMullen (2007). Schafft ein „neues“ Verfahren nicht einmal das, wird davon gesprochen, dass es zumindest „competitive“ zu bestehenden Verfahren ist. Des Weiteren fehlen häufig statistische Tests, mit denen die Performance verschiedener Algorithmen miteinander verglichen werden kann, da es sich durch die approximative Natur der Verfahren immer um statistische Experimente handelt. Zudem wohnen Zufallsmechanismen, wie die Mutation bei genetischen Algorithmen oder die Akzeptanzwahrscheinlichkeit bei Simulated Annealing, den meisten Verfahren inne, wodurch sie inhärent stochastisch und nicht deterministisch sind, weshalb wiederholte Zufallsexperimente für den Vergleich von Verfahren unabdingbar sind.

Aus diesen Gründen wird sich in dieser Arbeit auf die Verfahren konzentriert, die bereits in der Produktlinienoptimierung Verwendung fanden und diese entsprechend verbessert. „Neue“

Verfahren wie Harmony-Search von Geem et al. (2001) oder der Multiverse-Optimizer von Mirjalili et al. (2015) brachten bspw. keine Vorteile für die Produktlinienoptimierung, weshalb deren Ergebnisse in dieser Arbeit nicht berichtet werden. Des Weiteren wird sich auf die bewährten Verfahren konzentriert, da es hierzu bereits ein starkes theoretisches Fundament gibt und sich die Eignung dieser Verfahren für viele Optimierungsprobleme gezeigt hat.

3.2 Übersicht verschiedener Verfahren

In diesem Abschnitt werden verschiedene Heuristiken seit 1995 vorgestellt. Die Wahl des Zeitraums hängt zum einen mit dem Beginn der Literaturanalyse des Hauptthemas dieser Arbeit zusammen (siehe Abschnitt 4.1) und zum anderen mit einem Boom der Entwicklung von Heuristiken Ende der 1990er und Anfang der 2000er Jahre. Die Publikationen zu den Heuristiken werden ab dem Jahr 1995 berücksichtigt. Somit ist die Betrachtung konsistent zum Startzeitpunkt der Literaturanalyse der Produktlinienoptimierung.

Tabelle 1 soll einen Eindruck über die große Anzahl heuristischer Optimierungsalgorithmen geben. Außerdem werden diejenigen Verfahren fett hervorgehoben, die konzeptionell etwas Neues im Gegensatz zu den bestehenden Verfahren bieten können und nicht nur weitere Metaphern mit den gleichen Mechanismen darstellen. Wie sich zeigt, ist die Kritik von Soerensen (2013) berechtigt. Lediglich zwei Konzepte konnten identifiziert werden, die konzeptionell eine Neuheit im eigentlichen Sinn bieten. Dazu zählen die Particle-Swarm-Optimization von Kennedy/Eberhart (1995a) und Neural Combinatorial Optimization with Reinforcement Learning von Bello et al. (2016). Letzterer Ansatz scheiterte an den Benchmarkdatensätzen für das Problem des Handlungsreisenden insofern, dass die gefundenen Zielfunktionswerte nicht der Qualität etablierter Verfahren genügen konnten. Die anderen Verfahren imitieren mit einfachen Modellen das Verhalten von Wasser, Bienen, Imperien und weiteren Metaphern, bedienen sich dabei jedoch der grundlegenden Mechanismen von Genetischen Algorithmen (z. B. Selektion und Crossover in Differential Evolution), nicht-deterministischen Regeln zur Entscheidung, ob eine neue Lösung auch dann akzeptiert wird, wenn sie schlechtere Zielfunktionswerte liefert, um aus lokalen Optima zu entkommen wie bei Simulated Annealing (siehe z. B. den Multi-verse Optimizer) oder diversen Reinforcement-Learning-Mechanismen angelehnt an die Ameisenalgorithmen (siehe z. B. Bee-Colony-Optimization). Um einen Eindruck über das Gebiet zu vermitteln, wurden für den genannten Zeitraum diejenigen Verfahren für Tabelle 1 aufgeführt, die eine relevante Zahl an Zitationen aufweisen. Eine weitere Übersicht findet sich z. B. in Fausto et al. (2020).

Heuristik	**Quelle**	**Heuristik**	**Quelle**
Particle-Swarm Optimization	Kennedy/Eberhart (1995a)	Termite Colony Optimization	Hedayatzadeh et al. (2010)
Differential Evolution	Storn/Price (1997)	Eco-inspired Evolutionary Alg.	Parpinelli/Lopes (2011)

(Fortsetzung der Tabelle auf der nächsten Seite)

Heuristik	Quelle	Heuristik	Quelle
Grammatical Evolution	Ryan et al. (1998)	Galaxy-based Search Algorithm	Shah-Hosseini (2011)
Gene Expression	Ferreira (2001)	Big Bang - Big Crunch	Zandi et al. (2012)
Harmony Search	Geem et al. (2001)	Differential Search Algorithm	Civicioglu (2012)
Marriage in Honey Bees	Abbass (2001)	Electro-magnetism Optimization	Cuevas et al. (2012)
Bacterial Foraging	Passino (2002)	Flower Pollination Algorithm	Yang (2012)
Queen-bee Evolution	Jung (2003)	Krill Herd	Gandomi/Alavi (2012)
Bee Hive	Wedde et al. (2004)	OptBees	Maia et al. (2012)
Bee Colony Optimization	Teodorović/Dell'Orco (2005)	Water Cycle Algorithm	Eskandar et al. (2012)
Bees Swarm Optimization	Drias et al. (2005)	Weightless Swarm Algorithm	Ting et al. (2012)
Virtual Bees	Yang (2005)	Wolf Search	Tang et al. (2012)
Bees Algorithm	Pham et al. (2006)	Migrating Birds Optimization	Duman et al. (2012)
Cat Swarm	Chu et al. (2006)	Artificial Cooperative Search	Civicioglu (2013a)
Invasive Weed Optimization	Mehrabian/Lucas (2006)	Backtracking Optimization Search	Civicioglu (2013b)
Artificial Bee Colony	Karaboga/Basturk (2007)	Black Hole	Hatamlou (2013)
Bacterial-GA Foraging	Chen et al. (2007)	Dolphin Echolocation	Kaveh/Farhoudi (2013)
Central Force Optimization	Formato (2007)	Symbiotic Organisms Search	Cheng/Prayogo (2014)
Good Lattice Swarm Optimization	Su et al. (2007)	Grey Wolf Optimizer	Mirjalili et al. (2014)
Imperialist Competitive Algorithm	Atashpaz-Gargari/Lucas (2007)	Optics Inspired Optimization	Kashan (2015)
Intelligent Water Drop	Hosseini (2007)	Multi-verse Optimizer	Mirjalili et al. (2015)
River Formation Dynamics	Rabanal et al. (2007)	Stochastic Fractal Search	Salimi (2015)
Biogeography-based Optimization	Simon (2008)	Prey-predator Algorithm	Tilahun/Ong (2015)
Fast Bacterial Swarming	Chu et al. (2008)	Shark Smell Optimization	Abedinia et al. (2016)
Fish-school Search	Bastos Filho et al. (2008)	**Neural Combinatorial Optimization with Reinforcement Learning**	Bello et al. (2016)
Roach Infestation Algorithm	Havens et al. (2008)	Football Game inspired Algorithm	Fadakar/Ebrahimi (2016)
Bumblebees	Comellas/Martinez-Navarro (2009)	Water Evaporation Optimization	Kaveh/Bakhshpoori (2016)
Cuckoo Search	Yang/Deb (2009)	Elephant Herding Optimization	Wang et al. (2016)
Gravitational Search	Rashedi et al. (2009)	Sonar inspired Optimization	Tzanetos/Dounias (2017)
Group Search Optimizer	He et al. (2009)	Sperm Motility Algorithm	Raouf/Hezam (2017)
Human-Inspired Algorithm	Zhang et al. (2009)	Car Tracking Optimization	Chen et al. (2018)
League Championship Algorithm	Kashan (2009)	Pity Beetle Algorithm	Kallioras et al. (2018)
Paddy Field Algorithm	Premaratne et al. (2009)	Coyote Optimization Algorithm	Pierezan/Coelho (2018)
Bat Algorithm	Yang (2010b)	Farmland Fertility	Shayanfar/Gharehchopogh (2018)
Charged System Search	Kaveh/Talatahari (2010)	Emperor Penguins Colony	Harifi et al. (2019)
Consultant-guided Search	Iordache (2010)	Artificial Feeding Birds	Lamy (2019)
Firefly Algorithm	Yang (2010a)	Binary Whale Optimization	Reddy K et al. (2019)
Hierarchical Swarm Model	Chen et al. (2010)	Slime Mould Algorithm	Li et al. (2020)

Tabelle 1: Chronologische Übersicht von heuristischen Verfahren zur Lösung von Optimierungsproblemen seit 1995. Fettgedruckte Einträge bedeuten eine konzeptionelle Neuheit. (Quelle: Eigene Darstellung)

Auffällig ist, dass der Forschungsbereich trotz ausbleibender echter neuer Ideen und besserer Ergebnisse als der etablierten Verfahren weiterhin floriert. Die verwendeten Metaphern bedienen sich aus vielen verschiedenen Gebieten der Biologie, Chemie, Physik, Technik und sogar der Landwirtschaft. Vor allem das Verhalten von Tieren scheint eine besondere Faszination bei der Entwicklung von Heuristiken auf die Autoren auszuüben.

In den kommenden Abschnitten werden diejenigen Verfahren beschrieben, die für die späteren Simulationsrechnungen verwendet werden. Es handelt sich hierbei um lange etablierte Verfah-

ren, die zum Teil auch schon in der Produktlinienoptimierung Anwendung fanden. Aufgrund der fehlenden Innovationen in diesem Forschungsgebiert erscheint es zielführend, bewährte Verfahren zu verbessern und zu vergleichen.

3.3 Genetische Algorithmen

Man spricht bei Genetischen Algorithmen selten von „dem Genetischen Algorithmus“ in der Einzahl, weil es eine ganze Bandbreite von Genetischen Algorithmen gibt. Als erste Veröffentlichung zu diesem Thema, die einen klassischen Genetischen Algorithmus abbildet, gilt die Arbeit von Holland (1975). Seitdem haben die Genetischen Algorithmen eine sehr große Verbreitung mit sehr vielen verschiedenen Anwendungen gefunden (siehe z.B. Tabelle 3). Ihre Popularität verdanken sie zum einen ihrem einfachen Aufbau und der damit verbundenen relativ einfachen Implementierung. Zum anderen zeigen sie gegenüber anderen Heuristiken häufig eine gleichwertige oder überlegene Performance. Die Inspiration für einen solchen Algorithmus holte sich der Autor in der Natur. Er stellte fest, dass lebende Organismen im Allgemeinen hervorragend in der Lage sind, Probleme zu lösen. Außerdem bietet die Natur grundsätzlich zwei Arten des Lernens: das Gehirn und die Evolution. Die Vielseitig- und Anpassungsfähigkeit von Organismen aufgrund der Evolution und der natürlichen Auslese lassen die Anstrengungen von Informatikern bei der Entwicklung von hochspezialisierten Algorithmen, für deren Entwicklung es Wochen oder Monate benötigt, verblassen. Daher können sich Wissenschaftler die Evolution zum Vorbild nehmen, da die natürliche Selektion eine der größten Hürden in der Softwareentwicklung nehmen kann, nämlich die Spezifikation und die Interaktionen aller Eigenschaften eines Problems bereits im Vorhinein. So war es bereits früh möglich Durchbrüche bei dem Design von komplexen Systemen wie Flugzeugmotoren zu verzeichnen. Ein großer Vorteil bei Genetischen Algorithmen ist ihre Fähigkeit einen weiten Bereich des Lösungsraums zu erschließen, um somit gute Lösungen zu finden. Die meisten Organismen entwickeln sich durch zwei maßgebliche Prozeduren: natürliche Selektion und Reproduktion (vgl. Holland 1992, S. 66).

Die in Tabelle 2 aufgeführten Bezeichnungen sind in der Literatur nicht ganz einheitlich, so wird ein String zum Teil synonym zum Chromosomen verwendet. In der vorliegenden Arbeit wird jedoch die oben stehende Vereinbarung benutzt.

Eine der ersten ausführlichen Beschreibungen für die Anwendung Genetischer Algorithmen und deren Implementierung mithilfe von Computerprogrammen tätigte Goldberg (1989). Auf Grundlage dieser Arbeit wird nun eine allgemeine Beschreibung von Genetischen Algorithmen vorgenommen. Bei Genetischen Algorithmen handelt es sich um stochastische und randomisierte Suchalgorithmen auf Basis der natürlichen Evolution. Trotz der Randomisierung bestimmter Operationen handelt es sich nicht um einfache *random walks*, sondern die Genetischen Algo-

Bezeichnung	Metapher	mathematische Entsprechung
Gen	einzelner Teil eines Chromosoms oder eines Strings	einzelne Variable
String	beliebig großer Teil eines Chromosoms bestehend aus Genen	Teilmenge von Variablen
Chromosom	Darstellung des Parametersets durch einzelne Gene	eine zulässige Variablenkonfiguration (Lösung)
Population	Menge von Chromosomen	Teilmenge von zulässigen Variablenkonfigurationen (Lösungen)
Fitness	Überlebensfähigkeit eines Chromosoms	Zielfunktionswert

Tabelle 2: Beschreibung der Metaphern für Genetische Algorithmen und deren Entsprechung im Kontext der mathematischen Optimierung. (Quelle: Eigene Darstellung)

rithmen erschließen den Lösungsraum effizient durch die Nutzung von Informationen aus der Vergangenheit. Durch diese Informationen werden systematisch neue Konfigurationen entdeckt, die die erwartete Performance verbessern (Goldberg 1989, S. 1). Die Genetischen Algorithmen unterscheiden sich in vier Punkten grundlegend von traditionellen Methoden:

1. Genetische Algorithmen arbeiten mit einer Codierung des Parametersets und nicht mit den Parametern selbst.

2. Genetische Algorithmen arbeiten mit einer Population von Punkten und nicht mit einem einzelnen Punkt.

3. Genetische Algorithmen nutzen direkt die Werte der Zielfunktion und kommen ohne weitere Informationen bzw. Ableitungen von den Zielfunktionen aus.

4. Genetische Algorithmen nutzen probabilistische Transformationsregeln und keine deterministischen.

Der erste Punkt bedeutet, dass ein natürliches Parameterset gefunden werden muss, das eine endliche Länge des Chromosoms voraussetzt. Bei traditionellen Verfahren würde bspw. bei der Optimierung der Funktion $f = x^2$ auf einem Intervall $[a,b]$ versucht, den Parameter x selbst zu verbessern. Bei Genetischen Algorithmen würde nun das vorher festgelegte Parameterset verändert, um zu einer Verbesserung des Zielfunktionswertes zu gelangen. Der zweite Punkt bedeutet, dass nicht nur versucht wird, einen einzelnen Punkt in jedem Zeitschritt zu verbessern, sondern ein ganzes Set von Punkten, eine Population. Ein sehr wichtiger Vorteil von Genetischen Algorithmen besteht in Punkt 3, nämlich, dass keine Ableitungen von Funktionen gebildet werden

müssen. Umso höher die Dimension einer Funktion, umso aufwändiger ist es i. d. R. die Ableitungen bzw. die Jacobi- und Hessematrix zu bilden. Der vierte Punkt stellt sicher, dass zum einen ein möglichst großer Lösungsraum abgesucht werden kann und zum anderen, dass die Möglichkeit besteht, aus lokalen Optima zu entkommen und nicht vorzeitig zu konvergieren (Goldberg 1989, S. 7).

Aufgrund der Vielzahl Genetischer Algorithmen ist es schwierig, genau zu sagen, aus welchen genauen Gründen Genetische Algorithmen so gut funktionieren (Reeves 2003, S. 61). In der Literatur existieren grundlegend zwei Herangehensweisen, um sich diesem Problem zu nähern. Zum einen eine theoretische und zum anderen eine praktische Herangehensweise. In der theoretischen Herangehensweise wird versucht sich dem Problem mathematisch zu nähern. Eine Möglichkeit hierfür sind Markov-Ketten. Dies wurde u.a. in den Arbeiten von Vose (1993), Whitley (1993), Vose (1994), Vose/Wright (1995) versucht. Eine andere Möglichkeit zogen Shapiro et al. (1994) in Betracht. Sie näherten sich dem Problem über eine statistische Analyse. Während der Markov-Ketten-Ansatz versucht vorherzusagen wie sich jedes Chromosom der Population in der nächsten Iteration entwickeln wird, setzt der statistische Ansatz auf die Entwicklung der statistischen Eigenschaften der Population. Hierin wird u.a. gezeigt, dass der Selektionsoperator nicht von der Probleminstanz, sondern nur von den Fitness-Werten abhängig ist. Wohingegen Crossover- und Mutationsoperatoren für jede Probleminstanz individuell zu bestimmen sind (Shapiro et al. 1994, S. 17). Der bekannteste Ansatz, der versucht zu erklären aus welchen Gründen Genetische Algorithmen in der Optimierung gut funktionieren, bietet Holland (1975) mit dem *Schema Theorem*.

Quelle	**Anwendung**
Wang/Hsu (2010)	Lösung von Kreislaufproblemen in der umweltorientierten Logistik und Beschaffung
Karegowda et al. (2011)	Initialisierung und Optimierung von Verbindungsgewichten für Backpropagation in neuronalen Netzen
Moradi/Abedini (2012)	Lösung für den optimalen Standort und die optimale Dimensionierung von dezentralen Energieanlagen (z.B. Brennstoffzellen)
Roberge et al. (2013)	Routenplanung für unbemannte und eigenständig fliegende Flugkörper
Dutta et al. (2018)	Portfoliooptimierung mittels stochastischem Preisszenario

Tabelle 3: Anwendungsbeispiele für Genetische Algorithmen. (Quelle: Eigene Darstellung)

Für Genetische Algorithmen gibt es in der Literatur viele Anwendungsbeispiele. Beispielhaft sind hierfür einige in Tabelle 3 aufgeführt. Auffällig ist, dass Genetische Algorithmen aufgrund ihrer einfachen Implementierung und ihrer großen Flexibilität bei der Modellierung für viele

verschiedene Fragestellungen genutzt werden. Anwendungen in der Logistik, der Optimierung von dezentralen Energieanlagen, die Routenplanung für Drohnen, dem Finanzwesen oder der Optimierung von Verbindungsgewichten bei neuronalen Netzen zeigen diese Tatsache auf.

3.3.1 Beschreibung Genetischer Algorithmen

Im Folgenden wird die allgemeine Funktionsweise von Genetischen Algorithmen näher erläutert.

Im Allgemeinen werden die Eigenschaften eines Optimierungsproblems als Chromosomen bzw. Genketten modelliert, die aus einzelnen Genen bestehen. Prinzipiell sind alle Genetischen Algorithmen gleich aufgebaut. Sie bestehen aus den folgenden drei Schritten:

1. Selektion,

2. Rekombination bzw. Reproduktion durch Crossover,

3. Mutation.

In der Literatur gibt es verschiedene, nicht ganz einheitliche Bezeichnungsweisen dieser drei Schritte. Das Flussdiagramm in Abbildung 1 zeigt die Abfolge der Operatoren.

Dieser Mechanismus, der im Prinzip nur aus dem Kopieren von Strings und dem Austausch von Teilstrings zwischen den Chromosomen besteht, ist sehr einfach (vgl. Goldberg 1989, S. 10). Die Schwierigkeiten bestehen darin, dass jedes neue Problem eine andere Chromosomendarstellung benötigt, die jedes Mal neu modelliert werden muss. Das ist gleichzeitig der große Nachteil an Heuristiken gegenüber exakten mathematischen Lösungsverfahren. Diese haben einen Anspruch auf Allgemeingültigkeit, Heuristiken meist nicht.

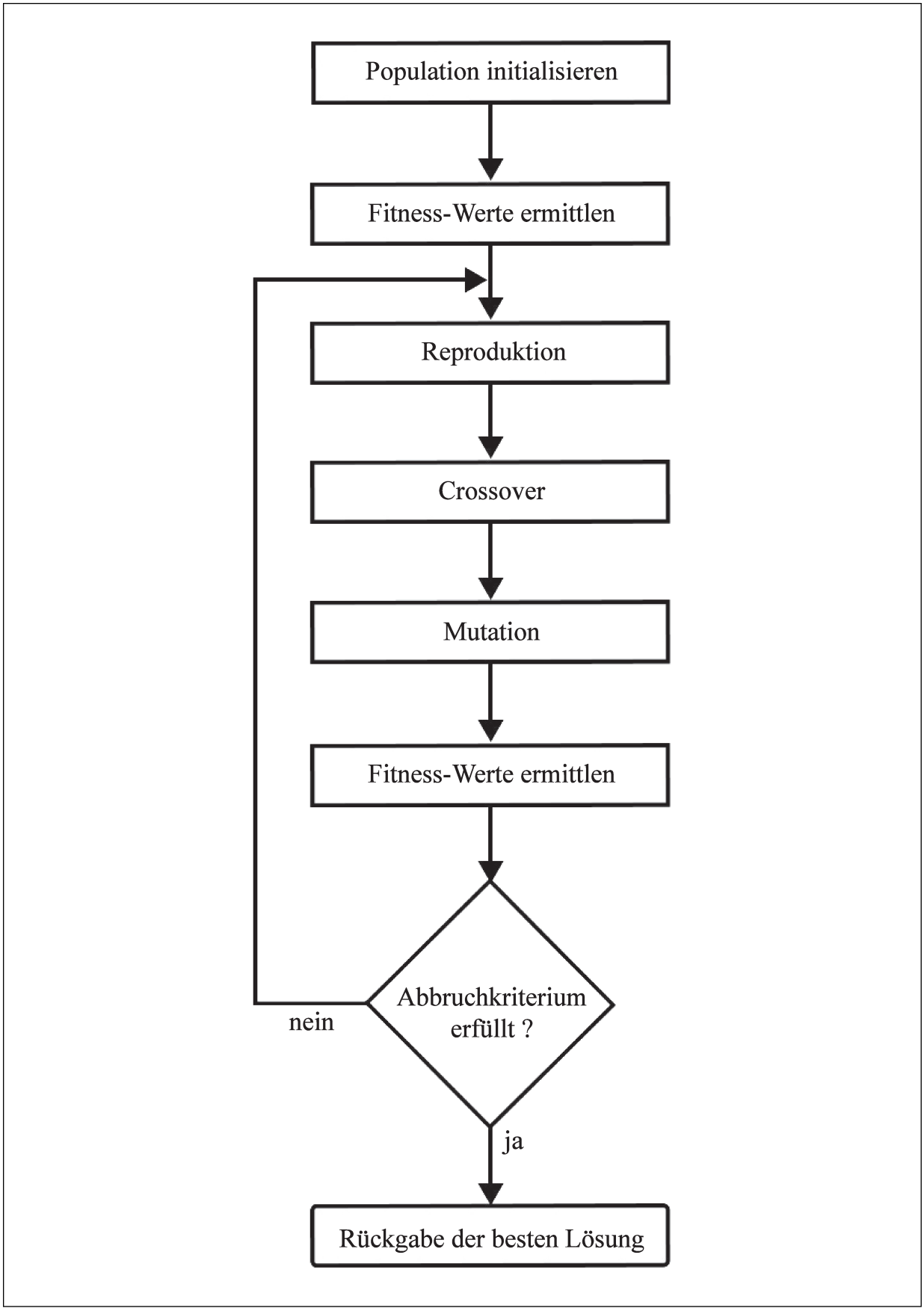

Abbildung 1: Flussdiagramm eines einfachen Genetischen Algorithmus. (Quelle: Eigene Darstellung)

Der prinzipielle Ablauf eines Genetischen Algorithmus ist in Abbildung 1 illustriert. Zunächst wird die Population initialisiert, d. h. es werden mehrere zulässige Variablenkonfigurationen (Chromosomen) zufällig erzeugt und zu einer Population zusammengefasst. Danach werden die zugehörigen Fitness-Werte, also die Zielfunktionswerte, ermittelt. Für die Reproduktion wird ein bestimmter, vorher festgelegter Anteil dieser Population ausgewählt. Welche Chromosomen zur Reproduktion ausgewählt werden, hängt von deren Fitness-Werten ab. Umso höher die Fitness-Werte, desto höher ist die Chance, dass ein Chromosom zur Reproduktion herangezogen wird. Der ausgewählte Anteil der Chromosomen wird durch den Crossover-Operator rekombiniert. Das bedeutet, dass zwei Strings an einer zufällig ausgewählten Stelle getrennt und zu einem oder zwei neuen Strings wieder zusammengeführt werden. Für den Crossover-Operator gibt es verschiedene Möglichkeiten, die in den folgenden Abschnitten näher erläutert werden. Somit entsteht aus den beiden *parents* die nächste Generation (*children*). Alle Generationen unterliegen dem darwinistischen Selektionsdruck, der nur denjenigen Individuen das Weiterkommen in eine neue Generation ermöglicht, die am besten angepasst sind („survival of the fittest“). Zufällige Mutationen haben bei Genetischen Algorithmen zwei Funktionen: zum einen stellen sie die Diversifizierung der Population mit neuen Eigenschaften sicher. Hierdurch wird verhindert, dass der Algorithmus zu schnell gegen ein ungünstiges lokales Optimum konvergiert. Zum anderen unterstützen Mutationen die lokale Suche in der späteren Suche, um ein lokales Optimum zu finden. Nachdem die Fitness-Werte der neuen Generation ermittelt wurden, entscheidet ein Abbruchkriterium, ob die Suche beendet oder fortgeführt wird. Ein Abbruchkriterium kann die Konvergenz gegen ein lokales Optimum sein. Das wird sichergestellt durch den Selektionsdruck, indem das beste Individuum immer in die neue Generation übernommen wird. Bei Konvergenz besteht die gesamte Population nur noch aus einem Individuum (Mühlenbein 1997, S. 143/156). Kann die Konvergenz nicht sichergestellt werden, z. B. weil das zugrundeliegende Problem zu langsam konvergiert, können weitere Abbruchkriterien sein: die Anzahl an Iterationen, die Laufzeit des Algorithmus oder eine bestimmte Anzahl von Iterationen, in der es keine Veränderung des aktuell besten Fitness-Wertes gibt. Nach Beendigung des Algorithmus wird die Lösung mit dem höchsten Fitness-Wert als beste gefundene Lösung zurückgegeben.

Eine wichtige Frage für Genetische Algorithmen ist die richtige, bestenfalls die optimale **initiale Populationsgröße** in Abhängigkeit der Länge der Chromosomen. Grundsätzlich besteht dabei ein Trade-off zwischen Effizienz und Effektivität. Ist die initiale Population zu klein, wird der Suchraum nicht effektiv erkundet. Ist die initiale Populationsgröße zu groß, kann nicht erwartet werden, dass eine Lösung des Optimierungsproblems in angemessener Zeit gefunden werden kann, was einer effizienten Suche widerspricht (Reeves 2003, S. 63). Diese Frage kann nicht eindeutig beantwortet werden, weshalb an dieser Stelle auf die Diskussion in Reeves (2003, S. 63) verwiesen wird.

Für die Terminierung Genetischer Algorithmen gibt es verschiedene Möglichkeiten. Je nach Ausgestaltung der genetischen Operatoren ist eine Konvergenz gegen ein lokales Optimum

nicht immer gegeben, weshalb Genetische Algorithmen zumindest theoretisch für immer laufen könnten. In der Praxis werden realistische Abbruchkriterien benötigt. Das können z.B. die Laufzeit oder die Anzahl untersuchter Chromosomen sein. Ein weiteres Abbruchkriterium kann sein, wenn die Population den Großteil ihrer Diversität verloren hat (Reeves 2003, S. 63). Außerdem könnte abgebrochen werden, wenn es über eine bestimmte Anzahl von Generationen kein neues bestes Chromosom mit höherem Fitness-Wert als in den vorherigen Generationen gibt oder sich das arithmetische Mittel der Population nicht oder kaum mehr verändert.

In den folgenden Abschnitten werden Mechanismen für die Selektion, die Reproduktion, die Mutation und die Populationswahrung vorgestellt.

3.3.2 Selektionsmechanismen

Für die Selektion existieren verschiedene Mechanismen, die einen Einfluss auf die Zusammensetzung der Chromosomen haben und somit die Entwicklung der Gesamtpopulation beeinflussen. Die wichtigsten Selektionsmechanismen werden im Folgenden vorgestellt. Es werden die englischen Bezeichnungsweisen für die verwendeten Fachbegriffe beibehalten, um Klarheit über die Bezeichnungen herzustellen, da zu diesem Thema keine relevante deutsche Literatur bekannt ist und somit keine Begriffskonventionen über die deutsche Übersetzung. Die bedeutendsten Selektionsmechanismen mit den Bezeichnungen

- Truncation-Selektion,
- Proportionate-Selektion bzw. Roulette-Wheel-Selektion,
- Ranking-Selektion,
- Tournament-Selektion,
- Stochastic-Universal-Sampling

werden folgend kurz beschrieben.

Der einfachste Mechanismus **Truncation-Selektion** stammt von Holland (1975). Hierbei wird ein vorher festgelegter Anteil der besten Chromosomen in die nächste Generation übernommen und für die Selektion bestimmt. Die übrigen, schlechtesten Chromosomen werden nicht in die nächste Generation übernommen.

Ein weiterer Selektionsmechanismus ist die **Proportionate-Selektion** bzw. die **Roulette-Wheel-Selektion**. Die Chromosomen werden proportional zu ihrem Fitness-Wert als Anteil an der Summe aller Fitness-Werte der Population ausgewählt. Ein Chromosom m wird selektiert mit der Wahrscheinlichkeit P_m (Goldberg/Deb 1991, S. 71):

$$P_m = \frac{f_m}{\sum_{m'=1}^{M} f_{m'}}. \tag{3.1}$$

Ein Mechanismus, um nicht die absolute Differenz zwischen den Fitness-Werten der Chromosomen in der Population zu berücksichtigen, sondern allein deren Rangfolge, bietet die **Ranking-Selektion** (Baker 1985, S. 102/103). Ziel dieses Mechanismus ist es, einzelne, aufgrund ihres Fitness-Werts dominierende Chromosomen in ihrer Reproduktionsfähigkeit zu hemmen, so dass diese Chromosomen nicht zu viel Nachwuchs erzeugen können und somit die Population vorschnell in eine bestimmte Richtung lenken. Es können unterschiedliche Varianten für die Gewichtung der Chromosomen infrage kommen, solange diese Gewichtungen zwei Regeln folgen: erstens, die Auswahlwahrscheinlichkeit eines Chromosoms steigt monoton mit seiner Position in der Population, wenn diese vom besten zum schlechtesten Fitness-Wert sortiert ist. Zweitens, es wird eine maximale Anzahl angegeben, bis zu dieser ein Chromosom für die Reproduktion ausgewählt werden darf.

Die **Tournament-Selektion** ist geeignet, um den Selektionsdruck innerhalb einer Population konstant hoch zu halten. Dabei werden eine bestimmte Anzahl von Chromosomen zufällig ausgewählt, wobei dasjenige Chromosom für die Reproduktion verwendet wird, das den höchsten Fitness-Wert besitzt (Goldberg/Deb 1991, S. 78). Meistens treten zwei Chromosomen gegeneinander an bis die gewünschte Anzahl an Chromosomen für die Reproduktion zur Verfügung steht.

Die Selektion via **Stochastic-Universal-Sampling** (Baker (1987)) hat die Fitness-Werte der Chromosomen als Grundlage. Es werden über der Summe der Fitness-Werte der Chromosomen der Population mit der Länge $\frac{1}{N}\sum_{m=1}^{M} f_m$, wobei N die Anzahl der Nachkommen darstellt, Intervalle erzeugt, die von einem zufälligen Startpunkt aus, der im ersten Intervall liegt, auf die verhältnismäßige Länge der Fitnesswerte der Chromosomen als Pointer zeigt. Diejenigen Produktlinien mit ihren Fitness-Werten, auf die dieser Pointer zeigt, werden für die Reproduktion selektiert. Der Vorteil des Stochastic-Universal-Sampling ist, dass es einen minimalen Spread, einen Bias von Null sowie ein optimales Laufzeitverhalten von $\mathcal{O}(N)$ besitzt (Baker 1987, S. 14/18).

Für eine ausführliche Beschreibung der Vor- und Nachteile dieser Verfahren sei z. B. auf Baker (1985; 1987), Goldberg/Deb (1991) oder Reeves (2003) verwiesen.

3.3.3 Reproduktionsmechanismen

Die Reproduktionsmechanismen werden durch verschiedene Crossover-Strategien beschrieben. In der vorliegenden Arbeit werden folgende Crossover-Strategien vorgestellt:

- One-Point-Crossover,
- Two-Point-Crossover,
- Uniform-Crossover.

Bei der Reproduktion von Chromosomen geht es darum, Gene eines Elternteils auszuwählen, die die zugehörigen Gene des anderen Elternteils ersetzen und umgekehren. Eine Möglichkeit hierfür ist der **Single-Point-Crossover**. Dabei wird zufällig ein Gen ausgewählt, ab dem die Gene zwischen den Elternteilen getauscht werden. Angenommen man hat zwei Elternteile (Chromosomen) der Form (Reeves 2003, S. 68)

$$(a_1, a_2, a_3, a_4, a_5, a_6) \quad \text{und} \quad (b_1, b_2, b_3, b_4, b_5, b_6)$$

und ein Zufallszahlengenerator gibt gleichverteilt eine Zahl zwischen zwei und sechs aus (ein Crossover ab Gen a_1 bzw. b_1 erzeugt nur wieder die alten Chromosomen), hier eine drei. Dann werden die Strings zwischen den Elternteilen ab Gen a_3 bzw. b_3 getauscht. Daraus entstehen zwei Nachkommen, die wie folgt aussehen:

$$(a_1, a_2, b_3, b_4, b_5, b_6) \quad \text{und} \quad (b_1, b_2, a_3, a_4, a_5, a_6)$$

Für den **Two-Point-Crossover** werden zwei zulässige Zufallszahlen erzeugt, um die Crossover-Punkte zu bestimmen. Angenommen diese Punkte bzw. Gene seien für das vorliegende Beispiel drei und fünf. Dann entstehen die folgenden Nachkommen:

$$(a_1, a_2, b_3, b_4, a_5, a_6) \quad \text{und} \quad (b_1, b_2, a_3, a_4, b_5, b_6)$$

Dieses Vorgehen kann bei entsprechender Chromosomenlänge beliebig auf mehr als zwei Crossover-Punkte erweitert werden. Die dritte Möglichkeit für Crossover stellt der **Uniform-Crossover**

dar. Hierbei wird jedes Gen mit einer Wahrscheinlichkeit von $P = 0.5$ zwischen den Elternteilen getauscht. Angenommen die Entscheidung fiel im Ausgangsbeispiel auf die Gene 1, 4 und 5. Dann ergeben sich diese Nachkommen:

$$(b_1, a_2, a_3, b_4, b_5, a_6) \quad \text{und} \quad (a_1, b_2, b_3, a_4, a_5, b_6)$$

In der vorliegenden Arbeit wird sich auf diese drei Crossover-Strategien beschränkt. Weitere Informationen und Crossover-Strategien können z. B. aus Reeves (2003) und Srinivas/Patnaik (1994) entnommen werden.

3.3.4 Mutationsmechanismen

Der Mutationsmechanismus ist keine Notwendigkeit für Genetische Algorithmen. Jedoch existieren kaum Implementierungen ohne Mutationsmechanismus. So konnte Vose (1994, S. 423) nachweisen, dass die Mutation eine wichtige Rolle für die Performance Genetischer Algorithmen spielt. Auch hier gibt es verschiedene Möglichkeiten, von denen zwei näher vorgestellt werden. Das theoretische Vorgehen ist, für jedes Gen eines Chromosoms mit einer kleinen Wahrscheinlichkeit P zu entscheiden, ob ein Gen mutieren soll oder nicht. Für die Wahl der Mutationsrate P gibt es verschiedene Möglichkeiten. Zum einen kann $P = const.$ gewählt werden. Zum anderen kann es in Abhängigkeit der Anzahl der Gene g bestimmt werden durch $P = \frac{1}{g}$. Diese beiden Mutationsmechanismen haben sich in der Praxis durchsetzen können, wobei sich vor allem die Variante in Abhängigkeit der Chromosomlänge bewiesen hat. Theoretisch würde also für jedes Gen eines Chromosoms in der Population eine Zufallszahl erzeugt werden, die mit P verglichen würde, um zu entscheiden, ob eine Mutation für dieses Gen stattfindet oder nicht. Da dies rechentechnisch sehr aufwändig ist, wird eine feste Anzahl von Chromosomen bestimmt, die in einem Gen, das zufällig gleichverteilt bestimmt wird, mutieren sollen (Reeves 2003, S. 70).

3.3.5 Populationswahrungsmechanismen

Die Funktion der Populationswahrungsmechanismen besteht darin, die Anzahl der Chromosomen für jede Generation einer Population konstant zu halten. Hierfür werden drei Mechanismen beschrieben (Balakrishnan/Jacob 1995, S. 322):

- Emigration-Mechanismus,

- Malthusian-Mechanismus,

- modifizierter Malthusian-Mechanismus.

Der **Emigration-Mechanismus** bestimmt die Population durch eine Populationsgröße M mit einer Anzahl S für die selektierten Chromosomen für die Reproduktion, die in $M - S$ Nachkommen münden. Beim **Malthusian-Mechanismus** werden zu der Populationsgröße M zunächst alle α erzeugten Nachkommen zu einer neuen Population der Größe $M + \alpha$ zusammengefasst, wobei die α Nachkommen mutiert werden. Anschließend werden diejenigen Chromosomen mit der geringsten Fitness aus der neu geformten Population entfernt, bis die ursprüngliche Größe von M wieder für die nächste Generation erreicht ist. Beim **modifizierten Malthusian-Mechanismus** werden zusätzlich zu den $M + \alpha$ noch die S Chromosomen für die Reproduktion zu einer Gesamtgröße von $M + \alpha + S$ zusammengefügt, wobei die $\alpha + S$ Nachkommen und deren Eltern dem Prozess der Mutation unterzogen werden. Der Vorteil besteht darin, dass die besten Chromosomen nochmals kopiert werden und somit die Population stärken. Der Nachteil ist der Verlust an Variabilität in der neuen Population. Nach diesem Prozess wird die Populationsgröße wieder auf die besten M Chromosomen geschrumpft.

Weiterhin ist es von Vorteil, einen Elite-Mechanismus (De Jong (1975)) zu berücksichtigen. Besitzen z. B. alle Nachkommen einer Population einen höheren Fitness-Wert als die Chromosomen, aus denen sie erzeugt wurden, werden diese Chromosomen nicht in die neue Generation überführt, obwohl es viel Aufwand bedeutete diese zu erzeugen und diese Chromosomen vielleicht in Regionen mit vielversprechenden Fitness-Werten liegen. Um diese Chromosomen nicht zu verlieren, kann eine bestimmte Anzahl der besten Chromosomen in die nächste Generation übernommen werden, auch wenn sie potentiell schlechtere Fitness-Werte haben als ihre Nachkommen (Reeves 2003, S. 71).

3.4 Particle-Swarm-Optimization

Die Particle-Swarm-Optimization wurde ursprünglich von Kennedy/Eberhart (1995b) für die Optimierung von stetigen, nicht-linearen Funktionen ersonnen. Particle-Swarm-Optimization hat seine Wurzeln im künstlichen Leben im Allgemeinen und in Vogel- und Fischschwärmen im Speziellen (Kennedy/Eberhart 1995b, S. 1942). Für die Entwicklung war die Hypothese entscheidend, dass der Austausch von Informationen in sozialen Gruppen einen evolutionären Vorteil bringt. Durch diesen Austausch von Informationen sind Vogel- oder Fischschwärme in der Lage, Nahrung zu suchen oder Raubtieren durch das Anpassen der Bewegungen an die des Schwarms zu entkommen. Diese Form der Schwarmintelligenz kann auf einfache Weise modelliert werden. Einzigartig an diesem Konzept ist die Fähigkeit, dass die einzelnen Partikel im

mehrdimensionalen Suchraum in die Richtung von Lösungen mit höherem Zielfunktionswert beschleunigen. Beispielhaft ist dies in Abbildung 2 illustriert. Anfangs verteilen sich die Partikel über den gesamten Suchraum („Iteration 1“ in Abbildung 2). Die Länge und Richtung der Pfeile der Partikel geben die Geschwindigkeit an, mit der sich die Partikel in eine bestimmte Richtung bewegen. Während des Optimierungsprozesses sammeln sich die Partikel in bestimmten Regionen des Suchraums, bis sie schließlich in „Iteration N“ in lokalen Optima zu finden sind. Im vorliegenden Beispiel hat ein Großteil der Partikel sogar das globale Optimum gefunden. Im Allgemeinen ist dies jedoch nicht sichergestellt.

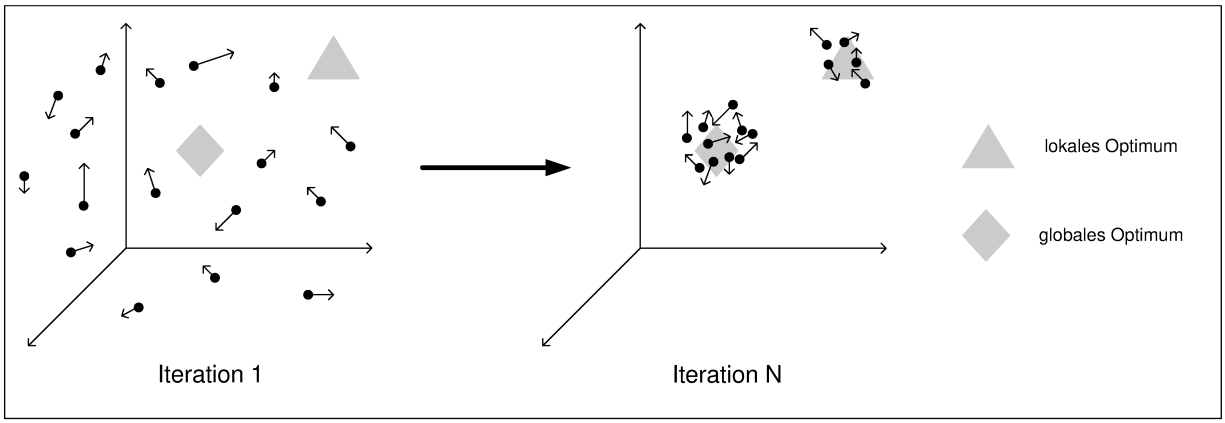

Abbildung 2: Verhalten der Partikel in der Particle-Swarm-Optimization. (Quelle: Eigene Darstellung in Anlehnung an Engelbrecht (2007, S.295/294)

Particle-Swarm-Optimization ist ein populationsbasierter Algorithmus, dessen Population aus mehreren Partikeln bzw. Individuen besteht. Diese Individuen entwickeln sich über Generationen hinweg, indem sie untereinander kooperieren und konkurrieren. Jedes Individuum passt seine Bewegungen im Suchraum auf Basis des eigenen Verhaltens sowie dem Verhalten der anderen Individuen an (Shi/Eberhart 1998, S. 69). Dabei repräsentiert ein Partikel bzw. ein Individuum eine zulässige Lösung des zugrundeliegenden Optimierungsproblems. Die folgende Beschreibung der Particle-Swarm-Optimization sowie die Variablenbezeichnungen lehnen sich an die Arbeiten von Kennedy/Eberhart (1995b), Shi/Eberhart (1998), Eberhart/Shi (2000) und vor allem Tsafarakis et al. (2011) an.

Für die Initialisierung der Population werden M Partikel in einem D-dimensionalen Raum zufällig erzeugt, wobei D die Anzahl der zu optimierenden Variablen darstellt. Jede Position jedes Teilchens stellt eine Lösung des Optimierungsproblems dar. Die Position eines Partikels wird angegeben durch einen D-dimensionalen Vektor $\mathbf{s}_i = (s_{i1}, s_{i2}, ..., s_{id}), i = 1,2,...,M, d = 1,2,...,D, \mathbf{s}_i \in \mathbb{R}^D$. Sobald jedes Partikel zufällig im D-dimensionalen Raum verteilt ist, werden die Zielfunktionswerte zu jedem Partikel berechnet. Für die Veränderung der Position eines Partikels wird für jedes Partikel $\mathbf{s}_i$ ein Geschwindigkeitsoperator $\mathbf{v}_i = (v_{i1}, v_{i2}, ..., v_{id}), i = 1,2,...,M, d = 1,2,...,D, \mathbf{v}_i \in \mathbb{R}^D$, eingeführt. Während des Optimierungsprozesses orientiert sich jedes Partikel an seiner bisher besten Position im Suchraum sowie an der besten Position des Partikels mit dem besten Zielfunktionswert der gesamten Population (siehe Gleichung

3.2). Danach werden die neuen Positionen der Partikel gemäß Gleichung 3.3 bestimmt. Je nach Modellierung des Algorithmus können sich die Partikel auch an der Position des lokal besten Partikels eines Teilschwarms orientieren, anstatt am besten Partikel des Gesamtschwarms. Jedes Partikel passt seinen Weg entsprechend seiner bisherigen besten Position p_{id} und der Position des besten Partikels p_d^g in der Population an. Die Geschwindigkeit und die Position eines Partikels werden in jeder Iteration t durch die folgenden Bildungsvorschriften wiedergegeben, wobei η_1 und η_2 Zufallsvariablen aus dem Intervall $[0,1]$ bezeichnen.

$$v_{id}(t+1) = v_{id}(t) + c_1\eta_1(p_{id} - s_{id}(t)) + c_2\eta_2(p_d^g - s_{id}(t)) \tag{3.2}$$

$$s_{id}(t+1) = s_{id}(t)v_{id}(t+1) \tag{3.3}$$

Die Koeffizienten c_1 und c_2 sind stochastische Beschleunigungskoeffizienten, die kontrollieren, wie weit und in welche Richtung sich ein Partikel während einer Iteration bewegt. Sie sind Gewichtungsfaktoren und werden üblicherweise auf den Wert 2 gesetzt (Eberhart/Shi 2001, S. 82). Je nach zugrunde liegendem Optimierungsproblem können andere Werte für c_1 und c_2 zu einer besseren Performance des Algorithmus führen (Tsafarakis et al. 2011, S. 15). Der Algorithmus wird abgebrochen, wenn er entweder konvergiert oder eine vorher definierte Anzahl von Iterationen durchlaufen wurde. Am Ende wird dasjenige Partikel mit dem höchsten Zielfunktionswert ausgegeben. Die Populationsgröße bleibt während des Optimierungsprozesses konstant.

3.4.1 Inertia weight

Ein Problem, dass dieser erste Standardalgorithmus hatte, ist der Exploration-Exploitation-Trade-off. Zunächst wurde dieses Problem angegangen, indem versucht wurde, einen Höchstwert für die maximale Geschwindigkeit der Partikel zu bestimmen. Das sollte verhindern, dass bei zu hoher Geschwindigkeit der Partikel die Fähigkeit der Exploration des Suchraums nachlässt, aber die Fähigkeit der lokalen Suche, also der Exploitation, verstärkt wurde. Somit wurde der Algorithmus zu häufig in lokalen Optima festgehalten, was am Ende zu schlechteren Lösungen führte (Eberhart/Shi 2001, S. 82). Um hierfür einen flexibleren Ansatz zu haben, der diesem Exploration-Exploitation-Trade-off durch eine höhere Anpassungsfähigkeit Rechnung trägt, führten Shi/Eberhart (1998, S. 70) ein sog. *inertia weight* für die Gewichtung der Geschwindigkeit eines Partikels der letzten Iteration ein. Dieses *inertia weight* w, das eine positive, reellwertige Konstante oder eine Funktion in Abhängigkeit der Iterationen sein kann, ergänzt die Gleichung 3.2, wie folgt:

$$v_{id}(t+1) = wv_{id}(t) + c_1\eta_1(p_{id} - s_{id}(t)) + c_2\eta_2(p_d^g - s_{id}(t)) \tag{3.4}$$

Die Autoren finden in ihrer Untersuchung heraus, dass für eine hohe Lösungsqualität ein *inertia weight* von $w \in [0.9, 1.2]$ zielführend ist (Shi/Eberhart 1998, S. 71). Eine andere Möglichkeit ist es, das *inertia weight w* dahingehend anzupassen, als dass es über die Iterationen abnimmt, um bestimmte Regionen des Suchraums näher zu untersuchen. Das kann auf folgendem Weg realisiert werden (Tsafarakis et al. 2011, S. 15):

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \cdot iter \tag{3.5}$$

Dabei bezeichnen w_{min} und w_{max} den kleinsten bzw. den größten Wert, den w annehmen darf. Die Variable $iter$ und der Parameter $iter_{max}$ bezeichnen dabei zum einen die aktuelle Iteration und die vorher definierte Höchstzahl an Iterationen für den Optimierungsprozess.

3.4.2 Constriction factor

Weitere Forschungsarbeiten von Clerc (1999) und Clerc/Kennedy (2002) auf diesem Gebiet zeigten, dass es nötig sein könnte, einen *constriction factor* χ in das Modell einzuführen, der die Konvergenz des Algorithmus sicherstellt. Dieser *constriction factor* χ ist nur eine Gewichtung der Geschwindigkeit der Partikel der vorhergehenden Iteration, sondern eine Gewichtung für die neu zu berechnende Geschwindigkeit der Partikel in der aktuellen Iteration. Die neue Geschwindigkeitsgleichung ergibt sich nun zu:

$$v_{id}(t+1) = \chi\left(v_{id}(t) + c_1\eta_1(p_{id} - s_{id}(t)) + c_2\eta_2(p_d^g - s_{id}(t))\right) \tag{3.6}$$

Durch eine Eigenwertanalyse wird der *constriction factor* χ mit folgender Formel angegeben:

$$\chi = \frac{2}{|2 - c - \sqrt{c^2 - 4c}|} \tag{3.7}$$

und der Parameter c zu:

$$c = c_1 + c_2 \quad \text{mit} \quad c > 4. \tag{3.8}$$

Eine wichtige Erweiterung des Aufbaus der Population bietet Engelbrecht (2007). Es geht um eine unterschiedliche räumliche Anordnung der Teilstrukturen der Population. Die oben beschriebene Populationstopologie, bezeichnet mit „GBest“ für den Austausch der Informationen zwischen dem besten Partikel in der Population und den anderen Partikeln, definiert als Nachbarschaft für alle in der Population befindlichen Partikel. Eine Erweiterung stellt die „LBest“-Populationstopologie dar, in der jedes Partikel eine kleinere, lokale Nachbarschaft besitzt, in der Informationen mit dem besten Partikel aus der lokalen Nachbarschaft ausgetauscht werden, aber nicht mehr mit dem besten Partikel der gesamten Population. Der Vorteil an der „LBest“-Populationstopologie liegt darin, dass es eine größere Diversität in der Population gibt, was zu einer besseren Exploration des gesamten Suchraums führt. Dafür konvergiert dieser Algorithmus langsamer als die Variante mit dem global besten Partikel, welche häufiger in lokalen Optima gefangen bleibt.

3.4.3 Diskretisierung der Operatoren

Um Kombinatorische und somit Diskrete Optimierungsprobleme mit der Particle-Swarm-Optimization lösen zu können, wurden mehrere Strategien entwickelt. Da der Standardalgorithmus für stetige Suchräume formuliert wurde, geben Kennedy/Eberhart (1997) eine Beschreibung für binäre Suchräume. Jedes Partikel bewegt sich in einem Zustandsraum, in dem die Variablen die Werte 0 oder 1 für jede Dimension annehmen können. Dabei repräsentiert die Geschwindigkeit v_{id} die Wahrscheinlichkeit, dass das Bit s_{id} den Wert 1 annimmt. Wenn die Geschwindigkeit hoch ist, ist die Wahrscheinlichkeit den Wert 1 für s_{id} zu erhalten höher. Umgekehrt ist die Wahrscheinlichkeit für $s_{id} = 0$ höher, wenn die Geschwindigkeit eines Partikels geringer ist. Um diesen Sachverhalt mathematisch modellieren zu können, wird eine Sigmoid-Funktion der Form:

$$\text{sig}(v_{id}) = \frac{1}{1 + \exp(-v_{id})} \tag{3.9}$$

verwendet, die den stetigen Zustandsraum in einen Wahrscheinlichkeitsraum transformiert (Tsafarakis et al. 2011, S. 16). In dieser Version werden die Geschwindigkeiten, je nach verwendeter Variante, gemäß der Gleichungen 3.2, 3.4 oder 3.6 geupdatet. Für das Positions-Update wird die folgende Gleichung verwendet:

$$s_{id}(t+1) = \begin{cases} 1, & \eta_3 < \text{sig}(v_{id}) \\ 0, & \text{sonst} \end{cases} \tag{3.10}$$

Dabei ist η_3 eine gleichverteilte Zufallsvariable aus dem Intervall $[0,1]$.

In diesem Abschnitt wurde die Funktionsweise der Particle-Swarm-Optimization mit ihren verschiedenen Varianten näher erläutert. Dargestellt wurden die Konzepte für die Gewichtung verschiedener Terme der Updategleichungen, wie *inertia weight* und *constriction factor*. Für die Erweiterung auf diskrete Optimierungsprobleme wurde eine mögliche Transformation vorgestellt.

3.5 Ant-Colony-Optimization

Das Lösen von kombinatorischen Optimierungsproblemen mittels Ameisenkolonien geht vor allem auf die Dissertation von Dorigo (1992) zurück. Als Koautor veröffentlichte er gemeinsam mit anderen Autoren eine erste Beschreibung der Optimierung mittels Ameisen (Colorni et al. 1991). Die Erforschung des Verhaltens von Ameisen auf der Nahrungssuche ebnete den Weg für die Entwicklung von Optimierungsverfahren auf Basis dieses Verhaltens. So stellte sich anfangs die Frage, wie Lebewesen, die (fast) blind sind, in der Lage sein können, den kürzesten Weg zu einer Nahrungsquelle und wieder zurück zu finden. Im Fall von Ameisen erledigen dies Pheromone, die die Ameisen auf ihrem Weg hinterlassen. So wird der Weg jeder einzelnen Ameise mit einer Pheromonspur markiert. Während sich eine isolierte Ameise hauptsächlich auf zufälligen Pfaden bewegt, kann eine Ameise, die eine Pheromonspur einer anderen Ameise entdeckt, dieser mit einer hohen Wahrscheinlichkeit folgen, wodurch die Pheromonspur weiter verstärkt wird, da diese Ameise ihrerseits ebenfalls Pheromone auf diesem Weg absondert. Dieses kollektive Verhalten führt dazu, dass umso mehr Pheromone auf einem Weg liegen, desto mehr Ameisen diesem folgen, wodurch wiederum mehr Pheromone auf diesem Weg liegen, weshalb weitere Ameisen folgen. Durch diese Rückkopplungsschleife, in der jede einzelne Ameise einem Pfad mit steigender Wahrscheinlichkeit folgt, der bereits viele Pheromone besitzt, können die Ameisen, wie in Abbildung 3 illustriert, den kürzesten Weg nach und nach von ihrem Nest zur Nahrungsquelle finden. (Colorni et al. 1991, S. 134)

Um diesen Vorgang zu modellieren, wurde das erste Ant-System entwickelt, auf dem alle weiteren Ant-Algorithmen aufbauen. Es handelt sich um einen populationsbasierten Optimierungsansatz, da eine Gruppe von Ameisen durch Informationsaustausch versucht, das Optimum des zugrunde liegenden Optimierungsproblems zu finden.

Unter dem Begriff Ant-Colony-Optimization werden verschiedene Algorithmen der gleichen Klasse zusammengefasst. Im Folgenden werden die Wichtigsten näher vorgestellt.

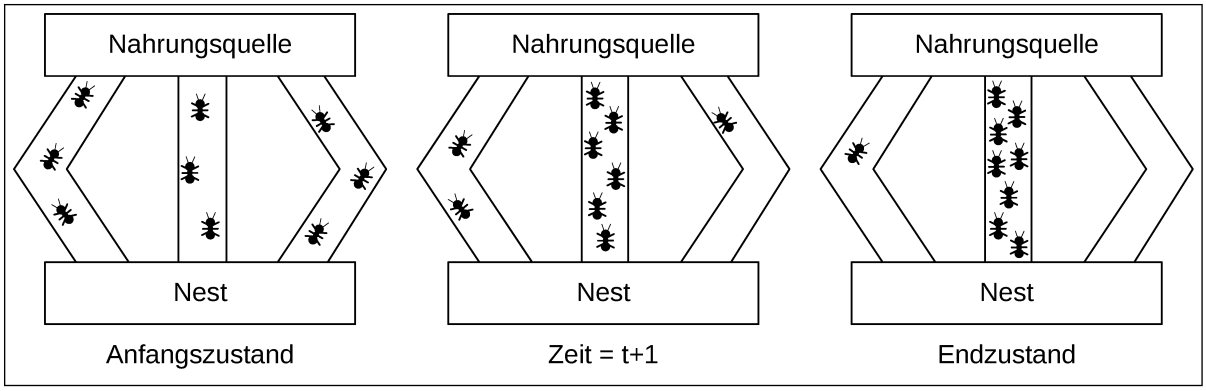

Abbildung 3: Verhalten der Ameisen bei der Nahrungssuche. Illustration der Rückkopplungsschleife zum Finden des kürzesten Weges zur Nahrungsquelle. (Quelle: Eigene Darstellung in Anlehnung an Dorigo et al. (1999, S. 139))

3.5.1 Ant-System

Das Ant-System ist eine erste Formulierung eines Ameisenalgorithmus von Colorni et al. (1991), Dorigo (1992). Der allgemeine Ablauf lässt sich wie folgt beschreiben. Für eine vorher festgelegte Anzahl von Iterationen wird eine bestimmte Anzahl von Ameisen festgelegt. In jeder Iteration entscheiden sich die Ameisen mit einer gewissen Wahrscheinlichkeit für einen zulässigen Wert der ersten Entscheidungsvariable. Ausgehend von dem Wert dieser ersten Entscheidungsvariable treffen die Ameisen wieder eine Entscheidung mit einer gewissen Wahrscheinlichkeit für einen Wert der zweiten Entscheidungsvariable und so weiter. Dieser Prozess stoppt, wenn jede Ameise eine vollständige, neue Lösung des zugrundeliegenden Optimierungsproblems konstruiert hat. Für die Anpassung dieser Wahrscheinlichkeiten dient die Pheromonmenge auf den entsprechenden Wegen. Je mehr Pheromone auf einem Weg liegen, desto größer ist die Wahrscheinlichkeit, dass sich eine Ameise für diesen entscheidet. Die platzierte Pheromonmenge hängt von den Zielfunktionswerten, der vollständigen Lösungen ab. Je besser dieser ist, desto mehr Pheromone werden auf die betreffenden Entscheidungsvariablen verteilt. Im Ant-System existieren drei unterschiedliche Vorgehensweisen, um die Pheromone auf den Wegen zu verteilen. Ohne auf die Einzelheiten einzugehen, sei erwähnt, dass die erfolgreichste Vorgehensweise davon das Platzieren von Pheromonen nach einer vollständigen Konstruktion einer Lösung ist und nicht schon während der Zwischenschritte, wenn lediglich Teillösungen vorliegen (Dorigo et al. 1999, S. 148). Ist eine Iteration beendet, sterben die Ameisen und eine neue Generation schwärmt in der nächsten Iteration wieder zur Nahrungssuche basierend auf den aktualisierten Wahrscheinlichkeiten für die Auswahl der Werte der Entscheidungsvariablen aus.

Nach dieser allgemeinen Beschreibung des Ant-Systems wird nun das zugehörige mathematische Modell beschrieben. Da sich alle Modellformulierungen in der Literatur beispielhaft auf das Problem des Handlungsreisenden beziehen, wird das Modell an dieser Stellen bereits an die Produktlinienoptimierungsmodelle mit den Bezeichnungen aus Abschnitt 2.2 adaptiert. Dabei

ist $k = 1,...,K$ die Anzahl der Eigenschaften eines Produkts und $l = 1,...,L_k$ bezeichnet die Anzahl der Eigenschaftsausprägungen, die zur Eigenschaft k gehören. Somit bezieht sich die Pheromonmenge $\tau_{kl}(t)$ von Eigenschaftsausprägung l der Eigenschaft k in Iteration t auf die gelernte Attraktivität, um Eigenschaftsausprägung l der Eigenschaft k auszuwählen. Die Stärke der Pheromonspur wird dabei während des Optimierungsprozesses angepasst, um die Erfahrung der Ameisen während der vergangenen Iterationen einfließen zu lassen. Die Ameisen platzieren dabei eine Pheromonmenge proportional zur Qualität der konstruierten Lösungen. Umso höher die Zielfunktionswerte der Produktlinienoptimierungsmodelle, desto mehr Pheromone werden auf der Eigenschaftsausprägung l der Eigenschaft k abgesondert. Durch dieses Vorgehen wird der Suchprozess in die Richtung von guten Lösungen gelenkt. Damit der Suchprozess nicht vorzeitig stagniert, können bereits platzierte Pheromone verdampfen (Evaporation, siehe Gleichung 3.12). Jede Ameise in der Population trifft ihre Entscheidung auf Basis der folgenden Übergangswahrscheinlichkeiten (Colorni et al. 1991, S. 137):

$$P_{kl}(t) = \frac{[\tau_{kl}(t)]^{\alpha}[\eta_{kl}]^{\beta}}{\sum_{l' \in \mathcal{N}_l} [\tau_{kl'}(t)]^{\alpha}[\eta_{kl'}]^{\beta}} \quad \forall l, l' \in \mathcal{N}_{kl} \tag{3.11}$$

Hierbei bezeichnet $\mathcal{N}_{kl}$ die Nachbarschaft der Eigenschaftsausprägung l innerhalb der Eigenschaft k. Diese Nachbarschaft besteht aus allen Eigenschaftsausprägungen einer Eigenschaft. Auf diesem Wege bahnen sich die Ameisen einen Weg durch die Eigenschaften und deren -ausprägungen bis sie eine vollständige Lösung des Problems (hier: Produktlinie) erzeugt haben (siehe Abbildung 4). Der Ausdruck η_{kl} beschreibt eine Information des zugrundeliegenden Optimierungsproblems, die über alle Iterationen gleich bleibt. Dies kann z. B. beim Problem des Handlungsreisenden die reziproke Entfernung zwischen zwei Städten sein oder bei der Produktlinienoptimierung die Teilstückdeckungsbeiträge jeder Eigenschaftsausprägung. Die Übergangswahrscheinlichkeiten P_{kl} kann entweder für jede Ameise in der Population angewendet werden oder es kann eine Elitestrategie implementiert werden, bei der nur diejenige Ameise mit dem höchsten Zielfunktionswert ein Update der Pheromone durchführen darf. Die Parameter α und β dienen zur Steuerung der relativen Wichtigkeit der Pheromonmenge und der Informationen des zugrundeliegenden Optimierungsproblems. Daher dienen diese Übergangswahrscheinlichkeiten als Tradeoff zwischen den stark genutzten Pfaden und der a priori Information des Optimierungsproblems. Die Wahl dieser Parameter ist problemabhängig und muss bestenfalls durch eine Sensitivitätsanalyse erfolgen.

Die Pheromonmenge $\tau_{kl}(t)$ der Eigenschaftsausprägung l der Eigenschaft k wird in jeder Iteration, wie folgt, geupdatet (Colorni et al. 1991, S. 136):

$$\tau_{kl}(t+1) = \rho\,\tau_{kl}(t) + \Delta\tau_{kl}(t) \tag{3.12}$$

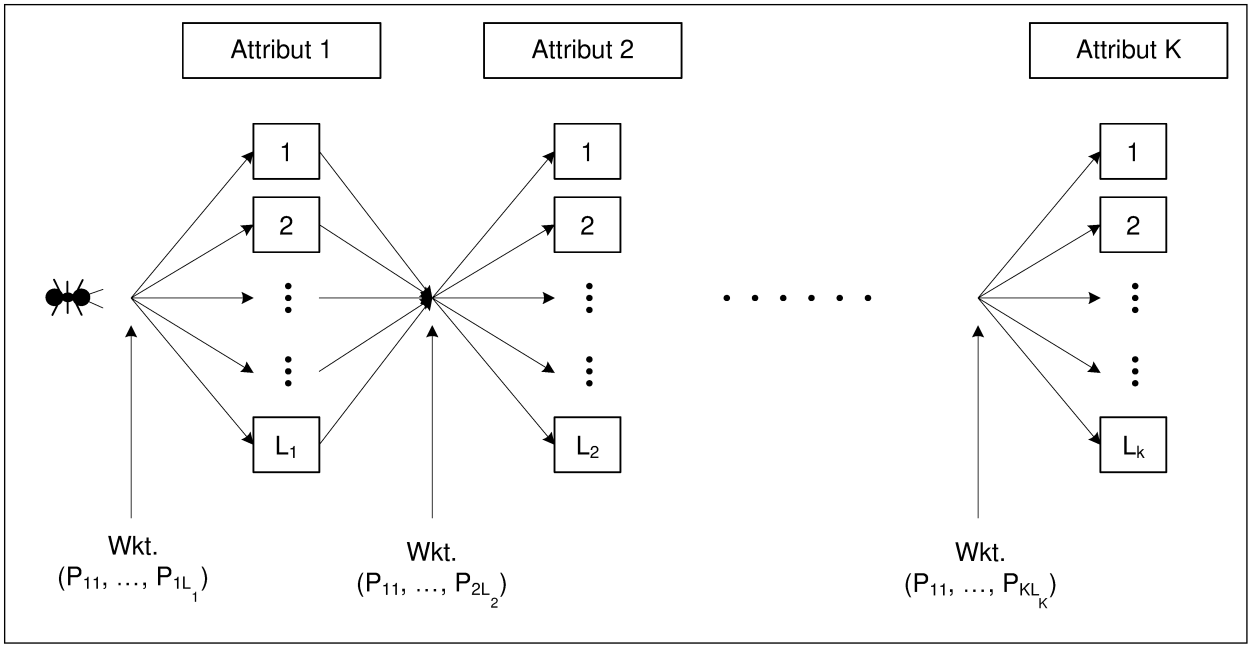

Abbildung 4: Veranschaulichung des Prinzips der Entscheidungen basierend auf Wahrscheinlichkeiten einer Ameise für eine Eigenschaftsausprägung einer Eigenschaft. (Quelle: Eigene Darstellung in Anlehnung an Albritton/McMullen (2007, S. 508))

Der Parameter ρ ist der Evaporationskoeffizient. Dieser wird verwendet, um die Pheromonmenge der Vergangenheit zu gewichten und den Updates in den aktuellen Iterationen ein größere Bedeutung zuzumessen. Es handelt sich um eine Art des langsamen Vergessens von früheren Pfaden. Die Änderung der Pheromonmenge $\Delta\tau_{kl}(t)$ in Iteration t bestimmt sich entsprechend (Dorigo et al. 1999, S. 149):

$$\Delta\tau_{kl}(t) = \begin{cases} Z(t), & \text{falls } (k,l) \in T(t) \\ 0, & \text{falls } (k,l) \notin T(t) \end{cases} \tag{3.13}$$

Dabei ist $\Delta\tau_{kl}(t)$ die Pheromonmenge, die entweder jede Ameise aus der Population in Iteration t auf der Eigenschaftsausprägung l der Eigenschaft k absondert oder nur die beste Ameise bei Verwendung einer Elitestrategie. $Z(t)$ ist der Zielfunktionswert der in Iteration t erzielt wurde. Da es sich bei der Produktlinienoptimierung um Maximierungsprobleme handelt, werden also diejenigen Eigenschaftsausprägungen mit einer größeren Menge Pheromon versehen, die zu einem höheren Zielfunktionswert führen. Für Minimierungsprobleme böte sich hierfür bspw. das Reziproke des Zielfunktionswertes an. $T(t)$ bezeichnet den genommenen Pfad, also die besuchten Eigenschaftsausprägungen $l = 1,...,L_k$ der Eigenschaften $k = 1,...,K$. Diejenigen Eigenschaftsausprägungen, die nicht zur Lösung gehören, bekommen den Wert $\Delta\tau_{kl}(t) = 0$ zugewiesen. Zur Initialisierung des Algorithmus ($t = 0$) wird auf jeder Eigenschaftsausprägung eine geringe Menge Pheromone verteilt (Colorni et al. 1991, S. 137).

Das Ant-System bot zwar ein neues Framework für die Optimierung, war jedoch für das Problem des Handlungsreisenden gegenüber anderen Algorithmen noch deutlich unterlegen (Dorigo/Stützle 2003, S. 262). Dies führte zu einer Reihe von Weiterentwicklungen auf diesem Gebiet.

3.5.2 Weiterentwicklungen des Ant-Systems

In diesem Abschnitt werden die wichtigsten Weiterentwicklungen des Ant-Systems kurz umrissen.

Da das Ant-System, mit dem Problem des Handlungsreisenden als Benchmark, zwar für kleinere Probleminstanzen gut als Heuristik geeignet ist, bei größeren Probleminstanzen jedoch nicht an die Performance anderer Heuristiken herankommt, wurden Abwandlungen entwickelt, die den Einsatz des Ant-Systems auch für größere Probleme möglich machen sollte (Dorigo et al. 1999, S. 150).

Eine Weiterentwicklung des Ant-Systems bietet bspw. das Ant-Colony-System (Gambardella/Dorigo 1996). Das Ant-Colony-System entwickelte sich aus dem Ant-Q-Algorithmus von Gambardella/Dorigo (1995), der eine Verbindung zwischen der Ant-Colony-Optimization und dem Reinforcement-Learning schaffen sollte. Es stellte sich heraus, dass die gleiche Performance mit deutlich einfacheren Pheromonupdateregeln des Ant-Colony-Systems erreicht werden konnten (Dorigo et al. 1999, S. 153). Das Ant-Colony-System verbessert das Ant-System, indem es der Exploitation der Informationen von Ameisen aus früheren Iterationen in Bezug auf die Exploration des Suchraums eine höhere Bedeutung zumisst (Dorigo/Stützle 2003, S. 264). Dies wird erreicht durch lokale und globale Updates für die Pheromonmenge $\Delta\tau_{kl}(t)$ (Gambardella/Dorigo 1996, S. 623). Das lokale Update der Pheromonmenge τ_{kl}^{loc} funktioniert, wie folgt:

$$\tau_{kl}^{loc}(t+1) = (1-\rho)\cdot\tau_{kl}(t) + \rho\cdot\tau_0 \tag{3.14}$$

Wobei $\rho \in (0,1]$ ein Evaporationskoeffizient und τ_0 ein Parameter ist. In einem nachfolgenden Schritt werden die Pheromonmengen einem weiteren Update unterzogen:

$$\tau_{kl}(t+1) = (1-\alpha)\cdot\tau_{kl}^{loc}(t+1) + \alpha\cdot Z_{best} \tag{3.15}$$

Der Parameter $\alpha \in (0,1]$ gewichtet die Pheromonmenge nach dem lokalen Update sowie dem

besten bisherigen Zielfunktionswert Z_{best}. Der Zweck hierfür ist diejenigen Eigenschaftsausprägungen zu verstärken, die Teil der bisher besten gefundenen Lösung sind. Grundsätzlich kann dieses Update sowohl von der global besten Ameise über alle Iterationen hinweg oder über die Ameise, die in einer einzelnen Iteration die beste Lösung fand, angewendet werden. Die lokalen Updates werden für jede Eigenschaftsausprägung während der Lösungskonstruktion durchgeführt. Das globale Update wird dahingegen erst ausgeführt, sobald die Ameisen eine vollständige Lösung (hier: Produktlinie) erzeugt haben (Gambardella/Dorigo 1996, S. 624).

Eine weitere Abwandlung besteht in dem **MIN-MAX Ant-System** von Stützle/Hoos (1997). Als direkte Weiterentwicklung zum Ant-System, das bei einer Elitestrategie lediglich Updates der Pheromonmenge der besten Ameise über alle Iterationen hinweg zulässt, hat die Eigenschaft schnell zu stagnieren, was weitere Verbesserungen des Zielfunktionswerts kaum ermöglicht (Stützle/Hoos 1997, S. 310). Die Pheromonmenge ist dann so hoch auf den Eigenschaftsausprägungen, dass die Ameisen die gleiche Produktlinie immer und immer wieder erzeugen würden. Um den Vorteil der Elitestrategie der besten Ameise zu nutzen und dem Nachteil der frühzeitigen Stagnation entgegenzuwirken, werden minimale und maximale Pheromonmengen für jede Eigenschaftsausprägung zugelassen, damit jede Eigenschaftsausprägung weiterhin die Chance hat, Teil der Lösung zu sein und so ggf. den Zielfunktionswert zu verbessern, da andere Regionen des Lösungsraum identifiziert werden können. Dieses Vorgehen führt dazu, dass die Auswahlwahrscheinlichkeit der Variablen, die zu schlechten Zielfunktionswerten führen, nie ganz Null wird, wodurch eine bessere Explorationsfähigkeit gegeben ist und insgesamt zu besseren Zielfunktionswerten führt. Die minimale und die maximale Pheromonmenge werden problemabhängig bestimmt. Für die maximale Pheromonmenge τ_{max} wird der bisher höchste Zielfunktionswert Z_{best} (bei Maximierungsproblemen) mit der Anzahl der Entscheidungsvariablen n multipliziert, was zu folgender Bildungsvorschrift führt (Stützle/Hoos 1998, S. 246):

$$\tau_{max}(t) = n \cdot Z_{best} \tag{3.16}$$

Die minimale Pheromonmenge bestimmt sich zu:

$$\tau_{min}(t) = \frac{a}{\Phi \cdot n^2} \tag{3.17}$$

Wobei a ein nicht näher spezifizierter konstanter Wert ist und Φ die durchschnittlichen Zielfunktionswerte repräsentiert. Diese beiden Maßnahmen führen dazu, dass die Explorationsfähigkeit des Algorithmus deutlich verbessert und somit die Performance gegenüber dem ursprünglichen Ant-System verbessert werden konnte (Stützle/Hoos (1997; 1998)). Beim Fortschreiten des Algorithmus nähern sich die minimale und die maximale Pheromonmenge langsam an, bleiben aber dennoch weit voneinander entfernt. Da sich die Beschreibung der Pheromonmengen auf

Minimierungsprobleme bezieht, wird der Zielfunktionswert Z_{best} im Verlauf immer kleiner und die durchschnittlichen Zielfunktionswerte ebenso, wodurch $\tau_{max}(t)$ kleiner und $\tau_{min}(t)$ größer werden. Für die Initialisierung der Pheromonmengen werden hohe Werte benötigt, damit sich der Algorithmus langsam entwickeln und vielversprechende Regionen im Suchraum finden kann (Stützle/Hoos 2000, S. 900). Die Höhe dieser Werte ist problemspezifisch und benötigt Wissen über das zugrunde liegende Optimierungsproblem.

Eine Übersicht mit weiteren Abwandlungen des Ant-Systems für verschiedene Optimierungsprobleme findet sich z. B. in Dorigo et al. (1999), Dorigo/Stützle (2003) oder Dorigo/Stützle (2010).

3.6 Simulated Annealing

Simulated Annealing („Simuliertes Abkühlen“) wurde von Kirkpatrick et al. (1983) als Verfahren formuliert, das für das Lösen von kombinatorischen Optimierungsproblemen Methoden aus der statistischen Mechanik verwendet (vgl. Kirkpatrick et al. 1983, S. 671). Inspiriert wurden die Autoren von dem Abkühlprozess kondensierter Materie. Der Begriff *Annealing* beschreibt einen thermischen Prozess, um einen niedrigen Energiezustand von Materie in einem Wärmebad zu erzeugen (vgl. Aarts/Korst 1989, S. 13). Dieser Prozess besteht aus zwei Schritten. Im ersten Schritt wird ein Festkörper so weit erhitzt, bis er schmilzt. Anschließend wird im zweiten Schritt die Temperatur sehr langsam abgesenkt, bis sich die Atome im Grundzustand befinden (vgl. Kirkpatrick et al. 1983, S. 672). Während der Festkörper eine flüssige Form besitzt, organisieren sich die Atome zufällig, wohingegen sie sich im Grundzustand in einem strukturierten Gitter anordnen, in dem die Energie des physikalischen Systems minimal ist. Dieser Grundzustand kann nur dann erreicht werden, wenn die Temperatur anfangs hoch genug und anschließend langsam genug verringert wird. Ist eine der beiden Bedingungen nicht erfüllt, nimmt der Festkörper einen nicht energieminimalen, sondern einen metastabilen Zustand an (vgl. Aarts/Korst 1989, S. 13). Der physikalische Abkühlprozess kann simuliert werden, indem die Evolution des Festkörpers algorithmisch nachgebildet werden kann, bis ein thermisches Gleichgewicht erreicht ist, was von Metropolis et al. (1953) gezeigt wurde. Mittels eines Monte-Carlo-Verfahrens werden zufällig mehrere Systemkonfigurationen erzeugt, die jeweils vom vorherigen Zustand abhängen und durch einen geeigneten Mechanismus in einen Folgezustand umgewandelt werden. Hierfür wird für einen Festkörper mit einer Energie E_t im momentanen Zustand t ein neuer Zustand $t+1$ mit Energie E_{t+1} hergestellt, der den alten Zustand nur wenig verändert, indem die Systemkonfigurationen nur minimal angepasst werden, z.B. durch das Ändern der Position eines Atoms im Festkörper. Die daraus resultierende Energiedifferenz ΔE ergibt sich somit zu:

$$\Delta E = E_{t+1} - E_t \tag{3.18}$$

Ziel ist es, die Energie des Systems zu minimieren. Ist die Energiedifferenz $\Delta E \leq 0$, so besitzt der Folgezustand E_{t+1} eine niedrigere (die gleiche) Energie als der vorherige Zustand und wird somit als neuer Zustand akzeptiert. Ist die Energiedifferenz $\Delta E > 0$, so wird der neue Zustand $t+1$ mit einer bestimmten Wahrscheinlichkeit akzeptiert (vgl. Kirkpatrick et al. 1983, S. 672; Aarts/Korst 1989, S. 14):

$$\exp\left(\frac{E_t - E_{t+1}}{k_B T}\right) \tag{3.19}$$

Wobei T die Temperatur des Wärmebades und k_B die Boltzmann-Konstante repräsentiert. Diese Akzeptanzregel wird *Metropolis-Kriterium* genannt. Der darauf basierende Algorithmus heißt Metropolis-Algorithmus. Wenn die Temperatur langsam genug gesenkt wird, erreicht der Festkörper ein thermisches Gleichgewicht für jede Temperatur. Der Metropolis-Algorithmus kann angewendet werden, um eine Folge von Lösungen eines kombinatorischen Optimierungsproblems zu erzeugen. Die daraus resultierende Analogie zwischen einem physikalischen System und einem kombinatorischen Optimierungsproblem basiert auf folgenden Annahmen (vgl. Aarts/Korst 1989, S. 15):

- Die Lösungen eines kombinatorischen Optimierungsproblems entsprechen dem Zustand eines physikalischen Systems.

- Der Zielfunktionswert einer Lösung entspricht der Energie eines Zustands.

Ein Kontrollparameter c wird eingeführt, um die Temperatur eines Systems zu symbolisieren. Simulated Annealing kann nun als eine Iteration des Metropolis-Algorithmus angesehen werden, welcher für die abnehmenden Werte des Kontrollparameters evaluiert wird. Für die folgenden Beschreibungen von Simulated Annealing wird die Nachbarschaftsdefinition aus Kapitel 3 zugrunde gelegt. Für eine zu minimierende Funktion mit Z_t und Z_{t+1} für zwei aufeinander folgende Zielfunktionswerte in den Iterationen t und $t+1$ ergibt sich das Akzeptanzkriterium aus der folgenden Akzeptanzwahrscheinlichkeit (vgl. Aarts/Korst 1989, S. 15):

$$P\{\text{akzeptiere Zustand } t+1\} = \begin{cases} 1 & \text{wenn } Z_{t+1} \leq Z_t, \\ \exp\left(\frac{Z_t - Z_{t+1}}{c}\right) & \text{wenn } Z_{t+1} > Z_t. \end{cases} \tag{3.20}$$

Der Parameter $c \in \mathbb{R}^+$ ist der Kontrollparameter. Die Akzeptanzwahrscheinlichkeit sorgt dafür, dass nicht nur Lösungen angenommen werden, die zu einer Verbesserung des Zielfunktionswerts führen ($Z_{t+1} \leq Z_t$), sondern es werden zum Teil Lösungen akzeptiert, die kurzfristig zu schlechteren Zielfunktionswerten führen können ($Z_{t+1} > Z_t$), falls die Wahrscheinlichkeit $\exp(\frac{Z_t - Z_{t+1}}{c}) > unif[0,1)$ ist, wobei $unif[0,1)$ eine gleichverteilte Zufallszahl charakterisiert, die im Intervall $[0,1)$ liegt. Der Vorteil dieses Mechanismus ist, dass Simulated Annealing hierdurch die Fähigkeit besitzt, aus lokalen Optima zu entkommen und weitere, vielversprechende Bereiche des Suchraums untersuchen kann, was die Chance erhöht, das globale Optimum zu finden (siehe Abbildung 5).

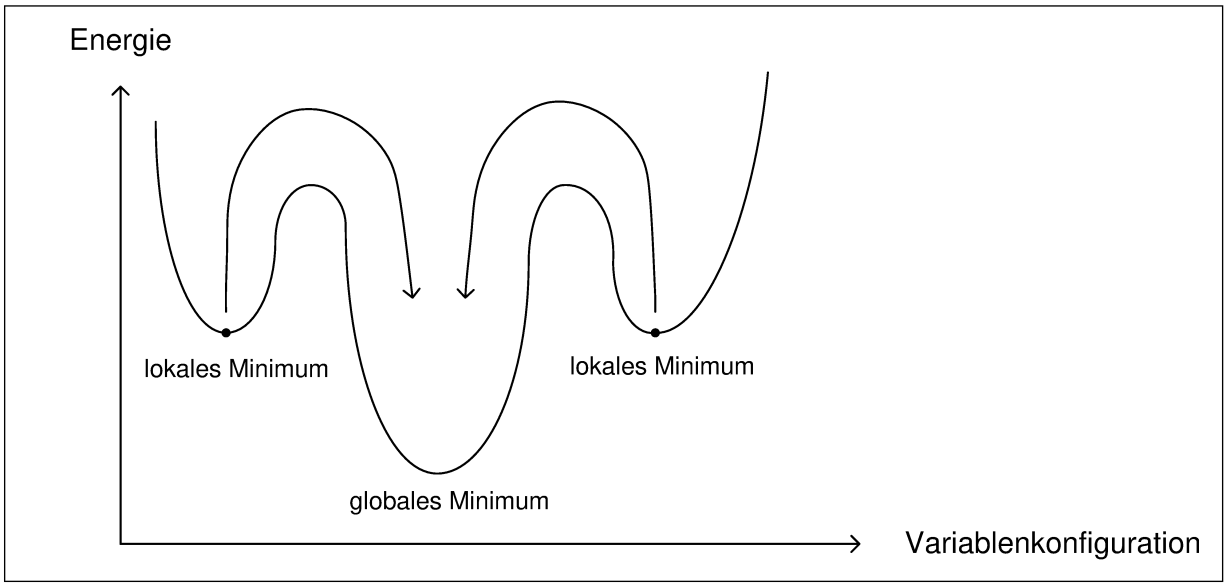

Abbildung 5: Entkommen aus lokalen Minima durch Sprünge induziert durch eine stochastische Akzeptanzregel für die gefundenen Lösungen von Simulated Annealing. (Quelle: Eigene Darstellung in Anlehnung an Ghasemalizadeh et al. (2016, S. 1671))

3.6.1 Cooling-Schedule

In diesem Abschnitt wird die Bedeutung des Cooling-Schedule im Zusammenhang mit dem Kontrollparameter c beschrieben.

Der Kontrollparameter c, der das Abkühlen symbolisiert, besitzt anfangs einen hohen Startwert. Ein hoher Wert von c bedeutet, dass anfangs viele schlechtere Lösungen akzeptiert werden, da die Wahrscheinlichkeit bzw. die Akzeptanzrate, dass ein neuer Zustand akzeptiert wird, noch relativ groß ist. Diese Akzeptanzrate verringert sich während des Suchprozesses über die Iterationen hinweg, bis (fast) keine schlechteren Lösungen mehr akzeptiert werden, sobald c gegen 0 geht. Simulated Annealing konvergiert asymptotisch gegen das globale Optimum. Da es in der praktischen Anwendung des Algorithmus jedoch nicht möglich ist, die asymptotische Konvergenz in jedem Fall zu erreichen, handelt es sich um eine Heuristik (vgl. Aarts et al. 1988, S.

189/190). Für die praktische Anwendung von Simulated Annealing auf konkrete Optimierungsprobleme muss also ein endlicher **Cooling-Schedule** derart konstruiert werden, dass zum einen die Eigenschaft des Entkommens aus lokalen Optima bestehen bleibt und zum anderen, dass eine lokale Suche gegen Ende des Algorithmus gewährleistet wird. Falls die Werte des Kontrollparameters c zu groß gewählt werden, wird die Wahrscheinlichkeit so groß, dass schlechtere Lösungen akzeptiert werden können und entspricht somit eher einer einfachen Zufallssuche, in welcher fast alle neuen Zielfunktionswerte akzeptiert werden. Ist der Kontrollparameter c anfangs hingegen zu klein, entsteht von Anfang an eine lokale Suche, in der nur noch Zielfunktionswerte akzeptiert werden, die einen besseren Wert besitzen, wodurch der Algorithmus frühzeitig in ein lokales Optimum konvergieren würde, da der Lösungsraum nicht adäquat abgesucht werden kann. Daher muss vor allem der Startwert des Cooling-Schedule entsprechend austariert werden. Um einen endlichen Algorithmus zu implementieren, der gegen ein (lokales) Optimum konvergiert, ist es nötig, eine homogene Markov-Kette von endlicher Länge für eine Folge von abnehmenden Werten des Kontrollparameters c zu generieren. Hierfür werden für den Cooling-Schedule die folgenden Konventionen getroffen (vgl. Aarts/Korst 1989, S. 57):

- Eine endliche Folge von Werten für den Kontrollparameter c, d. h.
  - einen Startwert c_0 des Kontrollparameters c,
  - eine monoton fallende Funktion für die Werte des Kontrollparameters c,
  - einen Endwert des Kontrollparameters c, der von einem Abbruchkriterium abhängig ist.
- Eine endliche Zahl von Zustandsänderungen für jeden Wert des Kontrollparameters c, d.h. eine endliche Länge jeder homogenen Markov-Kette.

Auf Grundlage dieser Konventionen können für den Kontrollparameter c die im nächsten Abschnitt aufgeführten Werte hergeleitet werden.

3.6.2 Kontrollparameter

Aufgrund dieser Definition des Cooling-Schedules muss der Kontrollparameter c für die praktische Implementierung wie folgt angepasst werden. Die Länge einer Markov-Kette des Cooling-Schedules für einen konkreten Wert c_m wird mit G_m bezeichnet. Der Startwert des Kontrollparameters c_0 muss derart gewählt werden, dass anfangs nahezu alle neuen Lösungen akzeptiert werden. Das kann erreicht werden, indem die Akzeptanzrate, also die Wahrscheinlichkeit, dass

ein neuer Zustand akzeptiert wird, nahe 1 liegt. Für den Kontrollparameter c_m kann z.B. die monoton fallende Folge

$$c_{m+1} = \alpha \cdot c_m, \quad m = 1,2,..., \tag{3.21}$$

verwendet werden, wobei α eine Konstante kleiner als 1 ist. Typische Werte für α liegen im Bereich $[0.8, 0.99]$ (vgl. Aarts/Korst 1989, S. 59). Der Endwert c_{end} des Cooling-Schedules wird als derjenige Wert definiert, für den nach einer gewissen Länge der Markov-Kette keine Verbesserung des Zielfunktionswerts mehr erreicht werden kann. Aarts/Korst (1989, S. 75) zeigen, dass die Wahl des Cooling-Schedules nicht entscheidend für die Performance und die Laufzeit von Simulated Annealing ist, solange die den Cooling-Schedule determinierenden Parameter vorsichtig gewählt werden. Aus diesem Grund wird auf eine ausführliche Auseinandersetzung mit verschiedenen Cooling-Schedules verzichtet und im Folgenden dargestellt, wie eine vorsichtige Wahl der Parameter vorgenommen werden kann. Hierzu wird der theoretischen Herleitung der Parameter für Simulated Annealing von Aarts/Korst (1989) gefolgt, die diese Größen aus den Eigenschaften der Statistischen Physik, bzw. der Statistischen Mechanik im Speziellen, herleiten. Den Überlegungen liegt zugrunde, dass das thermodynamische Gleichgewicht eines Systems (hier: kondensierende Materie) einer Boltzmannverteilung folgt, die die Wahrscheinlichkeit angibt, dass sich ein System im Zustand t mit der Energie E_t befindet. Durch diese Analogie lassen sich die statistischen Eigenschaften eines thermodynamischen Systems mittels der Theorie Stochastischer Prozesse auf Simulated Annealing übertragen. Der Kern der Herleitung ist, dass gezeigt werden kann, dass wenn die stationäre Verteilung der Boltzmannverteilung für jeden Wert des Kontrollparameters c erreicht wird, Simulated Annealing gegen das globale Optimum asymptotisch konvergiert (vgl. Aarts/Korst 1989, S. 19). Der Vorteil an diesem Vorgehen liegt in der Flexibilität nicht unbedingt vom problemspezifischen Erfahrungswissen für einen geeigneten Startwert für die Temperatur abhängig sein zu müssen, so wie es bspw. Kirkpatrick et al. (1983, S. 675) noch waren. Nicht nur für jedes andere kombinatorische Optimierungsproblem wäre ein Probieren eines geeigneten Startwerts nötig, sondern sogar für jede neue Probleminstanz eines speziellen kombinatorischen Optimierungsproblems, wenn sich die Zielfunktionswerte etwas stärker ändern, da die Akzeptanzwahrscheinlichkeiten von der Differenz der Zielfunktionswerte abhängen, weshalb der Kontrollparameter c jedes Mal entsprechend angepasst werden müsste.

Aus diesem generellen Problem erwächst die Notwendigkeit den Kontrollparameter c so einzustellen, dass eine initiale Akzeptanzrate χ_0 von neuen Zuständen nah an 1 gewährleistet werden muss. Aarts/Van Laarhoven (1985), Aarts/Korst (1989) bieten hierfür ein variables Framework zum Einstellen der Parameter, das genau dies ermöglicht. Bezeichne der Parameter s_1 die Anzahl der Zustände, für die $Z(t+1) \leq Z(t)$ und s_2 bezeichne die Anzahl der Zustände, für die $Z(t+1) > Z(t)$ gilt. $\overline{\Delta Z}^{(+)}$ sei das arithmetische Mittel der Differenz der Zielfunktionswerte

für die $Z(t+1) > Z(t)$ gilt. Dann kann die Akzeptanzrate eines neuen Zustands, wie folgt, approximiert werden (vgl. Aarts/Korst 1989, S. 60):

$$\chi \approx \frac{s_1 + s_2 \cdot \exp(\frac{-\overline{\Delta Z}^{(+)}}{c})}{s_1 + s_2} \tag{3.22}$$

Aus dieser allgemeinen Formulierung ergibt sich der Startwert c_0 für den Cooling-Schedule bei einer angenommenen Akzeptanzrate von χ_0 durch:

$$c_0 = \frac{\overline{\Delta Z}^{(+)}}{\ln(\frac{s_2}{s_2 \cdot \chi_0 - s_1(1-\chi_0)})} \tag{3.23}$$

Der Parameter χ_0 ist die avisierte Akzeptanzrate für schlechtere Zielfunktionswerte für den Start des Algorithmus. Parameter c_0 kann bestimmt werden, indem eine Folge von $s_0 = s_1 + s_2$ Lösungen generiert wird, aus denen c_0 über die Gleichung 3.23 berechnet werden kann. Eine geeignete Akzeptanzrate stellt dabei z.B. $\chi_0 = 0.95$ dar (vgl. Aarts/Korst 1989, S. 70). Ist ein Startwert für den Kontrollparameter c gefunden worden, lässt sich die Temperatur c_{m+1} in Abhängigkeit eines Parameters δ und der Standardabweichung der Zielfunktionswerte σ_{c_m} bestimmen. Dann ergibt sich c_{m+1} mit $\alpha = \frac{1}{1+\frac{c_m \cdot \ln(1+\delta)}{3\sigma_{c_m}}}$ aus 3.24 zu:

$$c_{m+1} = \frac{c_m}{1 + \frac{c_m \cdot \ln(1+\delta)}{3\sigma_{c_m}}}, \quad m = 0, 1, ..., \tag{3.24}$$

Der als δ bezeichnete *Distanzparameter* hat seinen Ursprung in der Überlegung, dass die m-te Markov-Kette der Lösungen mit einer bestimmten Länge nach einer bestimmten Anzahl m gegen ihre stationäre Verteilung konvergiert. Die Schrittweite, in der der Parameter c_m verringert wird, hängt also vor allem vom Distanzparameter δ ab. Dabei führen kleinere Werte von δ zu kleineren Verringerungen in c_m und größere Werte von δ zu größeren Verringerungen in c_m. Eine geeignete Wahl des Parameters δ ist $\delta = 0.1$, welcher in verschiedenen Simulationsrechnungen empirisch bestimmt wurde (vgl. Aarts/Korst 1989, S. 72). Da es in dieser Arbeit nicht um die technischen Details von Simulated Annealing gehen soll, wird auf die umfangreiche Herleitung des Wertes α und c_m auf Aarts/Korst (1989, S. 61 ff.) verwiesen.

Abschließend muss die Länge der Markov-Kette G_m für jede Temperatur c_m bestimmt werden. Diese wird auf die Anzahl der möglichen Nachbarschaftslösungen $\forall m$ gesetzt (vgl. Aarts/Korst 1989, S. 65).

3.6.3 Abbruchkriterium

Auf Basis des Konvergenzkriteriums, dass die Folge der Markov-Ketten gegen ihre stationäre Verteilung konvergieren, lässt sich ein Abbruchkriterium herleiten, bei welchem Simulated Annealing terminiert wird. Sei $\mathbb{E}_c(Z)$ der Erwartungswert der Zielfunktionswerte bzgl. des Kontrollparameters c im Gleichgewichtszustand (die Markovkette konvergiert gegen ihre stationäre Verteilung, ergo: $c_k \downarrow 0$). Weiterhin bezeichne Z_{opt} das globale Optimum des Optimierungsproblems. Des Weiteren bezeichne

$$\Delta\mathbb{E}_c(Z) = \mathbb{E}_c(Z) - Z_{opt} \tag{3.25}$$

die Differenz des Erwartungswerts der Zielfunktionswerte im Gleichgewichtszustand vom tatsächlichen Optimum. Der Algorithmus wird in dem Moment terminiert, in dem $\Delta\mathbb{E}(Z_c)$ sehr klein ggü. dem Erwartungswert der Zielfunktionswerte in einem früheren Zustand ist. Der Erwartungswert der Zielfunktionswerte früherer Zustände (große Werte für c), die sich noch nicht im Gleichgewichtszustand befinden, entsprechen approximativ dem arithmetischen Mittel aller zulässigen Zielfunktionswerte. Sei $\mathbb{S}$ die Menge aller zulässigen Lösungen, dann ist:

$$\lim_{c\to\infty} \mathbb{E}_c(Z) = \frac{1}{|\mathbb{S}|} \sum_{t\in\mathbb{S}} Z(t). \tag{3.26}$$

Das Optimum eines Optimierungsproblems Z_{opt} ist im Allgemeinen nicht bekannt, weshalb es durch einen anderen Wert angenähert werden muss. Daraus folgt, dass der Erwartungswert im Gleichgewichtszustand approximiert werden kann durch

$$\Delta\mathbb{E}_c(Z) \approx c\frac{\partial\mathbb{E}_c(Z)}{\partial c}, \tag{3.27}$$

falls $c \ll 1$ (vgl. Aarts/Korst 1989, S. 64) gilt. Die Veränderung im Erwartungswert der Zielfunktionswerte im Gleichgewichtszustand lässt sich durch eine Multiplikation des Kontrollparameters c und der ersten partiellen Ableitung des Erwartungswerts im Gleichgewichtszustand nach c annähern, wobei sich $\frac{\partial\mathbb{E}_c(Z)}{\partial c} = \frac{\sigma_c^2}{c^2}$ ergibt und $\sigma_c^2 = Var_c(Z)$ die Varianz im Gleichgewichtszustand darstellt (vgl. Aarts/Korst 1989, S. 20). Somit kann der Algorithmus schlussendlich für einen Index m terminiert werden, falls gilt:

$$\frac{c_m}{\lim\limits_{c\to\infty} \mathbb{E}_c(Z)} \frac{\partial\mathbb{E}_c(Z)}{\partial c}\bigg|_{c=c_m} < \varepsilon, \tag{3.28}$$

woraus sich nach dem Einsetzen der entsprechenden Terme das **Abbruchkriterium** von Simulated Annealing basierend auf Überlegungen der statistischen Mechanik ergibt:

$$\frac{c_m}{\frac{1}{|\mathbb{S}|}\sum_{t\in\mathbb{S}} Z(t)} \frac{\sigma_c^2}{c^2}\bigg|_{c=c_m} = \frac{\sigma_{c_m}^2}{\frac{c_m}{|\mathbb{S}|}\sum_{t\in\mathbb{S}} Z(t)} < \varepsilon. \tag{3.29}$$

Der Parameter ε ist eine sehr kleine Zahl, die z.B. auf den Wert $\varepsilon = 10^{-5}$ gesetzt werden kann, damit der Algorithmus beendet wird (vgl. Aarts/Korst 1989, S. 69). In der praktischen Anwendung sind natürlich alle Lösungen des Lösungsraums bekannt, weshalb nicht der Mittelwert für den gesamten Suchraum, sondern ein Ausschnitt der letzten Lösungen um den Kontrollparameter c herum verwendet wird, um das arithmetische Mittel einer geeigneten Teilmenge des Suchraums zu approximieren (z.B. die Lösungen der Markovkette m oder eine noch kleinere Teilmenge). Die Varianz $\sigma_{c_m}^2$ verwendet ebenso entweder die Varianz der Zielfunktionswerte aus der letzten Markovkette m oder eine Teilmenge daraus. Hierbei obliegt es dem Anwender das richtige Maß auszutarieren. Dieses Vorgehen wird auch *smoothing* genannt, um vorzeitige Konvergenz durch starke Fluktuationen in der Differenz der Zielfunktionswerte am Anfang des Algorithmus zu verhindern, wo die Werte für den Kontrollparameter c_m noch recht hoch sind (vgl. Aarts/Korst 1989, S. 64).

In Abschnitt 4.6 wird dieses Vorgehen dezidiert für die Produktlinienoptimierung und deren Zielfunktionen beschrieben. Hierbei ergeben sich insofern Unterschiede, als dass es sich bei den in dieser Arbeit berücksichtigten Produktlinienoptimierungsproblemen um Maximierungsprobleme handelt, das allgemeine Vorgehen jedoch anhand von Minimierungsproblemen verdeutlicht wurde.

4 Machine-Learning-Verfahren in der Produktlinienoptimierung

In diesem Kapitel werden die Produktlinienoptimierung und die Machine-Learning-Verfahren zusammengeführt. Es besteht aus einer Literaturanalyse mit verschiedenen Schwerpunkten. Zum einen wird eine Übersicht der Literatur für den Einsatz von Machine-Learning-Verfahren in der Produktlinienoptimierung ab 1995 gegeben. Zum anderen werden für die in dieser Arbeit verwendeten Verfahren zusätzlich weitere verfahrensspezifische Details in den jeweiligen Unterkapiteln beschrieben. Nachdem die Implementierung der Verfahren beschrieben wurde, werden Simulationsrechnungen durchgeführt, um die geeignetsten Vertreter der jeweiligen Verfahren zu bestimmen, die in Kapitel 5 gegeneinander gebenchmarkt werden, um die generelle Eignung der einzelnen Verfahren zur Lösung von Produktlinienoptimierungsproblemen herauszufinden.

4.1 Literaturübersicht

Die im Folgenden aufgeführte Literaturanalyse beginnt im Jahr 1995 und berücksichtigt nur Veröffentlichungen, in denen Heuristiken auf das Produktlinienoptimierungsproblem angewendet werden. Des Weiteren wird in diesem Abschnitt nicht auf die Details der verwendeten Heuristiken eingegangen. Diese werden dezidiert in späteren Abschnitten vorgestellt und diskutiert.

In Tabelle 4 sind die Beiträge zur Produktlinienoptimierung auf Basis der Conjointanalyse seit 1995 aufgeführt. Die Spalte **Zielfunktion** zeigt an, welche Zielfunktionen in den Publikationen verwendet werden. Dabei werden grundsätzlich drei verschiedene Zielfunktionen (Gewinn, Kundennutzen und Marktanteil) unterschieden. Die Spalte **Entscheidungsregel** zeigt an, wie die Entscheidungen der Konsumenten für ein bestimmtes Produkt zustande kommen. In dieser Arbeit wird untergliedert in deterministische und probabilistische Entscheidungsregeln. First-Choice bedeutet, dass die Produktauswahlentscheidungen deterministisch modelliert werden (siehe Abschnitt 2.2.2). BTL und Logit stehen für probabilistische Auswahlentscheidungen

(siehe Abschnitt 2.2.3). Der **Produktfall** gibt an, ob über ein einzelnes Produkt (Einproduktfall) oder über Produktlinien optimiert wird. Mit **Anzahl möglicher Lösungen** wird die Größe der Lösungsräume angegeben, also die Menge der zulässigen Produkte bzw. Produktlinien. Dabei wird die kleinste sowie die größte Anzahl von zulässigen Produktlinien in der entsprechenden Publikation gegeben. Für die Simulationsstudien müssen die Teilnutzenwerte für die Konsumenten erzeugt werden. Aus welcher Wahrscheinlichkeitsverteilung diese gezogen werden, kann aus der Spalte **Teilnutzenwerte** entnommen werden. In der Spalte **Lösungsverfahren** werden alle verwendeten Verfahren, exakte als auch heuristische, angegeben, wobei in der Spalte **beste Heur.** diejenige Heuristik genannt wird, die die höchste Lösungsqualität liefert, ergo, die höchsten Zielfunktionswerte erzielte. Weitere Rahmenbedingungen der Simulationen werden in **Eigenschaften der Simulation** aufgeführt, z. B. wie viele Konsumenten für die Simulationen erzeugt werden, wie der Status-quo-Markt generiert wird, wie viele Testprobleme erzeugt und analysiert werden sowie andere Besonderheiten die Simulationen betreffend. Aus der Spalte **Anwendung** geht hervor, ob reale Conjointdatensätze untersucht werden und aus welchem Gebiet diese stammen. In einigen Publikationen werden zwar reale Conjointdaten benutzt, jedoch der konkrete Anwendungsfall nicht aufgeführt (z. B. Camm et al. 2006 und Wang et al. 2009).

In Tabelle 5 sind die Faktoren und Level aller Publikationen abgebildet, in denen Simulationsstudien für die Heuristiken durchgeführt werden. Dabei ist die Anzahl der Eigenschaften und deren Eigenschaftsausprägungen angegeben. Zusätzlich wird ersichtlich, wie viele Produkte in die Produktlinie aufgenommen und wie viele Konsumenten in die Simulationsstudie einbezogen werden. Für Publikationen, in denen Wettbewerberreaktionen berücksichtigt werden, ist die Anzahl der Wettbewerber angegeben. Mit **Wiederholungen eines Problems** ist gemeint, wie oft eine erzeugte Probleminstanz (alle Faktoren und Level sowie die Teilnutzenwerte der Konsumenten bleiben fixiert) von den Heuristiken gelöst wird. Die in Tabelle 5 aufgeführten Faktoren und Level bilden nicht notwendigerweise vollständige Versuchspläne. Einige Autoren betrachten lediglich eine Teilmenge aller im Rahmen ihrer Simulationsstudien zulässigen Probleminstanzen.

Michalek et al. (2005; 2011), Luo (2011), Sadeghi et al. (2011) und Cao et al. (2012) erstellen mehrstufige Optimierungsmodelle, um die Perspektive der Marktforschung mit der Perspektive des Produktingenieurs zu verbinden. Dahinter steht der Gedanke, dass eine optimale Produktlinie aus Marketingsicht nicht notwendigerweise der optimalen Produktlinie im Herstellungsprozess entspricht. Dabei entstehen gemischt-ganzzahlige Optimierungsprobleme, die ebenfalls auf Heuristiken angewiesen sind. So implementieren z. B. Luo (2011) und Cao et al. (2012) einen Genetischen Algorithmus zur Lösung des zugrundeliegenden Marketing-Produktingenieur-Optimierungsproblems. Simulated Annealing wird von Luo (2011) und Sadeghi et al. (2011) zur Lösung ihres Ansatzes genutzt. Da sich diese Modelle fundamental von den in dieser Arbeit betrachteten Modellen unterscheiden, werden diese Hybrid-Ansätze nicht weiter berücksichtigt,

da der Fokus auch eher auf den Optimierungsmodellen und nicht auf der Lösung dieser mittels Heuristiken liegt. Eine kurze Zusammenfassung hierzu bieten Hauser (2011) und Liberali (2011). Tsafarakis et al. (2013) beziehen für die Optimierung von Produktlinien stetige Variablen in eine Abwandlung des probabilistischen Marktanteilsmaximierungsmodells ein. Zur Lösung verwenden sie die Particle-Swarm-Optimization, einen Genetischen Algorithmus sowie verschiedene Hybridisierungen und demonstrieren an einem Beispiel für Industriekräne die Möglichkeit, Optimierungsmodelle dieser Art mit Heuristiken zu lösen. Da das Optimierungsmodell grundlegende Unterschiede zu den hier betrachteten Modellen aufweist, wird es nicht weiter betrachtet.

Einen weiteren Ansatz für das Produktliniendesign verfolgen **Aydin et al. (2015)**, die für eine neue Produktlinie sowohl neue als auch wiederaufbereitete Produkte, Pricing-Entscheidungen von Unternehmen aus der Wertschöpfungskette und die Rücksenderate für die wiederaufbereiteten Produkte berücksichtigen. Die Auswahlentscheidungen für die Produkte der Konsumenten werden durch die Logit-Auswahlregel modelliert (Aydin et al. 2015, S. 3) und im mehrstufigen Optimierungsmodell werden Konkurrenzreaktionen über das Nash-Stackelberg-Gleichgewicht berücksichtigt. Da es sich hier um ein grundlegend anderes Optimierungsmodell handelt und keine Heuristiken zur Lösung dieses implementiert oder vorgestellt werden, wird dieses Modell in dieser Arbeit der Vollständigkeit halber genannt, aber nicht weiter ausgeführt.

Raman/Chhajed (1995) präsentieren ein Modell, das zum einen die Präferenzen der Kunden einbezieht als auch die beim Produktionsprozess anfallenden Kosten berücksichtigt. Dabei wird die Anzahl der möglichen Produkte für die Auswahl in die Produktlinie begrenzt, indem die aktiven Level der Attribute für jedes Produkt spezifiziert werden. Danach wird der Preis für jedes Produkt festgelegt und die Kosten für die Produktionsprozesse gemäß der benötigten Level zugeordnet. Um den Gewinn zu maximieren, werden die Kunden unter Berücksichtigung der Produktionskosten und der Preise den ausgewählten Produkten gemäß ihrer Präferenzen zugerechnet (Raman/Chhajed 1995, S. 190). Das zugrunde liegende Optimierungsproblem ist ebenfalls NP-hart. Deshalb geben die Autoren einen vierstufigen Algorithmus an, der Simulated Annealing verwendet. Es werden allerdings keine Heuristiken oder die Qualität von Lösungen verglichen oder versucht, größere Produktlinien zu optimieren. Vielmehr wird ein älterers sequentielles Verfahren mit einem neuen integrierten Verfahren in Bezug auf den Gewinn verglichen und gezeigt, dass das neue integrierte Verfahren dem älteren überlegen ist (Raman/Chhajed 1995, S. 196).

Balakrishnan/Jacob (1995) begegnen der Tatsache, dass aufgrund der zugrunde liegenden kombinatorischen Produktoptimierungsprobleme keine Lösung in angemessener Zeit gefunden werden kann, mit einem Ansatz, der ein Entscheidungsunterstützungssystem mit einer Heuristik, aus dem Bereich der Genetischen Algorithmen, verbindet.

Quellen	Zielfunktion	Auswahl-regel	Produkt-fall	Anzahl möglicher Produktlinien; {min,...,max}	Teil-nutzen-werte	Lösungs-prinzip	Lösungs-verfahren	höchste Ziel-funktions-werte (ohne exakte Verfahren)	Eigenschaften der Simulation	Anwendung
Raman/Chhajed (1995)	Gewinn	First Choice	Produkt-linie	$27 – 2.1 \cdot 10^{79}$	gleich-verteilt	heuristisch	SA	k.A.	10 Konsumenten; 40 Testprobleme; Preise gleichverteilt; zufälliger Status-quo-Markt	keine
Balakrishnan/Jacob (1995)	Marktanteil, Kunden-nutzen	First Choice	Einzel-produkt	256	k.A.	heuristisch	CE, DP, GA	GA	ein Testproblem	keine
Nair et al. (1995)	Marktanteil, Kunden-nutzen, Gewinn	First Choice	Produkt-linie	$120 – 6.3 \cdot 10^{5}$	gleich-verteilt	heuristisch	CE, DP, BS	BS	jeweils 435 Testprobleme; zufälliger Status-quo-Markt mit 3 Produkten	Telefone
Balakrishnan/Jacob (1996)	Marktanteil, Kunden-nutzen	First Choice	Einzel-produkt	16 – 390625	gleich-verteilt	heuristisch	CE, DP, GA	GA	jeweils 192 Testprobleme; zufälliges Status-quo-Produkt für jeden Konsumenten	keine
Steiner/Hruschka (2000)	Gewinn	Logit	Produkt-linie	hoch (nicht direkt vergleichbar)	k.A.	exakt	CE	k.A.	k.A.	keine
Chen/Hausman (2000)	Gewinn	Logit	Produkt-linie	k.A.	k.A.	exakt	Standard-verfahren	k.A.	k.A.	keine
Thakur et al. (2000)	Marktanteil	First Choice	Produkt-linie	120 – 523776	gleich-verteilt	exakt, heuris-tisch	GA, BS	BS	300 Testprobleme; Preise gleichverteilt; zufälliger Status-quo-Markt mit 3 Produkten	Telefone
Alexouda/Paparizzos (2001)	Gewinn	First Choice	Einzel-produkt, Produkt-linie	$2016 – 3.66 \cdot 10^{15}$	gleich-verteilt	exakt, heuris-tisch	GA, BS, GA/BS-Hybrid	GA	300 Testprobleme; gleichverteilte Deckungsbeiträge; zufälliger Status-quo-Markt mit 3 Produkten	keine
Shi et al. (2001)	Marktanteil	First Choice	Produkt-linie	$3125 – 5.04 \cdot 10^{102}$	beta- und gleich-verteilt	exakt, heuris-tisch	BS, DC, DP, GH, GA, NPA, mehrere NPA-Hybride	Hybrid aus NPA-GA-GH	217 Testprobleme; Konsumenten = 400; zufälliger Status-quo-Markt mit 3 Produkten	keine
Tarasewich/McMullen (2001)	Marktanteil	First Choice	Einzel-produkt	1024 – 59049	gleich- und normal-verteilt	exakt, heuris-tisch	CE, GA, TS, Pruning-Heuristik	Pruning-Heuristik	120 Testprobleme; zufälliger Status-quo-Markt	keine

(Fortsetzung der Tabelle auf der nächsten Seite)

Quellen	Zielfunktion	Auswahl-regel	Produkt-fall	Anzahl möglicher Produktlinien; {min,...,max}	Teil-nutzen-werte	Lösungs-prinzip	Lösungs-verfahren	höchste Ziel-funktions-werte (ohne exakte Verfahren)	Eigenschaften der Simulation	Anwendung
Steiner/Hruschka (2002)	Gewinn	Logit	Produkt-linie	$28 - 4.6 \cdot 10^{10}$	k.A.	z.T. exakt, heuris-tisch	CE, GA, GH	GA	276 Testprobleme	keine
Steiner/Hruschka (2003)	Gewinn	Logit	Produkt-linie	$28 - 4.6 \cdot 10^{10}$	k.A.	exakt, heuris-tisch	CE, GA, GH	GA	276 Testprobleme	Seife
Balakrishnan et al. (2004)	Marktanteil	First Choice	Produkt-linie	$6.12 \cdot 10^{26} - 1.88 \cdot 10^{38}$	gleich-verteilt	exakt, heuris-tisch	IBM CPLEX, GA, BS, GA/BS-Hybride	GA	jeweils 800 Testprobleme; Konsumenten = {200}; zufälliger Status-quo-Markt	keine
Fruchter et al. (2006)	Gewinn	First Choice	Einzel-produkt	k.A.	k.A.	heuristisch	GA	kein Ver-gleich	3 Testprobleme, Konsumenten = {30, 150, 250}	Laptops
Camm et al. (2006)	Marktanteil	First Choice	Einzel-produkt	$2.95 \cdot 10^{5} - 7.21 \cdot 10^{16}$	beta-verteilte Abwei-chungen von Attribut-mittel-werten	exakt, heuris-tisch	LRBB, GH	k.A.	96 Testprobleme, Konsumenten = {600, 1200}, zufälliger Status-quo-Markt	2 Daten-sätze mit realen Conjoint-daten
Albritton/McMullen (2007)	Marktanteil	First Choice	Einzel-produkt	$1.0 \cdot 10^{5} - 1.0 \cdot 10^{9}$	gleich-verteilt	exakt, heuris-tisch	CE, ACO	nicht ent-scheidbar	4200 Testprobleme; Konsumenten = {200, 250, 300}; kein Status-quo-Markt	keine
Belloni et al. (2008b)	Gewinn	First Choice	Produkt-linie	$56 - 5 \cdot 10^{15}$	gleich-verteilt	exakt, heuris-tisch	LRBB, BS, CoA, DC, DP, GA, NPA, PSH, SA	SA	120 Testprobleme, Konsumenten = {50, 100}; zufälliger Status-quo-Markt; Robustness-Test der optimalen Lösung	Kuriertaschen

(Fortsetzung der Tabelle auf der nächsten Seite)

Quellen	Zielfunktion	Auswahl-regel	Produkt-fall	Anzahl möglicher Produktlinien; {min,...,max}	Teil-nutzen-werte	Lösungs-prinzip	Lösungs-verfahren	höchste Ziel-funktions-werte (ohne exakte Verfahren)	Eigenschaften der Simulation	Anwendung
Wang et al. (2009)	Marktanteil	First Choice	Produkt-linie	$2.49 \cdot 10^{11} - 7.95 \cdot 10^{49}$	beta-verteilte Abwei-chungen von Attribut-mittel-werten	exakt	Branch& Price-Alg.	k.A.	Konsumenten = {2, 321, 1131}, zufälliger Status-quo-Markt; Robustness-Test der opti-malen Lösung	zwei kom-merzielle Anwen-dungen
Tsafarakis et al. (2011)	Marktanteil, Kunden-nutzen	BTL	Produkt-linie	$1.58 \cdot 10^{15} - 3.90 \cdot 10^{67}$	gleich-verteilt	heuristisch	GA, PSO	GA	160 Testprobleme, Konsumenten = {200, 700}, zufälliger Status-quo-Markt	Milch
Wang/Curry (2012)	Marktanteil	First Choice	Einzel-produkt	1728	aus HB-Schätzung	exakt	IBM CPLEX	keine	Robustness-Test der optimalen Lösung; kein Status-quo-Markt	Bankkonten
Voekler et al. (2013)	Marktanteil	First Choice	Einzel-produkt	$625 - 1.0 \cdot 10^{20}$	gleich-verteilt	heuristisch	ACO, BCO	ACO	450 Testprobleme; Konsumenten = {250, 500, 1000}; kein Status-quo-Markt	keine

Tabelle 4: Übersicht der Methoden in der Produktlinienoptimierung seit 1995 (Quelle: Eigene Darstellung in Anlehnung an Baier/Gaul (1999)). Anmerkung: Die Spalte „beste Heuristik“ bezieht sich auf diejenige Heuristik, die die höchsten Zielfunktionswerte erreicht hat. Die Werte für die Anzahl möglicher Produktlinien sind bis auf die erste Nachkommastelle der Zehnerpotenz gerundet. Legende: ACO = Ant-Colony-Optimization, Alg. = Algorithmus, BCO = Bee-Colony-Optimization, BS = Beam-Search[3], CE = komplette Enumeration, CoA = Coordinate Ascent, DC = Divide&Conquer-Heuristik, DP = Dynamic-Programming-Heuristik[4], GA = Genetischer Algorithmus, GH = Greedy-Heuristik, HB = hierarchischer Bayes, k.A. = keine Angabe, LRBB = Lagrange-Relaxation mit Branch&Bound, NPA = Nested-Partitions-Algorithmus, PSH = Product-Swapping-Heuristik, TS = Tabu-Search.

[3] Beam-Search wird den Arbeiten von Lowerre (1976) und Rubin (1978) zugeschrieben. Da sich Beam-Search als anderen Heuristiken unterlegen erweisen wird (vgl. Alexouda/Paparizzos (2001), Balakrishnan et al. (2004) und Belloni et al. (2008b)), wird es in dieser Arbeit nicht explizit behandelt. Eine kurze Beschreibung soll hier nichtsdestotrotz gegeben werden: Beam Search ist eine *breadth-first*-Suche, was bedeutet, dass die Suche für gute Lösungen an einem Wurzelknoten beginnt und von dort aus in die Tiefe des Suchbaums geht. Die Breite des Suchbaums wird dabei beschränkt. Eine vorher definierte Anzahl dieser Wurzelknoten wird festgelegt und die Suchbäume werden nach und nach so abgeschnitten, dass bessere Lösungen weiter untersucht werden und schlechte Suchregionen vermieden werden. Somit kann nicht sichergestellt werden, die optimale Lösung eines Problems zu finden. In der Produktlinienoptimierung stellt jede Wurzel ein Produkt dar, die um weitere vielversprechende Produkte auf Basis des Zielfunktionswerts erweitert wird. Für Implementierungsdetails wird auf Nair et al. 1995, S. 770 ff. verwiesen.

[4] Die Dynamic-Programming-Heuristik von Kohli/Krishnamurti (1987, S. 1526 ff.) erzeugt Näherungslösungen, indem sie die Produktprofile partiell nacheinander über die Eigenschaften aufbaut. Dabei wird der Einfluss bestimmter Level auf den Zielfunktionswert so ermittelt, dass vielversprechende Produktlinien mit besseren Zielfunktionswerten übrig bleiben und am Ende diejenige Produktlinie mit dem besten Zielfunktionswert als Lösung verwendet wird.

Durch eine Triangulation wird versucht, die Qualität der gefundenen Lösung durch die Bestimmung von unteren bzw. oberen Schranken, je nachdem, ob minimiert oder maximiert werden soll, einordnen zu können (Balakrishnan/Jacob 1995, S. 313). Für kleine Probleme implementieren die Autoren eine komplette Enumeration. Für größere Probleme wird zum Vergleich für den Genetischen Algorithmus die Dynamic-Programming-Heuristik von Kohli/Krishnamurti (1987) herangezogen (Balakrishnan/Jacob 1995, S. 318). Mithilfe des elektronischen Entscheidungsunterstützungssystem namens *GENESYS* kann ein User auswählen, welche Zielfunktion er maximieren möchte: den Marktanteil oder den Nutzen. Das Ziel dieser Veröffentlichung ist es nicht, einen vollständigen Vergleich von Heuristiken zu liefern, sondern eine Software vorzustellen, die es Entscheidern in Unternehmen ermöglicht einzelne Produkte mit verschiedenen Verfahren und Zielfunktionen zu optimieren.

Nair et al. (1995) führen die Beam-Search-Heuristik in die Produktlinienoptimierung ein. Die Autoren betrachten dabei das simultane Produktlinienoptimierungsproblem aus Kohli/Sukumar (1990). Simultan bedeutet in diesem Zusammenhang, dass drei Kriterien bei der Optimierung gleichzeitig einbezogen werden: der Marktanteil eines Produkts, der Nutzen eines Produkts für die Konsumenten und die Teilstückdeckungskosten der Eigenschaftsausprägungen (Nair et al. 1995, S. 768/769). Die Beam-Search-Heuristik stammt aus dem Bereich der künstlichen Intelligenz und ist weit verbreitet. Sie zählt zu den *breadth-first*-Verfahren, was bedeutet, dass sie vor allem in der Breite und weniger lokal nach Optima sucht. Damit die Suche auf lokale Lösungen eingegrenzt werden kann, muss die Breite der Suche eingeschränkt werden (Nair et al. 1995, S. 770). Das Verfahren sucht dabei vielversprechende Knoten, die in der weiteren Suche besonders berücksichtigt werden. Knoten mit schlechter Lösungsqualität werden dabei stetig beschnitten, um den Suchraum weiter einzugrenzen. Die vielversprechenden Knoten werden weiter in die Breite und die Tiefe entwickelt. Beam Search wird in dieser Arbeit nicht weiter detailliert betrachtet, da z. B. Belloni et al. (2008b) zeigen konnten, dass Genetische Algorithmen und Simulated Annealing bei den meisten simulierten Testproblemen insgesamt besser abschnitten, vor allem wenn die Anzahl möglicher Lösungen deutlich zunimmt. In Nair et al. (1995) wird zum Vergleich zu Beam Search eine Dynamic-Programming-Heuristik aus Kohli/Sukumar (1990) verwendet. Dabei zeigt sich, dass Beam Search der Dynamic-Programming-Heuristik in allen Vergleichskriterien überlegen (durchschnittliche Lösungsqualität, Standardabweichung der Lösungen, Optimum gefunden, Anzahl der besseren Lösungen im Vergleich zu Dynamic-Programming-Heuristik, Anzahl an guten Produktlinien, Laufzeit) ist, sowohl bei den simulierten als auch bei den realen Conjointdaten (Nair et al. 1995, S. 779). Für diese Ergebnisse werden die Faktoren, wie in Tabelle 5, verwendet. Als Anwendungsfall nehmen sich die Autoren einen Datensatz von Kohli/Sukumar (1990), der zwar echte Teilnutzenwerte einer Conjointstudie enthält, jedoch aus Gründen der Geheimhaltung als Telefone deklariert wurde. Die Faktoren und die Level für die Simulationsstudie bilden keinen vollständigen Versuchsplan. Die Autoren wählten zunächst 17 Datensätze aus 27 möglichen aus, die sich aus den Eigenschaften sowie deren Eigenschaftsausprägungen und der Anzahl der Produkte pro Produktlinie zusammensetzen.

Jedes dieser 17 Testprobleme wird für alle drei Konsumentenkonfigurationen verwendet. Insgesamt werden für jede Zielfunktion 51 Testprobleme gelöst, von denen 36 Stück zehnmal und 15 Stück fünfmal wiederholt werden, was insgesamt 435 Testproblemen entspricht. Die mittlere Performance über alle Zielfunktionen und Testprobleme hinweg zeigt einen Vorteil von Beam Search bzgl. der besten gefundenen Lösung im Verhältnis zum Optimum von 99.4 % zu 97.1 % gegenüber der Dynamic-Programming-Heuristik. In 58.2 %, 77.2 % und 85.7 % der Fälle für den Marktanteil, den Kundennutzen und den Gewinn findet Beam Search bessere Lösungen als die Dynamic-Programming-Heuristik. Beam Search ist der Dynamic-Programming-Heuristik auch bei der absoluten Häufigkeit, wie oft die optimale Lösung gefunden wurde, überlegen. Und zwar um die Faktoren 2.24, 3.88 und 4.96 für die entsprechenden Zielfunktionen. Nair et al. (1995) wenden Beam Search auf ein reales Problem an (aus Kohli/Sukumar (1990)) und vergleichen es mit der Performance der Dynamic-Programming-Heuristik. Auch hierbei bietet Beam Search deutliche Vorteile gegenüber der Dynamic-Programming-Heuristik bei allen wichtigen Vergleichskriterien.

Die erste umfassende Arbeit des Genetischen Algorithmus auf das Produktoptimierungsproblem angewendet, bieten **Balakrishnan/Jacob (1996)**. Die Autoren untersuchen vier Forschungsfragen in ihrer Arbeit. Inwiefern Genetische Algorithmen auf das Produktoptimierungsproblem angewendet werden kann, wie gut der implementierte Algorithmus gegen die Dynamic-Programming-Heuristik aus Kohli/Krishnamurti (1987) abschneidet, die Sensitivität der gefundenen Lösungen bei Variationen in der Wahl der Parameter sowie die Verallgemeinerung der Dynamic-Programming-Heuristik auf Eigenschaften mit unterschiedlicher Anzahl von Eigenschaftsausprägungen (Balakrishnan/Jacob 1996, S. 1105). Für den Vergleich werden kleinere Probleminstanzen erzeugt, so dass die Performance der betrachteten Heuristiken bzgl. des Optimums gemessen werden kann. Für die Vergleichsstudie wird ein vollständiger Versuchsplan erstellt. Dieser besteht aus insgesamt 48 Problemen, die jeweils vier Mal wiederholt werden, was in einer Gesamtanzahl von 192 Problemen mündet. Eine Übersicht der Faktoren und Level bietet Tabelle 5. Ein Status-quo-Markt wird nicht feststehend modelliert, sondern jedem Konsumenten wird zufällig ein Produkt als Status-quo-Produkt zugewiesen. Somit ist es also theoretisch möglich, dass der Status-quo-Markt aus genauso vielen Produkten besteht, wie es Konsumenten in der Studie gibt. Obwohl diese Annahme nicht realistisch ist, ergeben sich keine zu erwartenden negativen Effekte auf die Performanceevaluierung.

Die Testprobleme werden mit einer geeigneten Parameterauswahl (siehe Abschnitt 4.3) für den Genetischen Algorithmus mit der Dynamic-Programming-Heuristik verglichen. Hierbei zeigte sich für beide Zielfunktionen die Überlegenheit des Genetischen Algorithms gegenüber der Dynamic-Programming-Heuristik. Für die Marktanteilsmaximierung ergeben sich für die 192 Testprobleme eine mittlere Lösungsqualität gemessen am Optimum für den Genetischen Algorithmus von 99.13 % gegenüber von 96.67 % für die Dynamic-Programming-Heuristik. Beim Kundennutzen geht der Vergleich zugunsten des Genetischen Algorithmus mit 99.92 % zu 98.76

% aus. Ob die Ergebnisse statistisch signifikante Unterschiede aufweisen, wurde nicht geprüft. Allerdings traf der Genetische Algorithmus in 123 Fällen von 192 Testproblemen die optimale Lösung bei der Marktanteilsmaximierung zu 51 für die Dynamic-Programming-Heuristik. Für den Kundennutzen lagen die absoluten Treffer analog bei 175 zu 82 von 192 Testproblemen. Des Weiteren zeigt sich, dass die Varianz der Lösungen für den Genetischen Algorithmus geringer ist als bei der Dynamic-Programming-Heuristik.

In **Steiner/Hruschka (2000)** findet sich ein spieltheoretischer Ansatz zur Berücksichtigung von Konkurrenzsituationen für Produktlinien. Dabei steht u. a. die Bestimmung von simultanen Positions-Preis-Gleichgewichten im Vordergrund, aber auch die Existenz von ineffizienten Marktgleichgewichten als auch der Differenzierungsgrad zwischen Konkurrenzprodukten (Steiner/Hruschka 2000, S. 71). Für ihr Modell verwenden die Autoren eine Gewinnfunktion, die mit einer probabilistischen Entscheidungsregel für die Auswahlwahrscheinlichkeiten der Produkte, der Logit-Regel, gewichtet wird. Durch die Modellierung von Tatonnement-Prozessen, soll gezeigt werden, dass die Nash-Gleichgewichte für diese Szenarien tatsächlich existieren. Aufgrund der Notwendigkeit der Modellierung expliziter Tatonnement-Prozesse, ergibt sich eine kombinatorische Steigerung der Anzahl von zu betrachtenden potentiellen Marktgleichgewichten mit der Anzahl zur Verfügung stehender Produkte und der Anzahl von Wettbewerbern auf dem Markt (Steiner/Hruschka 2000, S. 81). Deshalb ist eine Enumeration eines effizienten Marktgleichgewichts nur bei einer geringen Anzahl von Wettbewerbern und einer geringen Anzahl von Kombinationen für die Produktlinien möglich. Zur Lösung dieses Problems schlagen die Autoren geeignete Heuristiken vor (Steiner/Hruschka 2000, S. 85).

Chen/Hausman (2000) stellen eine Methode vor, die die mathematischen Eigenschaften der Choice-Based Conjointanalyse in Hinblick auf das Produktliniengestaltungsproblem untersucht. Durch die speziellen Eigenschaften des Logit-Modells ist es den Autoren möglich, einen effizienten Algorithmus zu entwickeln, der die Optimierung für kleinere Problemgrößen möglich macht. Eine Einschränkung ist, dass lediglich über die aggregierten Teilnutzenwerte optimiert wird und nicht über die individuellen Teilnutzenwerte, die aus einer hierarchischen Bayes-Schätzung stammen (Chen/Hausman 2000, S. 327). Zur exakten Lösung des Optimierungsproblems wird gezeigt, dass eine optimale Lösung existiert, wenn die binäre Entscheidungsvariablen relaxiert werden, da die Zielfunktion unter bestimmten Voraussetzungen quasi-konvex ist (Chen/Hausman 2000, S. 330). Die optimale Lösung ergibt sich durch den Einsatz von Standardverfahren der nichtlinearen Optimierung. Trotz der Möglichkeit des Findens des globalen Optimums muss eingewendet werden, dass sich hiermit lediglich kleine bis mittelgroße Produktlinienoptimierungsprobleme lösen lassen, wie von Camm et al. (2006) und Belloni et al. (2008b) gezeigt werden konnte.

Thakur et al. (2000) erweitern die Modelle für Produktlinienoptimierung, die auf den Präferenzen von Konsumenten beruhen, um Produktpreise relativ zum Budget der Konsumenten.

Hierfür bedienen sich die Autoren des klassischen Marktanteilsmaximierungsmodells mit der First-Choice-Entscheidungsregel. Als Nebenbedingungen für das Optimierungsmodell werden Schwellwerte für die Konsumenten bestimmt, die den Kauf eines Produkts verhindern, wenn der Gesamtnutzen des Produkts zum Kauf führen würde, aber aufgrund der Beschränkung des Budgets des Konsumenten nicht möglich ist. Die Einbeziehung von Preisen für ein Produkt auf Eigenschaftsausprägungsebene bietet den Vorteil, Sensitivitätsanalysen bzgl. von Preis-Nachfrage-Elastizitäten und Preis-Gewinn-Beziehungen unter verschiedenen Bepreisungsszenarien. Trotz der Hinzunahme von Preisinformationen sei explizit erwähnt, dass im vorliegenden Modell nicht der Gewinn, sondern der Marktanteil maximiert werden soll. Ein Konsument kauft ein Produkt, wenn der Gesamtnutzen des neuen Produkts den Gesamtnutzen des Status-quo-Produkts übersteigt, ein Schwellenwert für jede Eigenschaftsausprägung erfüllt wird und der Preis des Produkts das Budget des Konsumenten nicht übersteigt. Um das zugrunde liegende Optimierungsmodell zu lösen, bedienen sich die Autoren des Genetischen Algorithmus sowie Beam Search auf Basis des von Nair et al. (1995) vorgestellten Ansatzes. Der Versuchsaufbau für die Simulation orientiert sich an Kohli/Sukumar (1990). Der Versuchsaufbau der Simulationsstudie kann aus Tabelle 5 entnommen werden. Aus diesen Konfigurationen wurden elf Probleme ausgewählt. Die zwei größten Probleme wurden aufgrund der langen Laufzeit der Algorithmen jeweils fünfmal für jede Konsumentenanzahl wiederholt, die anderen neun Probleme jeweils zehnmal. Der Genetische Algorithmus wurde lediglich für 100 Konsumenten getestet. Für die Teilnutzenwerte und die Schwellwerte für die Nebenbedingung wurde eine Gleichverteilung im Intervall $(0, 1)$ verwendet. Die Preisvektoren für die Eigenschaftsausprägungen wurden gleichverteilt im Intervall $(10, 50)$ erzeugt. Der Status-quo-Markt wird zufällig mit drei Produkten erzeugt. Über alle Probleme hinweg liefert Beam Search eine durchschnittliche Lösungsqualität von 99.4 % bzgl. der optimalen Produktlinien. Der Genetische Algorithmus bietet eine durchschnittliche Lösungsqualität von 94.0 % für 100 Konsumenten. Beam Search besitzt für alle Problemgrößen (Anzahl der Konsumenten = 100) eine geringere Standardabweichung als der Genetische Algorithmus. Die Autoren räumen ein, dass der Genetische Algorithmus mit einer Vergrößerung der Populationsgröße eine bessere Lösungsqualität erbringen könnte, jedoch die Laufzeit die technischen Möglichkeiten ihres Setups übersteigen würde. Diese Tatsache und die fehlende Sensitivitätsanalyse der Parameter des Genetischen Algorithmus sind wahrscheinlich der Grund für die Unterlegenheit des Genetischen Algorithmus gegenüber Beam Search, denn sowohl Alexouda/Paparizzos (2001) als auch Balakrishnan et al. (2004) zeigen in umfangreicheren Untersuchungen die Überlegenheit des Genetischen Algorithmus gegenüber Beam Search für die Produktlinienoptimierung. Die Wahrscheinlichkeit, dass die Abwandlung der Zielfunktion hier den Ausschlag zugunsten Beam Search gegeben hat, ist eher gering, da durch die zusätzlichen Restriktionen in den Nebenbedingungen die Anzahl der lokalen Optima verringert worden sein dürfte. Als Anwendungsfall bedienen sich die Autoren bei einem Datensatz von Kohli/Sukumar (1990), das zwar echte Teilnutzenwerte einer Conjointstudie enthält, jedoch aus Gründen der Geheimhaltung als Telefone deklariert wurde. Auf dieses Beispiel wurde der Genetische Algorithmus nicht angewendet, lediglich Beam Search, das allerdings für

eine verschiedene Anzahl von Produkten in einer Produktlinie Ergebnisse liefert, die nahe am Optimum liegen.

Alexouda/Paparizzos (2001) verbinden zwei Heuristiken, die, jede für sich genommen, eine bessere Performance haben als die Dynamic-Programming-Heuristik, Beam Search und Genetische Algorithmen. Durch die Kombination dieser Heuristiken sollen die Stärken beider genutzt werden. Beam Search bietet eine bessere Abdeckung des Suchraums, wohingegen Genetische Algorithmen vor allem gut in der lokalen Suche sind. Als Zielfunktion wird die Gewinnmaximierung aus Kohli/Sukumar (1990) verwendet. Jedoch wird einer möglichen Kannibalisierung der eigens angebotenen Produkte eines Unternehmens auf dem Status-quo-Markt Rechnung getragen, indem die Differenz der Deckungsbeiträge für die Optimierung verwendet wird, wenn ein Konsument von seinem Status-quo-Produkt des betreffenden Unternehmens zu einem neu angebotenen Produkt wechselt. Für das beschriebene Optimierungsmodell spielt diese Tatsache jedoch keine Rolle. Der Hybrid aus Genetischem Algorithmus und Beam Search ist kein Hybrid im eigentlichen Sinne. Er besteht darin, dass die von Beam-Search erzeugten Produktlinien in die Startpopulation eines Genetischen Algorithmus integriert und mit zufällig erzeugten Produktlinien aufgefüllt wird. Aufgrund der großen Lösungsräume lösen die Autoren acht kleinere Probleme mittels kompletter Enumeration. Für den Großteil der Simulationsstudie ist dies allerdings nicht möglich. Des Weiteren verfolgen die Autoren einen anderen Ansatz als ihre Vorgänger für die Simulationsstudie. Statt für jede Problemgröße einen Datensatz zu erzeugen und diesen z. B. zehnmal zu simulieren, werden zehn verschiedene Probleme gleicher Größe erzeugt.

In Tabelle 5 ist der Versuchsplan abgebildet, der nicht vollständig ist. Vielmehr werden bestimmte Problemgrößen ausgewählt. Insgesamt werden 300 Testprobleme erzeugt: 180 Stück mit 100 Konsumenten und 120 Stück mit 150 Konsumenten. Die verwendeten Deckungsbeiträge stammen aus einer Gleichverteilung. Die Simulationsstudie zeigt einen Vorteil des Genetischen Algorithmus mit zufälliger Startpopulation gegenüber Beam-Search von durchschnittlich 9.62 % für 100 Konsumenten und 7.94 % für 150 Konsumenten. Dieses Ergebnis deckt sich zumindest in der Tendenz mit den Ergebnissen von Nair et al. (1995). Ein Vergleich der Stärke der Unterschiede zwischen beiden Studien ist nicht möglich, da sich nicht nur die betrachteten Problemgrößen unterscheiden, sondern auch die verwendeten Parameter für den Genetischen Algorithmus. Weiterhin zeigt sich, dass der Hybrid zwischen Genetischem Algorithmus und Beam-Search zwar Beam-Search überlegen ist, nicht jedoch dem Genetischen Algorithmus mit rein zufälligen Startlösungen. So hat der Hybrid einen Vorteil in der durchschnittlichen Lösungsqualität gegenüber Beam-Search von 8.53 % bei 100 Konsumenten und 7.02 % bei 150 Konsumenten. Damit liegt der Hybrid knapp hinter der Lösungsqualität des Standardalgorithmus. Dieses Ergebnis ist nicht verwunderlich, sondern im Gegenteil zu erwarten, da durch die Beam-Search-Produktlinien bereits gute Lösungen in der Startpopulation existieren, wodurch der Genetische Algorithmus von vornherein in Richtung lokaler Optima von Beam-Search ge-

lenkt wird. Findet der Genetische Algorithmus also nicht zufällig durch Mutation eine Region, in der bessere Lösungen zu finden sind, verbessert er die Lösungen von Beam-Search höchstens lokal.

Shi et al. (2001) wenden ihren Nested-Partitions-Algorithmus auf das deterministische Marktanteilsmaximierungsproblem an, den sie in Shi/Ólafsson (2000) entwickelten. Das Besondere an diesem Algorithmus ist, dass er in Verteilung gegen ein globales Optimum konvergiert (Shi/Ólafsson 2000, S. 396/397). Der Algorithmus kann grob in vier Phasen unterteilt werden: das Partitionieren des Lösungsraums, einem Zufallsmechanismus zur Auswahl der weiteren Partitionen, der Auswahl von vielversprechenden Partitionen sowie einem Backtrackingmechanismus zur Anpassung der Wahrscheinlichkeiten für die Auswahl vielversprechender Partitionen. Um die Konvergenzgeschwindigkeit zu erhöhen, werden Heuristiken in die Zwischenschritte eingebunden: die Greedy-Heuristik, die Divide&Conquer-Heuristik, die Dynamic-Programming-Heuristik und ein Genetischer Algorithmus. Diese Heuristiken werden zum einen mit dem Nested-Partition-Algorithmus kombiniert und zum anderen als Benchmark für den Nested-Partition-Algorithmus verwendet. Die globale Suche wird über die Partitionen abgebildet, wohingegen die lokale Suche durch das Schätzen von vielversprechenden Eigenschaftsausprägungen implementiert wird (Shi et al. 2001, S. 1684). Ein großes Problem des Nested-Partition-Algorithmus ist, dass er zwar theoretisch in Verteilung gegen ein globales Optimum konvergiert, im praktischen Einsatz jedoch, im Gegensatz zu den exakten Verfahren von Camm et al. (2006), Belloni et al. (2008b) und Wang et al. (2009), ohne nachträgliche Enumeration nicht entschieden werden kann, ob es sich tatsächlich um das globale Optimum handelt (Shi et al. 2001, S. 1687/1688). Aus diesem Grund wird dieser Algorithmus in dieser Arbeit als Heuristik behandelt, da der Algorithmus nach bestimmten Abbruchkriterien determiniert wird, ohne, dass er notwendigerweise gegen das Optimum konvergiert. In Tabelle 4 wird dennoch von einem exakten Verfahren gesprochen, da die gefundene Lösung des Nested-Partitions-Algorithmus für kleinere Probleme durch komplette Enumeration als Optimum verifiziert wurde. Der betrachtete Lösungsraum ist der bis zu diesem Zeitpunkt größte im Vergleich zu anderen Publikationen auf diesem Gebiet. Es werden bis zu 20 Eigenschaften mit jeweils bis zu 20 Eigenschaftsausprägungen mit bis zu vier Produkten pro Produktlinie untersucht. Die Simulation gliedert sich in zwei Teile: zunächst wird das deterministische Marktanteilsmaximierungsproblem für den Spezialfall eines einzuführenden Produkts untersucht. Hierzu werden sieben Probleminstanzen erzeugt, wobei fünf Probleme von kleinem bis mittelgroßem Umfang sind und zwei Probleme besitzen einen großen Lösungsraum. Es werden alle in Tabelle 4 angegeben Algorithmen bis auf Beam-Search verwendet. Die Teilnutzenwerte für diese Simulationen werden aus einer uniformen Gleichverteilung gezogen. Der Algorithmus, der für den Einproduktfall die besten Zielfunktionswerte liefert, ist der Hybrid aus Nested-Partitions-Algorithmus in Kombination mit einem Genetischen Algorithmus sowie der Greedy-Heuristik. Ein abschließendes Urteil über die Überlegenheit dieses Hybrids gegenüber dem alleinstehenden Genetischen Algorithmus kann jedoch nicht ohne weitere Angaben der Parameterwahl des

Genetischen Algorithmus erfolgen. Für die Erweiterung auf Produktlinien wird der Hybrid aus Nested-Partitions-Algorithmus in Kombination mit einem Genetischen Algorithmus sowie der Greedy-Heuristik gegen Beam-Search für bis zu vier Produkte in der Produktlinie für die gleichen Datensätze aus dem Einproduktfall verglichen. Daraus resultieren 21 neue Testproblem, die jeweils zehnmal gelöst werden, also insgesamt 210 Testprobleme (für die gesamte Studie: 7 Testprobleme für den Einproduktfall + 210 Testprobleme für Produktlinien = 217 Testprobleme insgesamt; siehe Tabelle 5 für Details). Der Hybrid zeigt sich Beam-Search gemessen an der Lösungsqualität als überlegen. Nichtsdestotrotz leisten Shi et al. (2001) einen wichtigen Beitrag zur Produktlinienoptimierung durch das Einführen eines neuen Verfahrens.

Tarasewich/McMullen (2001) entwickeln ein an das deterministische Marktanteilsmaximierungsproblem angelehntes Optimierungsmodell, in dem sie zusätzlich zu den Präferenzen der Konsumenten die Präferenzen der Produktdesigner, also der Unternehmen, einbeziehen. Um einen Vergleich für die Heuristiken zum globalen Optimum zu haben, halten die Autoren den Lösungsraum sehr klein (max. 59049 mögliche Produkte), um alle Probleminstanzen mittels kompletter Enumeration lösen zu können. Es werden Teilnutzenwerte, sowohl für die Konsumenten als auch die Produktdesigner, aus einer Normalverteilung und einer Gleichverteilung gezogen, wobei kein Einfluss auf die Algorithmenperformance festgestellt werden konnte. Für den verwendeten Genetischen Algorithmus werden als Parameterwahl lediglich die Populationsgrößen angegeben. Für Tabu-Search wird die Größe der Tabu-Liste angegeben. Sonst werden keine weiteren Implementierungsdetails für die Algorithmen gegeben. Ausführlich wird hingegen die Funktionsweise der Pruning-Heuristik beschrieben. Diese Heuristik bestimmt zunächst den Zielfunktionswert für jede Produktdesignercharakteristik und schneidet diejenigen Eigenschaftsausprägungen ab, die eine geringe Wahrscheinlichkeit besitzen, gute Lösungen zu liefern. Alle übrigen Level der Produktdesigns werden mittels kompletter Enumeration berechnet (Tarasewich/McMullen 2001, S. 64), womit eine Limitierung auf höchstens mittelgroße Probleme der Pruning-Heuristik sofort ersichtlich ist, da das Abschneiden nicht vielversprechender Äste zwar Rechenaufwand spart, dieser jedoch für größere Probleminstanzen zu gering ist. Für die in der Veröffentlichung genutzten Probleminstanzen zeigt sich die Pruning-Heuristik sowohl dem verwendeten Genetischen Algorithmus als auch Tabu-Search überlegen. Diese Ergebnisse sind allerdings nicht auf die in dieser Arbeit verwendeten Produktlinienoptimierungsprobleme verallgemeinerbar, da zum einen die Probleme viel zu klein sind, ein unübliches Optimierungsmodell genutzt wurde und die Parameterkonfigurationen nicht en Detail angegeben wurden.

Steiner/Hruschka (2002) erweitern den von Balakrishnan/Jacob (1996) vorgestellten Einzelprodukt-Ansatz mit Genetischen Algorithmen auf den Produktlinienfall und führen eine umfangreiche Monte-Carlo-Simulation zur Bestimmung des optimalen Gewinns unter Berücksichtigung von Konkurrenzsituationen durch, wobei sie auf eine probabilistische Logit-Entscheidungsregel zurückgreifen. Steiner/Hruschka (2003) ähnelt der vorangegangenen Publikation von Steiner/Hruschka (2002) stark und nutzt die Ergebnisse aus Steiner/Hruschka (2002).

Als Vergleichsverfahren, um die Performance des verwendeten Genetischen Algorithmus einordnen zu können, entscheiden sich die Autoren für die Greedy-Heuristik von Green/Krieger (1985). Die Zielfunktion ist eine Gewinnfunktion mit probabilistischer Entscheidungsregel (Logit). Der Gewinn einer Produktlinie wird ermittelt, indem zunächst die variablen Kosten eines Produkts über die Teilstückkosten für jedes Level addiert werden. Diese Kosten werden vom Verkaufspreis abgezogen und mit dem Produkt aus der Summe der Größe des entsprechenden Segments mit der Auswahlwahrscheinlichkeit eines Produkts für das entsprechende Segment multipliziert. Eine Summenbildung über alle Produkte der Produktlinie liefert den Gewinn für die Produktlinie. Aus dem Versuchsplan aus Tabelle 5 ergeben sich 81 verschiedene Faktorkombinationen, von denen die Autoren 69 auswählten, die noch mit kompletter Enumeration gelöst werden können. Jedes dieser 69 Testprobleme wird viermal gelöst. Somit ergeben sich insgesamt 276 Testprobleme für die Simulationsstudie. Die Greedy-Heuristik mit der der Genetische Algorithmus verglichen wird, wählt zunächst ein Produkt aus und evaluiert den Zielfunktionswert. Danach wird ein weiteres Produkt gesucht, dass den höchstens Zuwachs für die Produktlinie bringt. Dieser Vorgang wiederholt sich, bis die Produktlinie aus der gewünschten Anzahl von Produkten besteht. Der Minimalzielfunktionswert für den Genetischen Algorithmus liegt bei 96.66 % bzgl. des Optimums. Weiterhin fand der Genetische Algorithmus in 84.78 % der Fälle die optimale Lösung des zugrunde liegenden Problems. Gegenüber der Greedy-Heuristik zeigte sich der Genetische Algorithmus in 66 Fällen überlegen, in 25 unterlegen und somit gab es in 185 Fällen ein Unentschieden, was die Lösungsqualität angeht.

Balakrishnan et al. (2004) erweitern ihren vorherigen Ansatz (Balakrishnan/Jacob 1996), in welchem sie einen Genetischen Algorithmus auf die Produktlinienoptimierung anwenden und diesen mit Beam-Search vergleichen. Es sei angemerkt, dass die Autoren davon sprechen, dass zu diesem Zeitpunkt Beam-Search der Status-quo-Algorithmus von Nair et al. (1995) in Sachen Lösungsqualität darstellt (Balakrishnan et al. 2004, S. 1). Jedoch konnten 2001 bereits Alexouda/Paparizzos (2001) zeigen, dass der Genetische Algorithmus Beam-Search und einer Hybridisierung überlegen ist. Wesentliche Unterschiede lassen sich dennoch ausmachen: die Zielfunktionen unterscheiden sich, die Größe der Lösungsräume unterscheidet sich maßgeblich (siehe Tabelle 4) und die Art und Weise der Hybridisierung ist eine andere. Balakrishnan et al. (2004) nehmen die Beam-Search-Lösungen nicht nur mit in die Startpopulation des Genetischen Algorithmus, sondern implementieren diese zusätzlich in den Ablauf des Genetischen Algorithmus. Die Zielfunktion für den Marktanteil wird standardmäßig gebildet: jedem Konsumenten wird ein Status-quo-Produkt zugewiesen. Nun wird diejenige Produktlinie gesucht, die die meisten Konsumenten von ihrem Status-quo-Produkt zu einem neu angebotenen Produkt aufgrund eines höheren Nutzens wechseln lässt. Die Autoren versuchen die Vorzüge beider Verfahren zu kombinieren. So wird Beam-Search die Fähigkeit nachgesagt, den Lösungsraum weiträumig zu durchsuchen, wohingegen der Genetische Algorithmus gegen lokale Optima konvergieren kann. Eine große Einschränkung bei Beam-Search ist der Determinismus der Methode und die daraus resultierende Gefahr, Lösungen anzubieten, die weit entfernt vom Optimum sind.

Quelle	Faktor	Level						
Nair et al. (1995)	Eigenschaften	4	5	6				
	Eigenschaftsausprägungen	2	3	4				
	Produkte pro Produktlinie	2	3	4				
	Wettbewerber	1	2	3				
	Wiederholungen eines Problems	4						
Balakrishnan/Jacob (1996)	Eigenschaften	4	6	8				
	Eigenschaftsausprägungen	2	3	4	5			
	Produkte pro Produktlinie	1						
	Konsumenten	100	200	300	400			
	Wiederholungen eines Problems	4						
Thakur et al. (2000)	Eigenschaften	4	5	6				
	Eigenschaftsausprägungen	2	3	4				
	Produkte pro Produktlinie	2	3	4				
	Konsumenten	50	100	150				
	Wiederholungen eines Problems	10	5					
Alexouda/Paparizzos (2001)	Eigenschaften	3	4	6	6	7		
	Eigenschaftsausprägungen	3	4	5	6			
	Produkte pro Produktlinie	2	3					
	Konsumenten	100	200					
	Wiederholungen eines Problems	10						
Shi et al. (2001)	Eigenschaften	5	6	7	8	9	10	20
	Eigenschaftsausprägungen	5	6	7	8	9	10	20
	Produkte pro Produktlinie	2	3	4				
	Konsumenten	400						
	Wiederholungen eines Problems	10						
Steiner/Hruschka (2002)	Eigenschaften	3	4	5				
	Eigenschaftsausprägungen	2	3	4				
	Produkte pro Produktlinie	2	3	4				
	Wettbewerber	1	2	3				
	Wiederholungen eines Problems	4						
Steiner/Hruschka (2003)	Eigenschaften	3	4	5				
	Eigenschaftsausprägungen	2	3	4				
	Produkte pro Produktlinie	2	3	4				
	Wettbewerber	1	2	3				
	Wiederholungen eines Problems	4						
Balakrishnan et al. (2004)	Eigenschaften	7	9					
	Eigenschaftsausprägungen	3	4	5	6	7		
	Produkte pro Produktlinie	7						
	Konsumenten	200						
	Wiederholungen eines Problems	10						

(Fortsetzung der Tabelle auf der nächsten Seite)

Quelle	Faktor	Level				
Albritton/McMullen (2007)	Eigenschaften	5	8	9	12	
	Eigenschaftsausprägungen	5	10			
	Produkte pro Produktlinie	1				
	Konsumenten	200	250	300		
	Wiederholungen eines Problems	25				
Belloni et al. (2008b)	Eigenschaften	3	5	7		
	Eigenschaftsausprägungen	2	3	5	8	
	Produkte pro Produktlinie	3	4			
	Konsumenten	50	100			
	Wiederholungen eines Problems	10				
Tsafarakis et al. (2011)	Eigenschaften	5	9			
	Eigenschaftsausprägungen	4	8			
	Produkte pro Produktlinie	6	9			
	Konsumenten	200	700			
	Wettbewerber	5	8			
	Wiederholungen eines Problems	20				
Voekler et al. (2013)	Eigenschaften	4	8	12	16	20
	Eigenschaftsausprägungen	5	8	10		
	Produkte pro Produktlinie	1				
	Konsumenten	250	500	1000		
	Wiederholungen eines Problems	10				

Tabelle 5: Faktoren und Level der Versuchspläne für die Publikationen mit Simulationsstudien. (Quelle: Eigene Darstellung)

Es können Datensätze erzeugt werden, die diese Tatsache nachweisen (Nair et al. 1995, S. 780/781). Die stochastische Natur des Genetischen Algorithmus soll diesem Problem begegnen. Als Zielfunktion wird der Marktanteil maximiert. Da die Autoren explizite Lösungen für manche Probleme exakt mithilfe von *CPLEX*[5] bestimmen, müssen sie das klassische Marktanteilsmaximierungsmodell in ein äquivalentes Problem umformen, das für *CPLEX* geeignet ist. *CPLEX* verwendet für das Finden optimaler Lösungen ein Branch&Bound-Verfahren. Für die Simulationsstudien werden acht hybridisierte Verfahren aus Genetischem Algorithmus und Beam-Search betrachtet, die sich wie folgt zusammensetzen: zwei Methoden für die Sortierung der Eigenschaftsausprägungen, eine lexikografische (siehe Erläuterungen in Abschnitt 4.3 zu Balakrishnan et al. (2004)) und eine Standardversion. Des Weiteren gibt es eine Version, in der die initiale Population des Genetischen Algorithmus zufällig erzeugt und eine, in der die von Beam-Search ermittelte Produktlinie in die initiale Population des Genetischen Algorithmus eingepflegt wird. Weiterhin wird eine Version kreiert, in der der klassische Mutationsoperator verwendet wird und eine hybridisierte Version des Mutationsoperators, in dem die ermittelte

[5]Ein mathematisches Optimierungsprogramm des Unternehmens IBM.

Produktlinie von Beam-Search vorgehalten wird und zufällig entschieden wird, ob die zu mutierende Eigenschaft zufällig geändert wird oder mit der gleichen Eigenschaft der von Beam-Search ermittelten Produktlinie ersetzt wird. Die Autoren versprechen sich davon u. a., dass erstens, Produktlinien mit höheren Marktanteilen gefunden werden als nur mit dem Genetischen Algorithmus bzw. Beam-Search. Und zweitens, dass der Hybridalgorithmus schneller gute Produktlinien findet als der Genetische Algorithmus. Letzteres konnten die Autoren in der Simulationsstudie zeigen, ersteres nicht. Die Simulationsstudien gliedern sich in zwei Teile. Im ersten Teil werden acht Probleme betrachtet. Von diesen acht Problemen besitzen vier Stück sieben Eigenschaften mit folgenden Eigenschaftsausprägungen: $\{6,3,7,4,5,3,3\}$. Die anderen vier Probleme besitzen neun Eigenschaften mit den Eigenschaftsausprägungen: $\{7,3,5,5,6,3,3,7,5\}$. Diese Probleme werden mit jeweils sieben Produkten pro Produktlinie gelöst und haben somit rund $6.12 \cdot 10^{26}$ bzw. $1.88 \cdot 10^{38}$ mögliche Produktlinien als Lösungen. Diese acht Probleme werden jeweils zehnmal von den eingesetzten Algorithmen gelöst. Weiterhin wird mithilfe von *CPLEX* versucht, die optimalen Produktlinien für jedes der acht Probleme zu finden. Dies gelang nur bei der Hälfte der Probleme, wobei *CPLEX* jeweils zehn Stunden Zeit gegeben wurde, um die optimale Produktlinie zu finden. Hier zeigt sich, dass es zwar gelungen ist, auch Probleme mittlerer Größe einer optimalen Lösung zuzuführen, jedoch ist dieser Ansatz für den Einsatz in der Marktforschung nicht praktikabel, da erstens schnelle Entscheidungen getroffen werden müssen und Balakrishnan et al. (2004) davon berichten, dass viele Probleme auch nach einer Woche Berechnungszeit nicht optimal gelöst werden konnten. Man würde als Marktforschungsunternehmen also in einen kostenintensiven kommerziellen Solver investieren, der nicht garantieren kann, eine optimale Produktlinie in angemessener Zeit zu finden.

Im zweiten Teil der Simulationsstudie werden insgesamt 800 Testprobleme mit acht verschiedenen Versionen des Genetischen Algorithmus gelöst. Die Teilnutzenwerte für die Konsumenten und die Eigenschaftsausprägungen werden aus einer Gleichverteilung im Intervall zwischen 0 und 1 erzeugt. Jedem Konsumenten wird ein zufälliges Status-quo-Produkt zugeordnet. Die Simulationsergebnisse zeigen zunächst, dass alle Versionen des Genetischen Algorithmus Beam-Search deutlich überlegen sind. In 95.93 % aller Testprobleme war Beam-Search unterlegen oder genauso gut – in 82.81 % aller Durchläufe war Beam-Search den Versionen des Genetischen Algorithmus unterlegen. Weiterhin zeigt die Varianzanalyse der Faktoren über alle Testprobleme, dass die Standardimplementierung des Genetischen Algorithmus (keine lexikografische Sortierung der Eigenschaften, zufällige initiale Population, Standardmutationsoperator) im Mittel die höchsten Marktanteile liefert (Balakrishnan et al. 2004, S. 10). Den einzigen Vorteil, den die hybridisierten Versionen des Genetischen Algorithmus haben, ist, dass diese Algorithmen schneller konvergieren und somit eine kürzere Laufzeit haben. Diese Tatsache ist wenig überraschend, da durch das Hinzufügen von schon akzeptablen Produktlinien, die von Beam-Search kommen, der Genetische Algorithmus schon in die Richtung eines lokalen Optimums „geschubst“ wird. Der Nachteil ist, dass Regionen des Suchraums mit besseren Produtklinien nicht „angesteuert“ werden können. Die Hybridisierung des Genetischen Algorithmus mit

Beam-Search brachte insgesamt keine besseren Produktlinien für die Marktanteilsmaximierung hervor. Alle hybridisierten Algorithmen schnitten insgesamt sogar schlechter ab als die Standardimplementierung des Genetischen Algorithmus (für Details zur Auswertung siehe Abschnitt 4.3). Aufgrund der deutlichen Simulationsergebnisse, deren Aussagekraft aufgrund der Vielzahl an Testproblemen gegeben ist, ist die Standardimplementierung des Genetischen Algorithmus zu bevorzugen.

Fruchter et al. (2006) nutzen einen Genetischen Algorithmus zur Optimierung des Gewinns eines Unternehmens für ein Einzelprodukt. Eine Erweiterung der bestehenden Modelle adressiert die Kannibalisierung von angebotenen Produkten in der Form, dass während des Optimierungsprozesses Informationen der Konsumenten über ihre Zahlungsbereitschaft inkludiert werden, so dass Konsumenten mit einer höheren Zahlungsbereitschaft nicht dasjenige Produkt wählen, das für Konsumenten mit einer geringeren Zahlungsbereitschaft eingeführt wurde. Dazu nutzen die Autoren einen Mechanismus, der eine abweichende Bepreisung der Produkte während der Optimierung zulässt. Es werden neue Preise, die zwischen den ursprünglichen liegen, zugelassen, wodurch sich allerdings der Suchraum während der Optimierung deutlich vergrößert, da die Anzahl der Preislevel steigt. Um diesem Problem zu begegnen, nutzen die Autoren einen Genetischen Algorithmus in Verbindungen mit Techniken aus der linearen Optimierung. Das verwendete Modell zur Gewinnoptimierung weicht in seinen Nebenbedingungen deutlich von den in dieser Arbeit betrachteten Modellen ab. Da dieses Modell keine weitere Beachtung in späterer Literatur findet und es ohne mathematische Beschreibung nicht sinnvoll ist, die Details darzulegen, wird hier nur eine allgemeine Beschreibung gegeben. Für die Details zum Modell sei auf Fruchter et al. (2006, S. 229 ff.) verwiesen. Nachdem der Genetische Algorithmus eine nicht optimale Produktlinie gefunden hat, werden die den Konsumenten zugeordneten Marken festgehalten und die optimalen Preise mittels linearer Optimierung bestimmt. Zum Testen ihres Modells führten Fruchter et al. (2006) eine Conjointanalyse mit 61 Probanden für Laptops durch. Sie verwendeten für die Studie sechs Eigenschaften mit zwei bis drei Eigenschaftsausprägungen, was in einer Gesamtzahl von 144 Produktkonfigurationen resultiert. Für die Simulationsstudie wurden drei simulierte Datensätze erzeugt. Die Autoren können zeigen, dass der Genetische Algorithmus gute Produktlinien in den simulierten als auch in dem realen Datensatz findet. Ein Vergleich verschiedener Algorithmen zur Lösung des Optimierungsproblems findet allerdings nicht statt.

Camm et al. (2006) präsentierten nach Balakrishnan et al. (2004) den ersten Versuch zum optimalen Lösen der deterministischen Marktanteilsmaximierung. Hierfür wurde ein hoch spezialisierter Algorithmus entwickelt, der die Lagrange-Relaxation mit Branch&Bound-Verfahren verbindet, um die optimale Lösung in angemessener Zeit zumindest für kleinere Optimierungsprobleme zu finden (Camm et al. 2006, S. 435). Um den Lösungsraum relativ klein zu halten, wird sich auf den Einproduktfall beschränkt. Die Untersuchung untergliedert sich in zwei Bereiche: eine Untersuchung auf Basis zweier realer Conjointdatensätze mit 294912 und 60466176

möglichen Produkten sowie 96 simulierter Conjointdatensätze mit bis zu ca. $7.21 \cdot 10^{16}$ möglichen Lösungen. Zur Konstruktion eines realistischen Status-quo-Marktes werden zunächst 500 bzw. 1000 Produkte erzeugt, die dafür infrage kommen. Hierin befinden sich auch Produkte, die nicht bereits auf dem Markt angeboten werden, um ein ideales Produkt für die Konsumenten simulieren zu können. Das Set der möglichen Status-quo-Produkte wird sodann verkleinert, indem für jeden Konsumenten diejenigen Produkte infrage kommen, die einen bestimmten Nutzenwert übersteigen. Aus diesen übrig gebliebenen Produkten wird eines zufällig ausgewählt und als Status-quo-Produkt für den Probanden ausgewählt (Camm et al. 2006, S. 441). Des Weiteren wird der Einfluss von Unsicherheiten in den Teilnutzenwerten anhand eines realen Conjointdatensatzes untersucht. Hierbei zeigte sich, dass das optimale Produkt aus der Punktschätzung der Teilnutzenwerte zum größten Teil wiedergefunden wurde. Lediglich in Eigenschaften, deren Eigenschaftsausprägung näher beieinander lagen, wurden auch regelmäßig andere Eigenschaftsausprägungen als optimale Eigenschaftsausprägung gefunden (Camm et al. 2006, S. 446).

Albritton/McMullen (2007) wenden die Ant-Colony-Optimization auf die deterministische Marktanteilsmaximierung für den Einproduktfall an. Die Ant-Colony-Optimization bildet das Verhalten bei der Nahrungssuche von in Sozialverbänden lebenden Insekten, hier Ameisen, nach. Eine Besonderheit im verwendeten Modell sind die Teilnutzenwerte der Konsumenten, die sich über die Zeit ändern. Die Probleminstanzen werden so klein gewählt, dass die von der Ant-Colony-Optimization gefundenen Lösungen gegen das Optimum gebenchmarkt werden kann. Da über mehrere Zeitschritte optimiert wird, in denen die Teilnutzenwerte der Konsumenten in jedem Zeitschritt durch einen Diffusionsprozess verändert werden, wobei die Eigenschaften und die Eigenschaftsausprägungen konstant gehalten werden (Albritton/McMullen 2007, S. 514). Ob es eine realistische Annahme ist, dass sich weder die Eigenschaften noch die Eigenschaftsausprägungen eines Produkts oder einer Dienstleistung innerhalb von 5-10 Zeitpunkten konstant bleiben, soll an dieser Stelle nicht diskutiert werden (Albritton/McMullen 2007, S. 509). Die in Tabelle 5 aufgeführten Faktoren und Level bilden keinen vollständigen Versuchsplan. Es wurden vier Problemgrößen selektiert und mit mehreren Parametern der Ant-Colony-Optimization kombiniert, wobei jede Probleminstanz jeweils 25-mal gelöst wurde. Daraus resultieren 4200 Beobachtungen, die die Autoren benutzen, um über eine Monte-Carlo-Simulation die besten Parameter für die Ant-Colony-Optimization zu finden. Die Simulationsstudie zeigt, dass die Ant-Colony-Optimization Lösungen nahe des mit kompletter Enumeration berechneten Optimums findet. Details zu den Parametern sowie der Implementierung der Ant-Colony-Optimization können in Abschnitt 4.5 eingesehen werden.

Belloni et al. (2008b) bieten einen umfassenden Vergleich von Heuristiken zur Lösung von Produktlinienoptimierungsproblemen sowie einen Ansatz zur Sicherstellung des globalen Optimums mithilfe einer Kombination aus Lagrange-Relaxation und eines Branch&Bound-Verfahrens. Eine Kombination aus Lagrange-Relaxation und Branch&Bound-Verfahren wurde vor-

her bereits von Camm et al. (2006) vorgeschlagen. Der Unterschied zwischen beiden Verfahren liegt im zugrunde liegenden Optimierungsproblem. Während Camm et al. (2006) sich der Marktanteilsmaximierung für ein einzelnes Produkt widmen, untersuchen Belloni et al. (2008b) ein Gewinnmaximierungsproblem für drei bis fünf einzuführende Produkte für eine Produktlinie. Es ist jedoch anzumerken, dass das Gewinnmaximierungsmodell das Marktanteilsmaximierungsmodell als Spezialfall enthält (Belloni et al. 2008b, S. 1548). Wang et al. (2009) erweitern das Marktanteilsmaximierungsmodell von Camm et al. (2006) um Produktlinien. Auch dieses Modell ist ein Spezialfall des Modells von Belloni et al. (2008b), wird jedoch durch Umformung in ein anderes bekanntes Optimierungsproblem gelöst. Details zu diesen exakten Verfahren befinden sich im Abschnitt 4.2. Die verwendeten Heuristiken werden in drei Gruppen eingeteilt. Die erste Gruppe von Heuristiken manipuliert während des Optimierungsprozesses die Attribute der einzelnen Produkte in der Produktlinie. Die zweite Gruppe von Heuristiken verwendet vollständige Produkte, die nach bestimmten Kriterien innerhalb der Produktlinie ausgetauscht werden. In die dritte Gruppe fallen diejenigen Heuristiken, die Produkte mit unvollständigen Eigenschaftsausprägungen erzeugen und evaluieren. Der ersten Gruppe werden die folgenden Heuristiken zugeordnet: Genetische Algorithmen (Details in Abschnitt 4.3), Simulated Annealing (Details in Abschnitt 4.6) und Coordinate-Ascent („one-opt“, „two-opt“, „three-opt“) als einfachste Methoden, die entweder eine, zwei bzw. drei Eigenschaftsausprägungen einer Produktlinie gleichzeitig verändern. Aufgrund der Definition der lokalen Nachbarschaft und der Tatsache, dass jede Veränderung einer Eigenschaftsausprägung akzeptiert wird, falls sich der Zielfunktionswert verbessert, ist die Existenz eines lokalen Optimums gesichert. In die zweite Gruppe werden die folgenden Heuristiken einsortiert: die Greedy-Heuristik (Green/Krieger 1985), Divide-and-Conquer-Heuristik (Green/Krieger 1993) und die Product-Swapping-Heuristik (Green/Krieger 1985). Die Greedy-Heuristik beginnt bei der Optimierung eines einzelnen Produkts und fügt in jedem Schritt ein weiteres Produkt aus einer Menge infrage kommender Produkte zur Produktlinie hinzu, was den Gewinn steigen lässt. Diese Methode findet nicht zwangsläufig ein lokales Optimum, da z. B. das erste Produkt kein lokales Optimum für die anschließenden Produkte darstellt (Green/Krieger 1985, S. 1546). Die Divide-and-Conquer-Heuristik teilt die Produktlinie in Gruppen von Eigenschaftsausprägungen ein. Die Autoren definieren ein einzelnes Produkt als Gruppe. Eine zufällige Produktlinie wird erzeugt, wobei alle Produkte bis auf eines konstant gehalten werden. Das einzelne Produkt wird enumeriert bis kein besserer Zielfunktionswert mehr gefunden wird. Das gleiche Vorgehen geschieht nachher mit den anderen Produkten in der Produktlinie. Insofern ist diese Heuristik lokal optimal, da die Nachbarschaftsdefinition hier aus einem vollständigen Produkt besteht. Die Product-Swapping-Heuristik beginnt mit einer zufälligen Produktlinie. Danach wird jedes Produkt in der Produktlinie nacheinander getestet, ob es durch ein neu erzeugtes Produkt ersetzt werden kann, das den Gewinn der gesamten Produktlinie erhöht. Ist dies möglich, wird ein altes durch das neue Produkt ersetzt und ein neues Produkt wird wieder auf die gleiche Art getestet usw. Die Existenz eines lokalen Optimums ist gesichert, da hier alle Produktlinien als Nachbarschaft definiert werden, die sich um genau ein Produkt unterscheiden. In die drit-

te Gruppen werden die Dynamic-Programming-Heuristik, die Beam-Search-Heuristik (kurze Beschreibung: siehe Fußnote 3 und 4 auf S. 71) und die Nested-Partitions-Heuristik (Shi et al. 2001). Die Nested-Partitions-Heuristik wird insofern angepasst, als dass kein Genetischer Algorithmus zum Finden von vielversprechenden Regionen wie in Shi et al. (2001) verwendet wird, sondern die Coordinate-Ascent-Methode mittels „one-opt“(siehe oben). Für die Evaluierung des exakten Verfahrens sowie der Heuristiken werden zwei Untersuchungen durchgeführt. Zum einen wird ein realer Conjointdatensatz über Kuriertaschen mit 324 Konsumenten verwendet. Der Aufbau der Monte-Carlo-Simulation kann aus Tabelle 5 entnommen werden. Es sei erwähnt, dass kein vollständiger Versuchsplan erstellt worden ist. Aus den 24 möglichen Faktor-Level-Kombinationen ($3 \cdot 4 \cdot 2$) wurden zwölf Testprobleme ausgewählt und mit den Heuristiken und dem exakten Verfahren gelöst. Es wird vermutet, dass die simulierten Testprobleme deutlich kleiner sind als das reale Problem, da für das reale Problem eine Woche Rechenzeit für das exakte Verfahren benötigt und für die simulierten Testprobleme eine durchschnittliche Laufzeit von ca. 11 Minuten angegeben wird (Belloni et al. 2008b, S. 1550). Bei den Heuristiken zeigt sich gegenüber vorherigen Publikationen ein einheitliches Bild: so konnte (z.T. wiederholt) bestätigt werden, dass Genetische Algorithmen der Dynamic-Programming-Heuristik, der Greedy-Heuristik sowie Beam-Search überlegen ist (Belloni et al. 2008b, S.1549). Für die Heuristiken zeigt sich ein eindeutiges Bild: die zwei Heuristiken, die am besten im Sinne der durchschnittlichen Lösungsqualität abschneiden, sind Simulated Annealing, dicht gefolgt vom verwendeten Genetischen Algorithmus. Aufgrund der eher kleinen betrachteten Probleme lässt sich kein abschließendes Urteil darüber fällen, ob Simulated Annealing dem Genetischen Algorithmus für die deterministische Gewinnmaximierung überlegen ist, da sie bzgl. ihrer Performance zu nah beieinander liegen.

Wang et al. (2009) entwickelten einen Branch&Price-Algorithmus zur Lösung der deterministischen Marktanteilsmaximierung und erweitern damit das Vorgehen von Camm et al. (2006) für Produktlinien. Dieses Modell ist ein Spezialfall der von Belloni et al. (2008b) verwendeten deterministischen Gewinnmaximierung, jedoch mit einer anderen Lösungsstrategie. Wang et al. (2009) lösen die deterministische Marktanteilsmaximierung für viele Problemgrößen in angemessener Zeit, vor allem die simulierten Datensätze. Für die zwei verwendeten realen Conjointdatensätze ist das Bild nicht einheitlich, da die Performance des Branch&Price-Algorithmus u. a. direkt von der Anzahl der Konsumenten abhängt. Für mehr Details sei auf Abschnitt 4.2 verwiesen. Eindeutig konstatieren lässt sich die Überlegenheit des Branch&Price-Algorithmus gegenüber dem kommerziellen Solver IBM CPLEX, der nur für einen kleinen Teil der Datensätze das Optimum innerhalb der maximal vorgegebenen Zeit finden kann, wohingegen der Branch&Price-Algorithmus für fast alle Probleminstanzen ein Optimum findet. Wie auch Camm et al. (2006) und Wang et al. (2009) testen die Autoren das gefundene Optimum eines kommerziellen Datensatzes auf Robustness. Sie können zeigen, dass das Optimum für dieses Beispiel sehr robust gegenüber Messfehlern bzw. Unsicherheiten in den Teilnutzenwerten ist (Wang et al. 2009, S. 1725).

Tsafarakis et al. (2011) führen für die probabilistische Marktanteilsmaximierung die Particle-Swarm-Optimization ein und vergleichen diese mit den Ergebnissen eines Genetischen Algorithmus. Im weiteren Verlauf werden Konkurrenzreaktionen anderer Marktteilnehmer über ein Nash-Gleichgewicht modelliert und die Auswirkungen für den griechischen Milchmarkt untersucht. Der eigentlichen Monte-Carlo-Simulation geht eine Simulationsstudie zur Bestimmung der besten Parameterkonfiguration für die Particle-Swarm-Optimization voraus. Für die Monte-Carlo-Simulation des Algorithmenvergleichs wird kein vollständiger Versuchsplan mit den Angaben aus Tabelle 5 erzeugt, sondern acht Probleme ausgewählt, welche jeweils 20-mal gelöst wurden, was in 160 Testprobleme mündet. Der Status-quo-Markt wird zufällig erzeugt, wobei er aus 6-20 Produkten besteht. Die Ergebnisse zeigen, dass der verwendete Genetische Algorithmus der Particle-Swarm-Optimization in allen relevanten überlegen ist: die durchschnittlichen Zielfunktionswerte sind höher, die schlechteste Lösung in der Endpopulation hat einen höheren Zielfunktionswert als bei der Particle-Swarm-Optimization und die Standardabweichung ist für den Genetischen Algorithmus geringer (Tsafarakis et al. 2011, S. 19). Die Konkurrenzreaktionen werden nicht in der Simulationsstudie betrachtet, sondern im praktischen Beispiel mit realen Conjointdaten. Dabei stellen die Autoren fest, dass sich über die Hälfte der Produkte des Markts geändert haben, was mit der Marktdynamik erklärt wird (Tsafarakis et al. 2011, S. 20). Eine ausführliche Beschreibung der Particle-Swarm-Optimization sowie Details zur Implementierung können aus Abschnitt 4.4 entnommen werden. Genauere Angaben des verwendeten Genetischen Algorithmus können aus Abschnitt 4.3 entnommen werden.

Wang/Curry (2012) untersuchen ein anderes bedeutendes Problem der Produktlinienoptimierung. Aufgrund seiner Wichtigkeit für das Forschungsgebiet wurde diese Publikation in die Literaturanalyse inkludiert. Die Teilnutzenwerte für jedes Individuum werden in der Praxis vor allem durch die auswahlbasierte Conjointanalyse (CBC) mittels HB-Schätzung berechnet. Diese individuellen Teilnutzenwerte unterliegen jedoch Schätzfehlern – sie besitzen eine inhärente Unsicherheit, da es sich lediglich um Punktschätzungen einer stationären Markovkette handelt. Die Autoren untersuchen, ob trotz dieser Unsicherheiten in den Teilnutzenwerten eine optimale Lösung existiert, gegen die gefundenen Lösungen durch Benutzung von 1000 HB-Draws in Wahrscheinlichkeit konvergiert. Es wird also geschaut, ob die Optimierung über die Punktschätzung stabile, optimale Produkte erzeugt. Für diese Untersuchung wurde ein realer Conjointdatensatz über Bankkonten verwendet. Die Resultate zeigen, dass die „robusten“ Lösungen der exakten Lösung mittels Punktschätzern sowohl im In-Sample- als auch Out-of-Sample-Fall zumindest für das vorliegende Beispiel überlegen sind (Wang/Curry 2012, S. 825).

Voekler et al. (2013) führen die Bee-Colony-Optimization (Teodorović/Dell'Orco 2005) in die Produktoptimierung ein und vergleichen sie mit der Ant-Colony-Optimization von Albritton/McMullen (2007) für den Einproduktfall der deterministischen Marktanteilsmaximierung. Bee-Colony-Optimization bildet das Verhalten eines Bienenvolks bei der Nahrungssuche nach. Jede Biene repräsentiert eine Lösung des zugrunde liegenden Optimierungsproblems, wobei sie

ein Produkt konstruieren, indem sie von einer Eigenschaftsausprägung einer bestimmten Eigenschaft mit einer gewissen Wahrscheinlichkeit zu einer Eigenschaftsausprägung der nächsten Eigenschaft gelangen. Zurück im Bienennest teilen die Bienen die Informationen über die gefundenen Lösungen, wobei gute Lösungen die Auswahlwahrscheinlichkeiten der guten Eigenschaftsausprägungen verstärken. Eine detailliertere Beschreibung kann Teodorović/Dell'Orco (2005) oder Voekler et al. (2013) entnommen werden. In der Simulationsstudie (siehe Versuchsplan in Tabelle 5) zeigt sich die Ant-Colony-Optimization der Bee-Colony-Optimization überlegen, weshalb die Bee-Colony-Optimization in der vorliegenden Arbeit nicht weiter betrachtet wird.

Zusammenfassend lassen sich aus Tabelle 4 einige Auffälligkeiten ablesen. Die wichtigsten Zielfunktionen sind der Marktanteil und der Gewinn. In den meisten Fällen wird die First-Choice-Entscheidungsregel zur Modellierung der Entscheidungen der Konsumenten verwendet. Ein Großteil der Publikationen widmet sich Produktlinien, die den Einzelproduktfall als Spezialfall enthalten und somit flexibler einsetzbar sind. Shi et al. (2001) untersuchten Probleminstanzen mit der größten Anzahl zulässiger Produktlinien ($5.04 \cdot 10^{102}$ Stück) aller Publikationen. Für die Erzeugung der Teilnutzenwerte der Konsumenten wird zumeist die Gleichverteilung $[0, 1]$ verwendet. Genetische Algorithmen scheinen sehr gute Produktlinien zu erzeugen, da sie häufig als beste Heuristik aus den Performancevergleichen hervorgehen. In Belloni et al. (2008b) ist Simulated Annealing dem verwendeten Genetischen Algorithmus leicht überlegen, ob diese Unterschiede jedoch statistisch signifikant sind, wurde nicht untersucht. Seit 1995 wurden sowohl anspruchsvollere Heuristiken entwickelt und auf Produktlinienoptimierungsprobleme angewendet als auch hoch spezialisierte exakte Verfahren, die sich eine Kombination aus Lagrange-Relaxation und Branch&Bound-Verfahren zunutze machen. Die Anwendung und Analyse für reale Conjointdatensätze ist ausbaufähig. Konkrete Produktlinien, die mithilfe der Heuristiken gefunden wurden, werden nicht auf ihre Praxistauglichkeit bzw. ihre Konsistenz und Sinnhaftigkeit überprüft. Der Fokus liegt auf der Performance der Heuristiken bzw. der exakten Verfahren, nicht auf den praktischen Implikationen konkreter Produktlinien.

4.2 Exakte Verfahren

4.2.1 Verfahren aus der Literatur

Um die zugrunde liegenden Produktlinienoptimierungsprobleme zu lösen, gibt es verschiedene Möglichkeiten.[6] Sehr kleine Probleme lassen sich mit kompletter Enumeration lösen, bei

[6] Der Nested-Partitions-Algorithmus von Shi et al. (2001) ist streng genommen ein exaktes Verfahren, da er zumindest in Verteilung gegen ein globales Optimum konvergiert. Da jedoch ohne Verifizierung der gefundenen

der der Gewinn bzw. der Marktanteil für jede mögliche Produktlinie berechnet wird und am Ende diejenige Produktlinie ausgewählt wird, die den höchsten Gewinn bzw. Marktanteil liefert. Eine weitere Möglichkeit ist die Nutzung kommerzieller oder freier Softwarelösungen, wie IBM CPLEX oder das *lpsolve*-Package in *R*. Balakrishnan et al. (2004) benutzten die Formulierung des Marktanteilsmaximierungsproblem von Kohli/Krishnamurti (1989, S. 1468), damit es mithilfe von IBM CPLEX exakt gelöst werden konnte. Von diesen in der Publikation acht untersuchten Probleminstanzen konnten jedoch lediglich vier innerhalb von 10 Stunden optimal gelöst werden. Bei den anderen Probleminstanzen wurde IBM CPLEX nach 10 Stunden terminiert und die beste gefundene Lösung wurde berichtet. Dabei zeigte sich, dass IBM CPLEX lediglich Lösungen fand, deren Zielfunktionswert zwischen 60-70% von der besten gefundenen Lösung der besten Heuristik betrug. Die Autoren räumen ein, dass sie die meisten Probleme selbst innerhalb von 600000s (166.67 Stunden bzw. 6.94 Tage) nicht optimal lösen konnten (Balakrishnan et al. 2004, S. 8). Für den Einsatz in der wissenschaftlichen oder unternehmerischen Praxis ist dies inakzeptabel, weshalb die Autoren schlussendlich auf Heuristiken zurückgreifen. IBM CPLEX verwendet das Branch&Bound-Verfahren zum Durchsuchen des Lösungsbaums. Obwohl es Balakrishnan et al. (2004) gelang, zwei Probleminstanzen mit ca. $1.88 \cdot 10^{38}$ möglichen Produktlinien optimal lösen, müssen diese Ergebnisse kritisch betrachtet werden. Es liegt die Vermutung nahe, dass diese Probleminstanzen eine günstige Struktur aufweisen oder IBM CPLEX gelangte zufällig schnell in die entscheidenden Äste des Branch&Bound-Baums. Für die vorliegende Arbeit wurde das deterministische Marktanteilsmaximierungsproblem gemäß Balakrishnan et al. (2004) mit IBM CPLEX, den *R*-Packages *lpsolve*, *lpsolveapi* sowie *cplexapi* implementiert. Trotz der Erzeugung mehrerer Probleminstanzen ist es nicht gelungen, diese für ca. 10^{12} mögliche Produktlinien innerhalb von 24 Stunden exakt zu lösen. Auch Camm et al. (2006) versuchten es mit IBM CPLEX, doch auch sie scheiterten für kleine Probleminstanzen mit $2.95 \cdot 10^5$ und $60.5 \cdot 10^6$ möglichen Produktlinien für zwei reale Conjointanalysen (Camm et al. 2006, S. 440). Ein weiteres starkes Indiz für die nicht zulässige Verallgemeinerbarkeit der gefundenen exakten Lösungen ist, dass Wang/Curry (2012) für das Testen der Robustheit der exakten Lösungen auf Basis von Messfehlern bei der Schätzung der Teilnutzenwerte nur ein optimales Einzelprodukt berechneten, also der Lösungsraum klein gehalten wurde (1728 möglichen Produkte) durch die Einführung lediglich eines neuen Produktes in den Markt (Wang/Curry 2012, S. 821). Die Wahl einer derartig kleinen Probleminstanz ist dem Umstand geschuldet, dass die Autoren für die Überprüfung der Konvergenz in Wahrscheinlichkeit für die exakte Lösung unter Messfehlern über die 1000 Draws aus der hierarchischen Bayesschätzung für jeden Probanden optimieren müssen. Trotz der Entwicklung zweier hoch spezialisierter Algorithmen (Camm et al. 2006, Wang et al. 2009) zur exakten Lösung des deterministischen Marktanteilsmaximierungsproblems müssen die Autoren die Probleminstanz zur Überprüfung der Konvergenzeigenschaften der optimalen Lösung sehr klein halten, da bei größeren Probleminstanzen nicht sichergestellt ist, dass die exakte Lösung in angemessener Zeit gefunden

Lösung nicht entschieden werden kann, ob es sich tatsächlich um das globale Optimum handelt, wird der Nested-Partitions-Algorithmus in dieser Arbeit als Heuristik geführt. Für Details siehe hierzu Abschnitt 4.1.

werden kann. Belloni et al. (2008b, S. 1548) beschreiben für ihren realen Conjointdatensatz u. a., dass sie, um die Rechenzeit in einem angemessenen Rahmen zu halten, die Nutzenwerte aller möglichen 3584 Kuriertaschen noch vor der Optimierung berechnen. Nur aus diesem Grund ist es überhaupt möglich, das Problem für fünf Produkte in der Produktlinie ($= 4.9 \cdot 10^{15}$ mögliche Produktlinien) optimal zu lösen. Selbst mit dieser Hilfe benötigt der Algorithmus zum Finden der optimalen Lösung eine Woche (Belloni et al. 2008b, S. 1549). Eine weitere große Herausforderung dieses Ansatzes ist die von den Autoren beschriebene Modellierung und deren Implementierung, deren Schwierigkeit mit „sehr hoch“ deklariert wird (Belloni et al. 2008b, S. 1549).

Quelle	**Produktfall**	**Zielfunktion**	**max. Anzahl möglicher Produktlinien**	**Solver**	**vorgegebene Rechenzeit**	**#Probleme exakt gelöst**
Balakrishnan et al. (2004)	Produktlinie	Marktanteil	$1.88 \cdot 10^{38}$	IBM CPLEX	10 Stunden	4 von 8
Camm et al. (2006)	Einzelprodukt	Marktanteil	$2.95 \cdot 10^{5}$ – $7.21 \cdot 10^{16}$	IBM CPLEX, Branch&Bound mit Lagrange-Relaxation	keine Vorgabe (max.: ca. 40 min)	34 von 34
Belloni et al. (2008b)	Produktlinie	Gewinn	$56 – 5 \cdot 10^{15}$	IBM CPLEX, Branch &Bound mit Lagrange-Relaxation	1 Woche für reales Problem bzw. durchschnittlich 659s (ca. 11 Minuten) für simulierte Probleme	13 von 13
Wang et al. (2009)	Produktlinie	Marktanteil	$2.49 \cdot 10^{11}$ – $7.95 \cdot 10^{49}$	IBM CPLEX, Branch&Price mit Lagrange-Relaxation	60 Minuten für simulierte Daten; 6 Stunden für reale Datensätze	165 von 168
Wang/Curry (2012)	Einzelprodukt	Marktanteil	1728	IBM CPLEX	3 Stunden	1000

Tabelle 6: Einsatz exakter Verfahren in der Produktlinienoptimierung. Der Produktfall entscheidet darüber, ob ein einzelnes Produkt (Einproduktfall) oder mehrere Produkte (Mehrproduktfall) optimiert werden. Die Zielfunktion zeigt das zu optimierende Ziel an. (Quelle: Eigene Darstellung)

Ein weiterer Hinweis dafür, dass die Ergebnisse von Balakrishnan et al. (2004) nicht verallgemeinerbar sind, sind die nachfolgenden Publikationen, die selbst mit größerer Rechenleistung und hoch spezialisierten Algorithmen nur selten in der Lage waren, Probleme mit derartig vielen möglichen Produktlinien zu lösen. Siehe hierfür Tabelle 6.

Ein effektiverer Weg zum Finden optimaler Lösungen ist die Kombination von Branch&Bound-Verfahren mit Lagrange-Relaxation, die die Suche durch das ständige Aktualisieren der oberen und unteren Grenze des relaxierten Problems einschränkt, wodurch schneller als beim reinen Branch&Bound-Verfahren nicht vielversprechende Äste des Baums abgeschnitten werden können, um den Suchraum zu verkleinern. Dieser Ansatz wurde als erstes von Camm et al. (2006) verfolgt. Um die Machbarkeit eines solchen Ansatzes für das Produktlinienoptimierungsproblem zu zeigen und einen kleinen Lösungsraum zu erzeugen, beschränken sich die Autoren auf den Einproduktfall für deterministische Marktanteilsmaximierungsprobleme. Die Umwandlung des ursprünglichen Problems in ein duales Lagrangeproblem erreichen die Autoren, indem sie das Problem durch das Hinzunehmen der Präferenznebenbedingungen in die Zielfunktion und die Entscheidungsvariablen im offenen Kontinuum relaxieren. Damit optimale Lösungen in akzeptabler Zeit gefunden werden können, geben die Autoren eine Startlösung mittels eines Greedy-Algorithmus vor, um den Suchraum von vornherein zu verkleinern. Belloni et al. (2008b) nutzen für die deterministische Gewinnmaximierung zwar ebenso eine Kombination aus Lagrange-Relaxation und Branch&Bound-Verfahren, heben jedoch hervor, dass sich ihr Ansatz grundlegend unterscheidet (Belloni et al. 2008b, S. 1548). Bei dem verwendeten Modell handelt es sich um eine Verallgemeinerung der von Camm et al. (2006) verwendeten deterministischen Marktanteilsmaximierung (Belloni et al. 2008b, S. 1544/1545). Und somit auch um eine Verallgemeinerung der von Wang et al. (2009) verwendeten deterministischen Marktanteilsmaximierung mit mehreren Produkten, da Wang et al. (2009) das verwendete Modell von Camm et al. (2006) um Produktlinien erweitern. Dennoch ergeben sich einige gravierende Unterschiede, die sich in der Modellierung der Lagrange-Relaxation und der Anwendung des Branch&Bound-Verfahrens widerspiegeln. Zum einen werden Produktlinien und nicht Einzelprodukte betrachtet und zum anderen wird eine Gewinn- statt einer Marktanteilsfunktion als Zielfunktion verwendet. Daraus ergeben sich eine andere Problemformulierung, eine andere Konstruktion der Lagrange-Relaxation sowie eine andere Herangehensweise für den Aufbau des Branch&Bound-Suchbaums. Aus Tabelle 6 geht hervor, dass Belloni et al. (2008b, S. 1550) im Mittel ca. 11 Minuten für die Lösung eines Problems benötigen. Dies lässt darauf schließen, wenn für den realen Conjointdatensatz eine Woche Rechenzeit benötigt wurde, dass die zwölf simulierten Probleminstanzen deutlich kleiner gewesen sind. Eine genaue Angabe der verwendeten Problemgrößen findet nicht statt.

Der letzte bekannte Versuch, ein Produktlinienoptimierungsproblem durch hoch spezialisierte Algorithmen zu lösen, stammt von Wang et al. (2009). Aufbauend auf der Vorarbeit von Camm et al. (2006), die die deterministische Marktanteilsmaximierung für den Einproduktfall optimal

lösten, entwickeln Wang et al. (2009) einen Branch&Price-Algorithmus für die Erweiterung auf Produktlinien. Der Branch&Price-Algorithmus ist eine Weiterentwicklung des Branch&Bound-Algorithmus zur Lösungen von ganzzahligen Optimierungsproblemen. Für jede Verzweigung des Branch&Bound-Suchbaums werden Listen mit Produktlinien erzeugt, die nur einen kleinen Teil der zulässigen Menge von Produktlinien berücksichtigen, da es aufgrund der großen Anzahl von Produktlinien nicht möglich ist für jede Verzweigung alle möglichen Produktlinien zu evaluieren und zu speichern. Daher wird bestimmten Regeln folgend, die Menge an relevanten Produktlinien für eine entsprechende Verzweigung klein gehalten. Für das Finden von geeigneten oberen und unteren Grenzen für den Branch&Bound-Suchbaum wird die Zielfunktion entsprechend gemäß Lagrange relaxiert. Um dieses Vorgehen anwenden zu können, wandeln die Autoren das Modell der deterministischen Marktanteilsmaximierung in ein *Maximum coverage problem* von Church/ReVelle (1974, S. 103) um. Auch die deterministische Marktanteilsmaximierung für Produktlinien ist ein Spezialfall der von Belloni et al. (2008b) deterministischen Gewinnmaximierung (Wang et al. 2009, S. 1720). Somit hätte das Vorgehen für die Lösung des zugrunde liegenden Optimierungsproblems von Belloni et al. (2008b) angewendet werden können. Ein Laufzeit-Performancevergleich beider optimalen Lösungsalgorithmen wurde nicht gegeben. Als Vergleichssolver für ihren Branch&Price-Algorithmus nutzen die Autoren IBM CPLEX, welcher für 16 von 168 Probleminstanzen (statt 165 von 168 wie der Branch&Price-Algorithmus; zwei reale Conjointdatensätze mit jeweils 1-4 Produkten pro Produktlinie entsprechen acht Probleminstanzen zuzüglich 32 simulierten Datensätzen mit jeweils 1-5 Produkten pro Produktlinie entsprechen 160 Probleminstanzen) die optimale Lösung innerhalb von einer Stunde fand. Es muss vermutet werden, dass Wang et al. (2009) deshalb in der Lage sind, Probleme dieser Größe (siehe Tabelle 6) zu lösen, da sie einen der wichtigsten Faktoren, neben der Anzahl möglicher Lösungen, entschärft haben, der zu langen Laufzeiten der Algorithmen führt: die Anzahl der betrachteten Konsumenten. Die Autoren geben zwar eine Anzahl von 600 Konsumenten an, räumen direkt danach jedoch ein, dass sie diese in lediglich zwei Segmente unterteilen (Wang et al. 2009, S. 1722), was die Laufzeit der Algorithmen sehr verkürzt, da die Anzahl der Nebenbedingungen und der nötigen Kalkulationen erheblich mit der Anzahl der Konsumenten wächst. Dies zeigt sich vor allem bei den beiden realen Conjointdatensätzen, die verwendet werden. So wird das Zeitlimit von sechs Stunden mit 1131 Konsumenten für einen realen Conjointdatensatz bereits ab $6.81 \cdot 10^{14}$ möglichen Lösungen nicht mehr eingehalten und kein Optimum angegeben (Wang et al. 2009, S. 1723).

Somit zeigt sich in diesem Abschnitt, dass es möglich ist einige Produktlinienoptimierungsprobleme mit exakten Verfahren zu lösen. Jedoch zeigt sich ebenso, dass diese Ergebnisse nicht auf alle Produktlinienoptimierungsprobleme und deren Probleminstanzen anwendbar und somit verallgemeinerbar sind. Hoch spezialisierte Algorithmen bieten zwar einen signifikanten Geschwindigkeitsvorteil gegenüber kommerziellen und nicht kommerziellen Solvern, sind allerdings sehr schwer zu implementieren und bieten keine Garantie Produktlinienoptimierungsprobleme mit realen Problemgrößen in angemessener Zeit zu lösen, weshalb man nach wie vor

auf den Einsatz von Heuristiken angewiesen ist.

4.2.2 Exakter Algorithmus für die probabilistische Marktanteilsmaximierung

Eine Ausnahme unter den vier in dieser Arbeit betrachteten Zielfunktionen stellt die probabilistische Marktanteilsmaximierung dar. Sie kann mithilfe des freien Solvers *lpSolve*, der über eine Schnittstelle mit anderen Programmiersprachen (hier: GNU R) kommunizieren kann, in unter einer Sekunde exakt gelöst werden. Dazu sind einige Schritte im Vorhinein nötig, um das Optimierungsproblem in eine entsprechende Form zu überführen. Im Folgenden wird zuerst gezeigt, dass die probabilistische Marktanteilsmaximierung äquivalent zum Buyers'-Problem (Nutzenmaximierung aller Konsumenten) ist, dessen Formulierung sich z. B. in Kohli/Sukumar (1990, S. 1467) findet. Anschließend wird bewiesen, dass die optimale Produktlinie den R Produkten mit den höchsten Nutzenwerten über alle Konsumenten entspricht. Damit wird gezeigt, dass es zulässig ist, einen Algorithmus zu entwickeln, der in der ersten Iteration zunächst das Produkt mit dem maximalen Nutzen findet, es anschließend für die nächste Iteration als neue Nebenbedingung hinzufügt und somit von der Optimierung ausschließt. In der zweiten Iteration wird das Produkt mit dem zweithöchsten Nutzen berechnet. In der dritten Iteration werden die Produkte mit dem höchsten und zweithöchsten Nutzen von der Optimierung ausgeschlossen, indem sie wieder als neue Nebenbedingungen hinzugefügt werden. Dieser iterative Prozess findet solange statt, bis R Produkte mit dem höchsten Nutzen generiert wurden, indem iterativ R Optimierungen algorithmisch ausgeführt werden.

Die Basis des Algorithmus bildet die Äquivalenz zwischen der probabilistischen Marktanteilsmaximierung und dem Buyers'-Problem. Die Zielfunktion für die probabilistische Marktanteilsmaximierung aus Abschnitt 2.2.3.1 wurde folgendermaßen formuliert:

$$\sum_{j=1}^{R}\sum_{i=1}^{I} p_{ij} = \sum_{i=1}^{I} \frac{\sum_{j=1}^{R} u_{ij}^{\alpha}}{\sum_{j_M=1}^{J_M} u_{ij_M}^{\alpha}}, \quad \alpha \geq 0 \longrightarrow \max! \tag{4.1}$$

Hierbei bezeichnet u_{ij} den Nutzen des i-ten Konsumenten für das j-te Produkt. Die Menge J_M, $j_M = 1,...,J_M$ bezeichne alle auf dem Markt verfügbaren Produkte. Der Nenner des Bruchs besteht also aus den neu einzuführenden Produkten R, sowie den bisher verfügbaren Status-quo-Produkten, womit sich der Nenner darstellen lässt als:

$$\sum_{j_M=1}^{J_M} u_{ij_M}^{\alpha} = \sum_{j=1}^{R} u_{ij}^{\alpha} + \sum_{j_{SQ}=1}^{J_{SQ}} u_{ij_{SQ}}^{\alpha} \,, \forall i = 1,...,I,$$

mit $J_M = R + J_{SQ}$. Die Nutzenwerte der Produkte des Status-quo-Markts sind bekannt und können vor der Optimierung berechnet werden, womit diese Summe letztlich eine Konstante darstellt. Wenn diese Summe eine Konstante ist, reicht es, die folgende Summe zu maximieren:

$$\sum_{j=1}^{R}\sum_{i=1}^{I} u_{ij}^{\alpha} = \sum_{j=1}^{R}\sum_{i=1}^{I}\left(\sum_{k=1}^{K}\sum_{l=1}^{L_K}\beta_{ikl}x_{jkl}\right)^{\alpha} \,,$$

was für $\alpha = 1$ der Buyers'-Problem-Zielfunktion entspricht. Durch die Monotonie dieser Zielfunktion spielt der Parameter $\alpha \geq 0$ für die optimale Produktlinie keine Rolle, es gilt:

$$\arg\max_x \left(\sum_{j=1}^{R}\sum_{i=1}^{I}\left(\sum_{k=1}^{K}\sum_{l=1}^{L_K}\beta_{ikl}x_{jkl}\right)^{\alpha}\right) = \arg\max_x \left(\sum_{j=1}^{R}\sum_{i=1}^{I}\left(\sum_{k=1}^{K}\sum_{l=1}^{L_K}\beta_{ikl}x_{jkl}\right)\right) \,,$$

weshalb der Parameter $\alpha \geq 0$ o. B. d. A. für die weiteren Betrachtungen vernachlässigt werden kann. Dieses Monotonieverhalten und die Äquivalenz zum Buyers'-Problem gilt ebenso für die Logit-Entscheidungsregel, mit der die probabilistische Marktanteilsmaximierung ebenfalls durchgeführt werden kann.

Nachdem die Äquivalenz der probabilistischen Marktanteilsmaximierung zum Buyers'-Problem gezeigt wurde, soll im nächsten Schritt bewiesen werden, dass diejenige Produktlinie den Zielfunktionswert maximiert, die die R Produkte mit den höchsten Nutzenwerten über alle Konsumenten enthält. Hierfür wird die Zielfunktion umgeschrieben zu:

$$\sum_{j=1}^{R}\sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L_K}\beta_{ikl}x_{jkl} = \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L_K}\beta_{ikl}x_{1kl} + \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L_K}\beta_{ikl}x_{2kl} + ... + \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L_K}\beta_{ikl}x_{Rkl} \quad (4.2)$$

$$= \sum_{i=1}^{I} u_{i1} + \sum_{i=1}^{I} u_{i2} + ... + \sum_{i=1}^{I} u_{iR} \,. \quad (4.3)$$

Sei $U_j = \sum_{i=1}^{I} u_{ij}, j = 1,...,J$ der Nutzen des Produkts j über alle Konsumenten. Weiterhin sei $(\mathbb{U}, \geq)$ eine geordnete Menge der Nutzenwerte aller zulässigen Produkte über alle Konsumenten, wobei $\mathbb{U} = \{U_1, U_2, ..., U_J\}$ mit $U_1 \geq U_2 \geq ... \geq U_J$. Dann beschreibt die Teilmenge

$\mathbb{U}^R = \{U_1, U_2, ..., U_R\} \subseteq \mathbb{U}$ die Menge der Nutzenwerte der R Produkte mit den höchsten Nutzenwerten aller zulässigen Produkte, woraus folgt:

$$\max_x \left(\sum_{j=1}^{R} \sum_{i=1}^{I} \sum_{k=1}^{K} \sum_{l=1}^{L_K} \beta_{ikl} x_{jkl} \right) \underset{4.3}{\cong} \max_x \left(\sum_{i=1}^{I} u_{i1} + \sum_{i=1}^{I} u_{i2} + ... + \sum_{i=1}^{I} u_{iR} \right) \cong U_1 + U_2 + ... + U_R \ .$$

Das heißt, der Gesamtnutzen der besten Produktlinie entspricht der Summe der Produkte mit den höchsten Nutzenwerten über alle Konsumenten. □

Damit wurde die erste wichtige Eigenschaft für den exakten Algorithmus gezeigt. Die zweite wichtige Eigenschaft entsteht durch die Ausnutzung der Kommutativität der Summen in der Zielfunktion mit der die Abhängigkeit von i, also den Konsumenten, vor dem Optimierungsprozess aufgehoben werden kann. Sei $\beta_{kl} = \sum_{i=1}^{I} \beta_{ikl}, k = 1, ..., K; l = 1, ..., L_K$, dann gilt:

$$\sum_{j=1}^{R} \sum_{i=1}^{I} \sum_{k=1}^{K} \sum_{l=1}^{L_K} \beta_{ikl} x_{jkl} = \sum_{j=1}^{R} \sum_{k=1}^{K} \sum_{l=1}^{L_K} x_{jkl} \sum_{i=1}^{I} \beta_{ikl} = \sum_{j=1}^{R} \sum_{k=1}^{K} \sum_{l=1}^{L_K} \beta_{kl} x_{jkl} \ . \tag{4.4}$$

Das bedeutet, dass die Teilnutzenwerte jeder Eigenschaftsausprägung über alle Konsumenten aufaddiert werden können, wodurch ein einzelner Teilnutzenwert-Vektor für alle Konsumenten entsteht, der nur noch von den Eigenschaftsausprägungen abhängt. Das Aufheben dieser Abhängigkeit ist zentral für den exakten Optimierungsalgorithmus, da hierdurch die Komplexität der Zielfunktion für *lpSolve* soweit verringert werden kann, dass die Optimierungsrechnungen für beliebige Problemgrößen durchgeführt werden können.

Unter Zuhilfenahme der beiden gezeigten Eigenschaften der Zielfunktion der probabilistischen Marktanteilsmaximierung lässt sich ein Algorithmus angeben, der wie folgt funktioniert. Zunächst werden die Teilnutzenwerte gemäß 4.4 über alle Konsumenten für jede Eigenschaftsausprägung aufsummiert (Abbildung 6, Zeile 1). Über den entstehenden Teilnutzenwert-Vektor β_{kl} wird optimiert. Dann werden die Nebenbedingungen gemäß 2.18, 2.19 und 2.20 aus Abschnitt 2.2.3.1 modelliert (Abbildung 6, Zeile 2). Anschließend beginnt der iterative Teil des Algorithmus. In der ersten Iteration wird das Produkt mit dem höchsten Nutzenwert U_1 berechnet (Abbildung 6, Zeile 4), wobei die vierte Nebenbedingung NB_4 nicht berücksichtigt wird, da sie erst ab der zweiten Iteration von Bedeutung ist. Das Produkt $j = 1$ mit dem höchsten Zielfunktionswert wird in einem Vektor *Produktlinie* gespeichert, der am Ende des Algorithmus die R Produkte mit den höchsten Zielfunktionswerten enthält. In Zeile 6 des Algorithmus wird eine neue Nebenbedingung NB_4 erzeugt, die das gefundene Produkt $j = 1$ mit dem höchsten Zielfunktionswert von der Optimierung ausschließt. In der zweiten Iteration werden Produkte mit Zielfunktionswerten verboten, die einen Zielfunktionswert von U_1 besitzen, in

dem die neue Nebenbedingung NB_4 zu Zeile 4 hinzugefügt wird. Ergo, besitzt in der zweiten Iteration der Optimierung dasjenige Produkt den höchsten Zielfunktionswert, welches global den zweithöchsten Zielfunktionswert U_2 besitzt. In der dritten Iteration werden alle Produkte verboten, die mindestens einen Zielfunktionswert von U_2 besitzen, also die Produkte mit den zwei höchsten Zielfunktionswerten, usw. Der Algorithmus endet, wenn die R Produkte mit den R höchsten Zielfunktionswerten bestimmt wurden, aus denen sich am Ende die Produktlinie zusammensetzt. Schlussendlich werden die Produkte als Argumente in die probabilistische Marktanteilsmaximierungszielfunktion eingesetzt (Abbildung 6, Zeile 8) und somit der global optimale Marktanteil der optimalen Produktlinie berechnet. Am Ende wird der maximale Marktanteil und die optimale Produktlinie ausgegeben (Abbildung 6, Zeile 9). Der beschriebene Algorithmus ist in Abbildung 6 illustriert.

Algorithmus 1: Exakter Algorithmus für M_{prob}

Result: Optimale Produktlinie für M_{prob}
Data: Teilnutzenwertmatrix β

1 Aufsummieren der Teilnutzenwerte: $\beta_{kl} = \sum_{i=1}^{I} \beta_{ikl}, k = 1,...,K; l = 1,...,L_K$
2 $NB_1, NB_2, NB_3 \leftarrow$ Nebenbedingungen 2.18, 2.19 und 2.20 für M_{prob} erzeugen
3 **for** $j \leftarrow 1$ **to** R **do**
4 $\quad Max[j] \leftarrow$ lpSolve: $\max_x \left(\sum_{k=1}^{K} \sum_{l=1}^{L_K} \beta_{kl} x_{jkl} \mid NB_1, NB_2, NB_3, (NB_4)_{\text{if } j \neq 1} \right)$
5 $\quad Produktlinie[j] \leftarrow$ Produkt j
6 $\quad NB_4 \leftarrow \sum_{k=1}^{K} \sum_{l=1}^{L_K} \beta_{kl} x_{jkl} < Max[j]$
7 **end**
8 setze *Produktlinie* in M_{prob} ein und erhalte M_{prob}^{opt}
9 **return** M_{prob}^{opt} *und Produktlinie*

Abbildung 6: Aufbau des exakten Algorithmus für die probabilistische Marktanteilsmaximierung. (Quelle: Eigene Darstellung)

Mit diesem exakten Algorithmus wird es in den Simulationsrechnungen möglich sein, zumindest für die probabilistische Marktanteilsmaximierung das globale Optimum zu bestimmen. Damit kann die Performance der Heuristiken am tatsächlichen Optimum gemessen werden. Die Krux des Algorithmus liegt in der Tatsache, dass die Teilnutzenwerte der Konsumenten auf einen einzigen Vektor zusammengestaucht werden können, wodurch die Geschwindigkeitsvorteile erst zum Tragen kommen und eine exakte Lösung möglich wird. Für die anderen Zielfunktionen in dieser Arbeit ist dies nicht möglich. Bei den Zielfunktionen mit deterministischer Entscheidungsregel muss für jeden einzelnen Konsumenten verglichen werden, ob wenigstens ein Produkt in der Produktlinie den Nutzen seines Status-quo-Produkts übersteigt. Eine Zusammenfassung der Präferenzen ist hier also nicht möglich. Ebenfalls nicht möglich ist das Zusammenstauchen der Zielfunktion der probabilistischen Gewinnmaximierung auf einen einzelnen

Vektor, da hierfür die Präferenzen zunächst mit den individuellen Gewichten $\omega_i, i = 1,...,I$ und den Teilstückdeckungsbeiträgen $d_{ikl}, i = 1,..,I; k = 1,...,K; l =,...,L_K$ multipliziert und dann zu einem einzelnen Vektor aufaddiert werden müssten, was aber nicht mehr zu einem globalen Optimum für diese Zielfunktion führen würde. Die Verhältnisse der Auswahlwahrscheinlichkeit p_{ij} eines Produkts j von Konsumenten i würden zerstört, wenn zunächst die Gewichte ω_i mit den Teilnutzenwerten β_{ikl} und dieses Ergebnis mit den Teilstückdeckungsbeiträgen d_{ikl} multipliziert werden würde, was nötig wäre, um einen einzelnen zu optimierenden Vektor zu erhalten. Der entstehende Vektor wäre nicht mehr äquivalent zum ursprünglichen Optimierungsproblem und würde somit „falsche Optima“ liefern.

Die *Forschungsfrage 1* kann damit positiv beantwortet werden.

4.3 Genetische Algorithmen in der Produktlinienoptimierung

Für die Lösung von kombinatorischen Optimierungsproblemen ist der Genetische Algorithmus vielfach erfolgreich angewendet worden. Als Unterklasse von kombinatorischen Optimierungsproblemen sind hier sowohl diskrete als auch binäre Optimierungsprobleme zu nennen. Klassische binäre Probleme sind bspw. das Rucksackproblem (*knapsack problem*) oder das Problem des Handlungsreisenden (*traveling salesman problem*) (vgl. Reeves 2003, S. 69 pp.). In diesem Unterkapitel wird zunächst eine Übersicht über die Literatur in der Produktlinienoptimierung auf Basis der Conjointanalyse gegeben, in welcher die verwendeten Operatoren aufgeschlüsselt und das Vorgehen der Autoren anschließend beschrieben wird.

4.3.1 Literaturübersicht

Genetische Algorithmen wurden im Bereich der Produktlinienoptimierung bisher am häufigsten angewendet. In Tabelle 7 sind die Quellen aufgeführt, die sich eines Genetischen Algorithmus bedienen, um eine bestimmte Zielfunktion zu maximieren. Dabei wird unterschieden, ob es sich um einen Ein- oder Mehrproduktfall handelt, also entweder nur ein einzelnes Produkt oder eine Produktlinie mit mindestens zwei Produkten optimiert wird. Des Weiteren werden die Strategien für die Reproduktions-, Crossover- und Mutationsmechanismen dargestellt. Außerdem wird die Art der Kodierung der Produkte bzw. Produktlinien erfasst. Der Grund hierfür liegt darin, dass binär kodierte Chromosomen, also hier die Eigenschaftsausprägungen mit 0 oder 1 kodiert, eine deutlich längere Laufzeit haben als multinomial kodierte Chromosom, hier die Eigenschaften, weil die Chromosomen bei binärer Kodierung generell länger sind und somit mehr Daten verarbeitet werden müssen. Dabei wird eine Eigenschaftsausprägung eines Produkts als Gen

interpretiert. Ein Chromosom bzw. eine Genkette repräsentieren ein ganzes Produkt bzw. eine ganze Produktlinie.

Quelle	**Kodierung**	**Selektion**	**Crossover**	**Mutation**	**Pop.-größe**	**Populations-wahrungs-mechanismus**	**Anmerkung**
Gutsche (1995)	binär	k.A.	k.A.	k.A.	k.A.	k.A.	kein Modell; keine Implementierung
Balakrishnan/Jacob (1995)	binär	Truncation	bis 5-point	frei wählbar	frei wählbar; max. 400	Emigration, Malthusian, modifizierter Malthusian	kein systematischer Vergleich
Balakrishnan/Jacob (1996)	binär	Truncation	uniform	attributweise	50, 100, 200	Malthusian	Mutationsrate besitzt keinen signifikanten Einfluss
Thakur et al. (2000)	k.A.	Truncation	1-point	attributweise	20	Emigration	keine Sensitivitätsanalyse der Parameter
Alexouda/Paparizzos (2001)	binär	Truncation	uniform	attributweise	150	Emigration	keine Simulationen für das Bestimmen geeigneter GA-Parameter
Shi et al. (2001)	k.A.	k.A.	k.A.	k.A.	k.A.	k.A.	20 % Mutationsrate; Simulationen für das Bestimmen geeigneter GA-Parameter
Tarasewich/McMullen (2001)	k.A.	k.A.	k.A.	k.A.	50, 100, 200	k.A.	4 Mutationsraten; GA-Parameter für Simulation nicht aufgeführt (außer Pop.gr)
Steiner/Hruschka (2002)	binär	Tournament	1-point	attributweise	30–250 in 20er Schritten	Emigration	5 Crossover-Wkt.; 3 Mutationsraten
Steiner/Hruschka (2003)	binär	Tournament	1-point	attributweise	150	k.A.	GA-Parameter für Simulation nicht aufgeführt
Balakrishnan et al. (2004)	multinomial	Truncation	uniform	attributweise	400	Emigration	keine Simulationen für das Bestimmen der GA-Parameter
Fruchter et al. (2006)	multinomial	Tournament	uniform	attributweise	k.A.	Emigration	keine Simulationsstudie; grundlegend anderer Aufbau der Strings
Belloni et al. (2008b)	k.A.	Truncation	1-point	5 %, attributweise	500, 2000	modifizierter Malthusian	1-Point-Crossover besser als Uniform-Crossover; Abbruch, wenn finale Population homogen
Tsafarakis et al. (2011)	multinomial	Truncation	uniform	4 %, attributweise	100	Malthusian	keine Simulationen für das Bestimmen der GA-Parameter

Tabelle 7: Einsatz Genetischer Algorithmen in der Produktlinienoptimierung. Der Produktfall entscheidet darüber, ob ein einzelnes Produkt (Einproduktfall) oder mehrere Produkte (Mehrproduktfall) optimiert werden. Die Zielfunktion zeigt das zu optimierende Ziel an. Die Kodierung entscheidet, ob die Chromosomen binär oder multinomial codiert wurden. Die Reproduktions-, Crossover- und Mutationsstrategien werden zusätzlich dargestellt. Abkürzungen: GA: Genetischer Algorithmus; Pop.größe: Populationsgröße; Pop.wahr.mech.: Populationswahrungsmechanismus. (Quelle: Eigene Darstellung)

Gutsche (1995) schlägt für die Preisoptimierung von Produktlinien einen Genetischen Algorithmus vor. Der Autor startet mit einer fest vorgebenenen Population, wobei er Population als

Menge aller Genketten bezeichnet. Eine Genkette besteht aus mehreren Genen, die binär kodiert sind, wobei die 1 dafür steht, dass ein Produkt eine Eigenschaftsausprägung besitzt und 0 steht dafür, dass ein Produkt die Eigenschaftsausprägung nicht besitzt. Die Population wird zufällig initialisiert und einem Fitnesstest unterzogen, was bedeutet, dass die Zielfunktionswerte jedes Chromosoms berechnet werden. Die Fitnesswerte stellen die Überlebenswahrscheinlichkeit eines einzelnen Chromosoms dar. Umso höher der Fitnesswert eines Chromosoms ist, desto höher ist die Überlebenswahrscheinlichkeit des Chromosoms für den nächsten Schritt und desto höher ist die Wahrscheinlichkeit für die Reproduktion ausgewählt zu werden. In der nächsten Phase, dem Crossover, werden zufällig zwei Chromosom ausgewählt, die „Eltern“, die jeweils einen Teil ihrer Informationen, also ihrer Gene, geben und daraus ein neues Chromosom, einen Nachkommen, erzeugen. Hierbei wird ein Ankerpunkt gewählt, der entscheidet, an welcher Stelle die Chromosomen geteilt und wieder zusammengesetzt werden. Genauere Angaben, z. B. wie der Ankerpunkt ausgewählt wurde, werden nicht gemacht. Der Nachkomme besitzt wieder einen Fitnesswert und ist dementsprechend besser oder schlechter angepasst. In der Mutationsphase werden einzelne Gene willkürlich gemäß einem Prozentsatz ausgewählt, und zufällig in 0 oder 1 geändert, um der Genetischen Diversität Rechnung zu tragen. Am Ende dieses Prozesses wird der Zyklus von vorn gestartet bis ein Abbruchkriterium erreicht wurde oder der Algorithmus gegen ein (lokales) Optimum konvergiert. Es sei darauf hingewiesen, dass der Autor lediglich die allgemeine Funktionsweise des Genetischen Algorithmus für Produktlinien beschreibt und keine Implementierungen stattfanden. Deshalb ist eine Bewertung der Güte des Algorithmus nicht möglich. Aus diesem Grund findet sich keine Angabe in der Tabelle 7 bei Reproduktion und Mutation, da hierfür keine konkreten Prozentzahlen genannt wurden (Gutsche 1995, S. 192 ff.).

Balakrishnan/Jacob (1995) liefern die erste Implementierung eines Genetischen Algorithmus für ein Produktoptimierungsproblem sowie einen ersten Vergleich mit einer anderen Heuristik zur Lösung solcher Probleme, der Dynamic-Programming-Heuristik von Kohli/Krishnamurti (1987; 1989). Die Autoren zeigen wie der Genetische Algorithmus auf das Produktoptimierungsproblem angewendet werden kann. Weiterhin untersuchen sie die Sensitivität der gefundenen Lösungen durch die Variation der einzelnen Parameter. (vgl. Balakrishnan/Jacob 1995, S. 1105) Das Grundprinzip ist das gleiche wie bei Gutsche (1995): Die Chromosomen entsprechen den Produktprofilen und die Gene der Chromosomen entsprechen den Eigenschaftsausprägungen der Produkte. Als Grundlage für die Produktoptimierung werden die Teilnutzenwerte aus einer Conjointanalyse benutzt (vgl. Balakrishnan/Jacob 1995, S. 1106). Dabei maximieren sie zwei Zielfunktionen: den Marktanteil sowie den Kundennutzen und finden heraus, dass es einen signifikanten Unterschied im Vergleich zur Performance der Dynamic-Programming-Heuristik zugunsten des Genetischen Algorithmus gibt. In dieser Veröffentlichung stellen die Autoren ihr *Decision Support System* in den Vordergrund und weniger eine intensive Auseinandersetzung mit den Parametern für Genetische Algorithmen bzw. dem Vergleich mit anderen Algorithmen. Nichtsdestotrotz zeigen sie die Überlegenheit Genetischer Algorithmen gegenüber der

Dynamic-Programming-Heuristik. Allerdings lediglich für ein Beispielproblem, bei dem die Dynamic-Programming-Heuristik ein Produkt findet, dass ca. 90 % des Marktanteils gegenüber der besten Lösung des Genetischen Algorithmus findet. Für ihr System geben die Autoren keine Parameterkonfigurationen vor; der Benutzer kann diese in den vorgegebenen Grenzen selbst bestimmen.

Balakrishnan/Jacob (1996) schließen eine ausführliche Auseinandersetzung mit den Parametern des Genetischen Algorithmus in einer Erweiterung ihrer Arbeit von 1995 an. Die Autoren bemühen zur Kodierung der domänenspezifischen Bezeichnungen die folgenden Metaphern: das Produkt wird als Chromosom bzw. String bezeichnet. Die Eigenschaften entsprechen Genen und die Eigenschaftsausprägungen werden als Allele bezeichnet. Das Produkt ist als binärer String kodiert. Für die Initialisierung der Population werden eine bestimmte Anzahl von Produkten zufällig ausgewählt. Die implementierten Zielfunktionen geben den Marktanteil des neu einzuführenden Produkts mittels First-Choice-Entscheidungsregel und den Kundennutzen an, wobei der Kundennutzen über die Aufsummierung der entsprechenden Teilnutzenwerte geschieht. Nach der Evaluation der Zielfunktionswerte für die einzelnen Produkte in der Population werden die Selektion-, Crossover- und Mutationsoperatoren angewendet. Die Parameter des Genetischen Algorithmus wurden wie folgt gewählt: Eine bestimmte Anzahl zufällig erzeugter Produkte bilden die initiale Population. Die Hälfte der Produkte in jeder Population über die Iterationen hinweg, werden für die Reproduktion selektiert. Dieser Mechanismus stellt sicher, dass die besten Produkte immer die nächste Generation erreichen. Ein Nachteil dieser Strategie kann sein, dass der Algorithmus zu schnell in einem lokalen Optimum endet. Für den Crossover werden zwei Produkte zufällig ausgewählt. Mittels *uniform crossover* mit einer Wahrscheinlichkeit von 0.5, dass eine Eigenschaft ausgetauscht wird, werden zwei neue Produktkandidaten erzeugt. Hierbei wird allerdings sichergestellt, dass genau die Hälfte der Attribute zufällig ausgewählt wird und ein Crossover stattfindet. Die Mutation wird derart vollzogen, dass jedes Produkt in der Population mit einer gewissen Wahrscheinlichkeit ausgewählt wird und eine Eigenschaft gleichverteilt geändert wird. Durch die Reproduktion und die Crossover wächst die Population auf die doppelte Größe an. Sobald dieser Wert erreicht wird, werden die schlechtesten Lösungen aussortiert, so dass die anfängliche Größe der Population wieder erreicht wird und der Prozess von vorn beginnt. Der Algorithmus stoppt, wenn der gleitende Mittelwert der Zielfunktionswerte in der gesamten Population über eine Anzahl von Iterationen stagniert. In ihrer Sensitivitätsanalyse testen die Autoren verschiedene Parameterkonfigurationen, jedoch vergleichen sie keine Reproduktions-, Crossover- oder Mutationsstrategien miteinander. Der vollständige Versuchsplan besteht aus vier Faktoren mit den folgenden Leveln: Populationsgröße $= \{50, 100, 200\}$, Crossover $= \{0, K/4, K/2, 3K/4\}$, wobei K für die Anzahl der Eigenschaften steht, Mutationsrate $= \{0.00, 0.01, 0.10, 0.25, 0.30\}$ und der Änderungsrate des Abbruchkriteriums mit Verbesserungen des Zielfunktionswerts von 2% und 0.2%. Die Sensitivitätsanalyse wird mithilfe einer Varianzanalyse für die Marktanteilsmaximierung sowie die Kundennutzenmaximierung durchgeführt. Für die Marktanteilsmaximierung sind die Hauptef-

fekte Populationsgröße, Crossover und die Änderungsrate der gemittelten Lösungen der Population signifikant. Die Mutationsrate besitzt hingegen keinen statistisch signifikanten Einfluss auf dem 5 %-Niveau. Die gemittelte Anzahl an Iterationen bis zur Konvergenz des Algorithmus liegt bei 7.35. Im Mittel liegen die Lösungen für die Marktanteilsmaximierung bei 96.8 % der optimalen Lösung. Für den Kundennutzen liegt dieser Wert im Mittel bei 97.9 %, wobei im Mittel 8.48 Iterationen bis zur Konvergenz benötigt werden. Die Ergebnisse sind ähnlich derer der Marktanteilsmaximierung. So besitzen alle Haupteffekte, bis auf die Mutationsrate, einen statistisch signifikanten Einfluss auf dem 5 %-Niveau. Für beide Zielfunktionen verhalten sich die statistisch signifikanten Haupteffekte wie erwartet: eine größere Populationsgröße sowie eine kleinere Änderungsrate für das Abbruchkriterium führen zu besseren Ergebnissen. Für die Hauptstudie wurden auf Grundlage der Ergebnisse der Sensitivitätsanalyse die folgenden Parameter gewählt: Populationsgröße: 100; Mutationsrate: 0.3; Änderungsrate der gemittelten Lösungen: 2 %. Obwohl die mittlere Lösungsqualität für die Änderungsrate des Abbruchkriteriums bei 0.2 % und die Populationsgröße mit 200 Produktkandidaten bessere Ergebnisse liefern als die genannten, entscheiden sich die Autoren wahrscheinlich aufgrund des Kompromisses wegen der CPU-Laufzeit für die genannte Variante. Die CPU-Laufzeit verlängert sich mit den besten Parametern insofern entscheidend.

Thakur et al. (2000) implementieren einen Genetischen Algorithmus für ein Marktanteilsmaximierungsmodell. Die initiale Population wird zufällig erzeugt. Nach Bestimmung der Zielfunktionswerte der in der Population befindlichen Produktlinien wird ein bestimmter Anteil der Population mit den höchsten Zielfunktionswerten ausgewählt. Mithilfe dieser Produktlinien werden neue Produktlinienkandidaten mittels des Crossover-Operators erzeugt. Die Produktlinien werden hierfür zufällig ausgewählt und es wird ein *one point*-Crossover ausgeführt, bei dem der Crossover-Punkt jedes Mal zufällig gesetzt wird. Aus diesen Crossover-Operationen werden dann die Nachkommen gebildet. Diese Nachkommen sowie die besten Eltern zzgl. der mutierten Produktlinien, stammend aus dem Pool von Eltern und Nachkommenn, bilden die neue Population. Für die Mutation werden aus der Population zufällig mehrere Produktlinien herausgenommen und in einer Eigenschaft zufällig geändert. Der Genetische Algorithmus wird terminiert, wenn entweder eine maximale Anzahl an Iterationen erreicht ist oder innerhalb von vier Iterationen keine Verbesserung des Zielfunktionswerts erreicht wurde, die über einen gewissen prozentualen Wert hinausgeht. Für die Vergleichsstudie wurden die Parameter wie folgt gewählt: Populationsgröße: 20; Anzahl der Nachkommen: 10; Anzahl mutierender Produktlinien: 2. Für die Abbruchbedingung des Algorithmus wurde ein Wert von 0.5 % vorgegeben.

Alexouda/Paparizzos (2001) nutzen für ihre Implementierung des Genetischen Algorithmus eine Emigrationsstrategie als Populationswahrungsmechanismus. Konkret werden in jeder Iteration die besten 40 % einer Population selektiert und für die Reproduktion verwendet. Weitere 40 % der Population werden über einen Crossoveroperator gleichverteilt aus den besten Eltern erzeugt. Die letzten 20 % der Population werden über den Mutationsoperator erzeugt, indem

aus den Eltern sowie den Nachkommenn zufällig die entsprechende Anzahl ausgewählt wird und zufällig in einer Position (Eigenschaft) verändert werden. Sobald über zehn Iterationen keine Verbesserung des besten Zielfunktionswerts stattfindet, wird der Algorithmus gestoppt. Für den Vergleich werden zwei Versionen des Genetischen Algorithmus implementiert: bei der Ersten wird die Startpopulation zufällig erzeugt. Bei der zweiten Version werden die von Beam Search erzeugten Produktlinien in die Startpopulation integriert. Aufgefüllt wird diese Startpopulation mit zufällig ausgewählten Produktlinien. Für den Genetischen Algorithmus wird keine Simulation durchgeführt. Die Parameter werden im Vorhinein gesetzt und während der Simulationen nicht verändert. Die Populationsgröße beträgt 150. Dieses Vorgehen ist im Allgemeinen kritisch zu betrachten, da, wie Balakrishnan/Jacob (1996) zeigen konnten, die Parameter einen entscheidenden Einfluss auf die Qualität der Lösung haben.

Shi et al. (2001) benutzen einen Genetischen Algorithmus, um die Performance ihres Nested-Partition-Algorithmus zu testen. Die einzige Angabe, die sie zum Genetischen Algorithmus machen, ist, dass sie eine Mutationsrate von 20 % verwenden. Des Weiteren wenden sie den Genetischen Algorithmus nicht auf Produktlinien, sondern auf ein einzelnes einzuführendes Produkt an. Weiterhin kombinieren sie den Nested-Partition-Algorithmus mit einem Genetischen Algorithmus, was zu einer höheren Lösungsqualität führt als der Genetische Algorithmus alleine. Kritisch zu beurteilen ist jedoch, dass keine Angaben zu den sonstigen wichtigen Parametern des Genetische Algorithmus gemacht wurden. Auch die Laufzeiten wurden nicht berichtet. Somit ist es schwierig, ein aussagekräftiges Urteil übder den Performancevergleich zu ziehen. Gegenüber den anderen verwendeten Heuristiken (Dynamic-Programming-Heuristik, Greedy-Heuristik, Divide&Conquer-Heuristik) schnitt der Genetische Algorithmus besser ab.

Tarasewich/McMullen (2001) vergleichen ihre entwickelte Pruning-Heuristik u.a. mit einem Genetischen Algorithmus, für den kaum Angaben zu den verwendeten Parametern gemacht werden. Die Populationsgröße ist 50, 100 und 200. Es wurden vier verschiedene Mutationsraten genutzt, ohne Angabe des genauen Anteils an der Populationsgröße. Aufgrund der ungenauen Angaben und der unüblichen Zielfunktion wird hier nicht auf das weitere Vorgehen eingegangen.

Steiner/Hruschka (2002) führen eine umfangreiche Monte-Carlo-Simulation zur Bestimung der Parameter durch. Hierfür wählen sie fünf Crossover-Wahrscheinlichkeiten $(0.6, 0.7, 0.8, 0.9,$ $1.0)$ und drei Mutationsraten $(0.00, 0.01, 0.05)$. Die Populationsgröße wird in zwanziger Schritten von 30 auf 250 Produktlinien erhöht, wobei festgestellt wird, dass eine größere Population erwartungsgemäß zu besseren Zielfunktionswerten führt (vgl. Steiner/Hruschka 2002, S. 591). Des Weiteren zeigt sich, dass für fast alle Problemgrößen eine Crossover-Wahrscheinlichkeit von 1.0 bessere Zielfunktionswerte liefert, als wenn ein Crossover mit einer geringeren Wahrscheinlichkeit stattfindet. Das Gleiche gilt für geringere, aber vorhandene (ungleich 0) Mutationsraten für eine Eigenschaft eines Produkts in der Produktlinie, wobei sich 1 % als am besten

erwies für steigende Problemgrößen (vgl. Steiner/Hruschka 2002, S. 590). Als Abbruchkriterium wurde ein gleitender Durchschnitt von 0.2 % gewählt.

Steiner/Hruschka (2003) erweitern den Einproduktfall von Balakrishnan/Jacob (1996) auf Produktlinien und ziehen als Vergleich die Greedy-Heuristik sowie den Gewinn als Zielfunktion heran. Eine Produktlinie wird binär kodiert, indem mehrere Produkte als einzelner String dargestellt werden. Die initiale Population wird zufällig erzeugt bevor die Operatoren Reproduktion, Crossover und Mutation auf diese Population angewendet werden. Als Reproduktionsoperator wird die Tournament-Selektion verwendet. Die Tournament-Selektion ist so implementiert, dass aus der Population zufällig zwei Produktlinien mit Wiederholung gezogen werden und sich nur diejenige Produktlinie fortpflanzt, die den höheren Gewinn besitzt. Aus dieser Menge von Produktlinien werden zufällig zwei Produktlinien ohne Wiederholung für den Crossover ausgewählt, die mit einer gewissen Wahrscheinlichkeit miteinander verbunden werden. Um neue Nachkommen zu erzeugen, wird ein 1-Point-Crossover eingesetzt. Der Mutationsoperator wählt zufällig mit einer geringen Wahrscheinlichkeit eine Eigenschaft aus, die mutiert, also geändert wird. Als Abbruchkriterium wird der Ansatz mittels gleitender Mittelwerte gemäß Balakrishnan/Jacob (1996) benutzt. Für die Simulationsstudie wurde eine Sensitivitätsanalyse für die Parameter durchgeführt. Die Autoren geben jedoch weder die Details der Sensitivitätsanalyse, noch die expliziten Parameter für den verwendeten Genetischen Algorithmus an. Der Populationswahrungsmechanismus bleibt ebenso unerwähnt. Für das Beispiel mit realen Conjoint-Daten (Seife) geben die Autoren eine Populationsgröße von 150, eine Crossoverwahrscheinlichkeit von 0.9 und eine Mutationswahrscheinlichkeit von 0.05 an.

Balakrishnan et al. (2004) erweitern ihre ursprüngliche Arbeit über Genetische Algorithmen für die Produktoptimierung (vgl. Balakrishnan/Jacob 1996) vom Einzelprodukt hin zur Produktlinie. Sie adressieren eine Fragestellung, die zu einer ineffektiven Funktionsweise von Genetischen Algorithmen führen kann. Angenommen ein Softdrinkhersteller möchte eine Produktlinie bestehend aus zwei Produkten mit zwei Eigenschaften entwickeln. Die Produktlinie sei wie folgt aufgebaut: Cola mit Zucker und Orangenlimonade mit Stevia. Diese Produktlinie bleibt die gleiche, wenn die Reihenfolge der Produkte getauscht wird: Orangenlimonade mit Stevia und Cola mit Zucker. Für den Genetischen Algorithmus handelt es sich hierbei allerdings um zwei verschiedene Produktlinien. Daraus resultieren die folgenden Probleme: der Suchraum ist größer, da es für jede Produktlinie bestehend aus R Produkten nun $R!$ Möglichkeiten gibt, die gleiche Produktlinie zu bilden. Ein weiteres Problem sind die Nachkommen, die beim Crossover entstehen. Diese können sehr verschiedene Nachkommen hervorbringen. Um diese Probleme zu vermeiden, ordnen die Autoren die Produktlinie lexikografisch. Diese Modellierung hat jedoch keinerlei Auswirkungen auf die Lösungsqualität des Genetischen Algorithmus für das Produktlinienoptimierungsproblem (Balakrishnan et al. 2004, S. 7). Durch die Hybridisierung des Genetischen Algorithmus mit Beam Search in der initialen Population und im Mutationsoperator (für Details siehe Abschnitt 4.1) konnten insgesamt keine Verbesse-

rungen in der Lösungsqualität der Produktlinie festgestellt werden. Im Gegenteil, die Simulationsergebnisse zeigen sogar deutlich die Überlegenheit des Standardalgorithmus mit zufälliger initialer Population und klassischem Mutationsoperator. Der Standardalgorithmus war z. B. in 69.25 % der Testprobleme dem Hybrid mit Beam-Search-Produktlinien in der initialen Population mindestens ebenbürtig und in 57.12 % aller Testprobleme strikt besser. Insgesamt sind die Implementierungen ohne Hybridisierung (sowohl mit lexikografische Sortierung der Eigenschaften und ohne lexikografische Sortierung) den Algorithmen mit Hybridisierung in 52,.37 % der Testprobleme ebenbürtig und in 35.12 % der Testprobleme strikt besser. Da die lexikografische Sortierung keine Auswirkungen auf die Lösungsqualität hat, lässt sich konstatieren, dass der Standardalgorithmus aufgrund der einfacheren Implementierung bei gleich guten Produktlinien als bester Algorithmus in der vorliegenden Arbeit zu wählen ist (siehe Tabelle 4, Abschnitt 4.1).

Fruchter et al. (2006) implementieren einen Genetischen Algorithmus zur Optimierung einer Produktlinie für den Gewinn eines Unternehmens unter Verwendung der Zahlungsbereitschaft von Konsumenten. Die initiale Population wird zufällig erzeugt. Die Kodierung und der Aufbau der Stringrepräsentation unterscheidet sich grundsätzlich von denen, in den anderen Publikationen. Dies liegt an der Andersartigkeit des verwendeten Modells. Ein String ist aufgebaut aus einem Gen, das anzeigt ob eine bestimmte Marke angeboten wird oder nicht. Der Preis eines Produkts wird über insgesamt vier Gene abgebildet. Zwei Gene geben die Granularität der Preisintervalle an und zwei andere geben den Index der Konsumenten an, deren Zahlungsbereitschaft größer ist als die Kosten der Herstellung des Produkts. Des Weiteren kommen dreimal so viele Gene hinzu, wie es mögliche Produktkonfigurationen gibt. Diese Anzahl ist vom zugrunde liegenden Problem abhängig. Die Population wird zufällig initialisiert. Für die Reproduktion werden zufällig zwei oder mehrere Strings ausgewählt und anschließend per Tournament-Selektion diejenige Produktlinie für die Reproduktion bestimmt, deren Zielfunktionswert am größten ist. Ein vorher festgelegter Anteil an Strings wird zufällig erzeugt und in die Population integriert, um eine größere Vielfalt in der Population zu bekommen. Für den Crossover werden zufällig zwei Strings, aus der Tournament-Selektion kommend, miteinander per uniformem Crossover verbunden, was zu zwei neuen Strings führt. Der Mutationsoperator arbeitet mit einer Mutationswahrscheinlichkeit von 1 %. Als Abbruchkriterium für den Genetischen Algorithmus wählen die Autoren entweder die Konvergenz des Algorithmus oder eine vorher definierte Anzahl von Iterationen. Konvergenz wird angenommen, wenn sich die beste Produktlinie in einer bestimmten Anzahl von Iterationen nicht verändert hat.

Belloni et al. (2008b) verwenden in ihrer Arbeit einen Genetischen Algorithmus, der vergleichbare Ergebnisse wie Simulated Annealing bietet. Aufgrund der geringen betrachteten Problemgrößen und dadurch, dass kein statistischer Test zur Überprüfung eines Unterschieds zwischen den Stichproben, bestehend aus den Zielfunktionswerten, durchgeführt wurde, kann kein abschließendes Urteil über die bessere Performance eines der Algorithmen gefällt werden. Für die

Populationsgröße wählen die Autoren 500 und 2000 Produktlinien. Für den realen Conjointdatensatz bringt die größere Population kaum Vorteile, weshalb 500 Produktlinien verwendet wurden. In der Simulationsstudie werden 2000 Produktlinien genutzt, da es für die simulierten Datensätze Unterschiede in der Performance zugunsten der größeren Population gab. Das allgemein Vorgehen wird anhand von 500 Produktlinien verdeutlicht. Zunächst wird eine initiale Population von 500 Produktlinien erzeugt und deren Zielfunktionswerte berechnet. Die 250 Produktlinien mit den höchsten Zielfunktionswerten werden für die Reproduktion verwendet, indem daraus 125 Paare zufällig mit Wiederholung gezogen werden, die ihrerseits 250 neue Produktlinien durch einen 1-Point-Crossover erzeugen. Diese resultierenden 500 Produktlinien werden kopiert und jede Eigenschaft der Produkte wird mit 5 %-iger Wahrscheinlichkeit mutiert. Von diesen nun 1000 Produktlinien werden die 500 Stück mit dem höchsten Zielfunktionswert für die nächste Generation ausgewählt. Terminiert wird der Algorithmus, wenn die finale Population nur noch aus 500 gleichen Produktlinien besteht (vgl. Belloni et al. 2008a, S. 1).

Tsafarakis et al. (2011) vergleichen eine Implementierung eines Genetischen Algorithmus mit der Particle-Swarm-Optimization. Für die initiale Population werden 100 Produktlinien zufällig erzeugt. Von diesen 100 Produktlinien wird eine nicht näher erläuterte Anzahl verwendet, um mit einer Wahrscheinlichkeit von 90 % einen Uniform-Crossover durchzuführen. Danach wird jede Produktlinie mit einer Wahrscheinlichkeit von 4 % derart mutiert, dass zufällig eine Eigenschaft ausgesucht und mit einer neuen Eigenschaftsausprägung versehen wird. Aus diesen neu erzeugten Produktlinien aus, Crossover und Mutation, werden die 100 Produktlinien in die nächste Generation übernommen, die die höchsten Zielfunktionswerte haben. Terminiert wird der Algorithmus, wenn der gleitende Durchschnitt der Zielfunktionswerte der besten drei Produktlinien in der Population sich innerhalb der letzten fünf Iterationen nicht um mehr als 0.2 % ändert. Kritisch anzumerken ist, dass Belloni et al. (2008b) bereits Populationsgrößen von bis zu 2000 Produktlinien betrachtet haben, wohingegen Tsafarakis et al. (2011) lediglich eine Populationsgröße von 100 nutzen. Das könnte die vergleichbaren Performancewerte der Particle-Swarm-Optimization erklären. Zu erwarten ist, dass die Particle-Swarm-Optimization schlechter abschneidet als eine Implementierung des Genetischen Algorithmus, wenn die Populationsgröße deutlich größer gewählt wird. Die vorliegende Arbeit schließt diese Lücke.

Zusammenfassend lassen sich einige Auffälligkeiten konstatieren (siehe Tabelle 7). Die verwendete Kodierung der Produktlinien zeigt keine eindeutige Präferenz. Binäre und nominale Kodierung halten sich ungefähr die Waage. Das dominierende Verfahren für die Selektion ist die Truncation-Selektion. Der Grund hierfür ist sicherlich ihre Einfachheit. Die meist verwendeten Crossover-Strategien sind der Uniform- und der 1-Point-Crossover. Die Mutationsraten schwanken sehr. Einer ähnlich starken Schwankung unterliegen die Populationsgrößen, die von 20 Produktlinien bis 2000 reichen. Alle Populationswahrungsmechanismen werden ähnlich häufig verwendet. Die wichtigste Auffälligkeit ist, dass in der Literatur zum Teil ungenü-

gend darauf eingegangen wird, welche Parameterkonfigurationen aus welchen Gründen genutzt wurden. Ausführliche Untersuchungen dazu führten lediglich Shi et al. (2001), Tarasewich/McMullen (2001) und Steiner/Hruschka (2003) durch. Da es keine allgemeingültige Parameterkonfiguration gibt, die für alle Optimierungsprobleme gut funktioniert, ist das Vorgehen vieler Autoren kritisch zu bewerten. Weiterhin werden, teilweise abgesehen vom Crossover, keine Mechanismen für die Populationswahrung oder verschiedene Selektionsmechanismen miteinander verglichen. Eine systematische Untersuchung im Rahmen der Produktlinienoptimierung steht somit noch aus. Des Weiteren muss festgestellt werden, dass sich die Art und Weise der Implementierung des Mutationsoperators z.T. stark unterscheidet. Während z. B. Steiner/Hruschka (2002, S. 585) und Balakrishnan et al. (2004, S. 6) die Eigenschaften eines Produkts mit einer gewissen Wahrscheinlichkeit mutieren, kopieren Belloni et al. (2008a, S. 1) die Eltern sowie die Nachkommen und mutieren 5 % der resultierenden Produktlinien in einer Eigenschaft eines Produkts in einer Produktlinie. Alexouda/Paparizzos (2001, S. 169) verwenden ebenso eine feste Anzahl von Produktlinien für die Mutation (20 % der Population). Im Endeffekt sind beide Vorgehensweisen unter bestimmten Voraussetzungen äquivalent und dürften keinen oder nur einen geringen Einfluss auf die Qualität der Zielfunktionswerte haben.

4.3.2 Monte-Carlo-Simulation zur Bestimmung der Operatoren des Genetischen Algorithmus

Im Folgenden wird Genetischer Algorithmus mit GA abgekürzt.

In diesem Abschnitt werden die Operatoren des GA identifiziert, die für die in dieser Arbeit verwendeten Zielfunktionen die besten Zielfunktionswerte liefern. Dabei wird der Fokus nicht auf die Parameter des GA (Mutationsrate, Crossover-Wahrscheinlichkeit, etc.) gelegt, sondern auf die verwendeten Operatoren: Populationswahrungsmechanismus, Selektionsverfahren und die Crossover-Verfahren, da diese Operatoren und deren Eignung für die Produktlinienoptimierung noch nicht systematisch untersucht wurden. Deshalb werden für die Parameter für die Monte-Carlo-Simulation die folgenden, bewährten Parameter aus der Literatur verwendet und während der Simulation nicht verändert:

- Populationswahrungsmechanismus und Selektion:
  - Emigration: 50 % der besten Produktlinien einer Generation werden in die nächste Generation übernommen und zur Reproduktion verwendet (Belloni et al. 2008a, S. 1),
  - für beide Malthusian-Strategien: 50 % der besten Produktlinien werden für die Er-

zeugung der Nachkommen verwendet (Belloni et al. 2008a, S. 1). Die Eltern werden nur in die folgende Generation übernommen, wenn sie einen entsprechend höheren Zielfunktionswert nach dem Verkleinern der Population auf die ursprüngliche Populationsgröße besitzen.

- Populationsgröße: Die Populationsgröße wird für diese Simulation auf den Wert 500 gesetzt (Belloni et al. 2008a, S. 1). Umso größer die Population, desto besser ist die Performance des GA. Daher wird die Populationsgröße in späteren Simulationen nach oben angepasst, sobald der GA gegen die anderen Algorithmen in dieser Arbeit antritt. Für die Simulation zur Bestimmung des geeignetsten GA reicht eine Populationsgröße von 500, da diese für alle GA-Kombinationen die gleichen Voraussetzungen bedeutet.

- Crossover-Wahrscheinlichkeit: 100 %, da hiermit bessere Zielfunktionswerte als mit einer geringeren Crossover-Wahrscheinlichkeit ermittelt werden können (Steiner/Hruschka 2002, S. 590),

- Mutationsrate: Jedes Gen (hier: jede Eigenschaft) der aus dem Crossover entstandenen Produktlinien wird in einer Eigenschaft eines Produkts zufällig mit der Wahrscheinlichkeit $P = \frac{1}{g}$ mutiert (Reeves 2003, S. 70), wobei g in der Produktlinienoptimierung die Anzahl der Eigenschaften multipliziert mit der Anzahl der in die Produktlinie aufzunehmenden Produkte darstellt. Die Vorteile sind ein adaptives Verhalten der Mutationsrate bei sich ändernder Problemgröße sowie die Tatsache, dass im Mutationsschritt für alle Nachkommen im Mittel eine Eigenschaft eines Produkts in der Produktlinie verändert wird, wodurch verhindert wird, dass durch die Mutation zu große oder zu kleine Sprünge im Lösungsraum gemacht werden. Des Weiteren werden durch diese Mutationsrate im Mittel verschiedene Nachbarschaftslösungen generiert, was die lokale Suche stärkt. Die Konvergenz des Algorithmus würde bei ungünstig gewählter oder konstanter Mutationsrate über alle Probleminstanzen hinweg verlangsamt bzw. zu stark beschleunigt und die lokale Suche würde zu schwach bzw. zu stark, so dass die Diversität in der Population verloren ginge und der GA zu schnell gegen ein lokales Optimum konvergiert.

- Abbruchbedingung: Abbruch des Algorithmus, wenn der Zielfunktionswert innerhalb von 10 Generationen nicht weiter verbessert werden konnte.

Die Monte-Carlo-Simulation zur Bestimmung der am besten geeigneten Operatoren wird mittels eines vollständigen Versuchsplans durchgeführt. Auf die Verwendung der empirisch vorliegenden Verteilung der Eigenschaften und deren Ausprägungen sowie der Verwendung realistischer Umfrageteilnehmerzahlen, wie in Abschnitt 5.1, wird an dieser Stelle verzichtet, um die Faktoren und Level der Monte-Carlo-Simulation besser kontrollieren zu können und um sie damit vor allem deutlich zu beschleunigen. Somit ergeben sich die in Tabelle 8 verwendeten

Faktoren	**Level**
Eigenschaften	5 10 15
Eigenschaftsausprägungen	5 10 15
Produkte pro Produktlinie	3 9
Konsumenten	200
Wiederholungen eines Problems	10

Tabelle 8: Faktoren und Level für die Erstellung eines vollständigen Versuchsplans für die Bestimmung geeigneter Operatoren eines Genetischen Algorithmus für die Produktlinienoptimierung in Anlehnung an die in der Literatur durchgeführten Simulationen. (Quelle: Eigene Darstellung)

Faktoren und Level für die Monte-Carlo-Simulation, woraus sich 180 ($3 \times 3 \times 2 \times 1 \times 10$) Datensätze ergeben, die von jeder Operatorenkombination für jede der vier Zielfunktionen (siehe Abschnitt 2.2) gelöst werden. Der Versuchsplan orientiert sich an den in der Literatur verwendeten Versuchspläne (siehe Tabellen 4 und 5). Die individuellen Teilnutzenwerte werden, wie in Abschnitt 5.1 beschrieben, aus einer multivariaten Normalverteilung gezogen, wobei die Elemente des Erwartungswertvektors aus einer Gleichverteilung $[-1,1]$ stammen und die Varianz für jedes Element $\sigma^2 = 0.25$ beträgt. Der Status-quo-Markt wird zufällig erzeugt. Die Auswahl dieser Operatoren richtet sich nach den bisher in der Literatur zur Produktlinienoptimierung verwendeten Operatoren (siehe Tabelle 7), um zu untersuchen, ob bestimmte Operatoren einen größeren Einfluss auf die Qualität des Zielfunktionswerts haben als andere. Die Selektionsoperatoren für den Populationswahrungsmechanismus aus Tabelle 9 wurden ausgewählt, da es diejenigen sind, die auch in bisherigen Veröffentlichungen in der Produktlinienoptimierung genutzt wurden. Die Selektionsmechanismen aus Tabelle 9 sind Truncation, Tournament und Stochstic-Universal. Während Truncation und Tournament bereits in der Produktlinienoptimierung verwendet wurden, wird die Selektion um Stochastic-Universal ergänzt, da es sehr gute Eigenschaften im Sinne von Spread, Bias und Laufzeit besitzt (vgl. Baker 1987, S. 18). Die beiden Crossover-Operatoren wurden aus Tabelle 7 übernommen. Die Abbruchbedingung wurde ggü. Balakrishnan/Jacob (1996) und Steiner/Hruschka (2002) angepasst, da die Moving-Average-Regel der Veränderung der Zielfunktionswerte der besten drei Produktlinien in der Population über eine bestimmte Anzahl von Iterationen kleiner als 0.2 % zu schlechteren Zielfunktionswerten als die hier verwendete Abbruchbedingung führte.

Ein weiterer wichtiger Aspekt ist, dass nicht für jede Problemgröße oder jede Zielfunktion ein anderer Genetischer Algorithmus gewählt werden soll. Zur Evaluation gegen andere Algorithmen soll eine konkrete Operatorenkombination und somit ein einzelner GA gefunden werden, der für alle vier Zielfunktionen geeignete (lokale) Optima findet.

Die ausgewählten Operatoren für die Simulationsstudie befinden sich in Tabelle 9. Aus der Kombination dieser Operatoren ergibt sich eine Gesamtzahl von 18 verschiedenen Genetischen Algorithmen ($3 \times 3 \times 2$), die jede Probleminstanz gemäß Tabelle 8 lösen.

Populationswahrungs-mechanismus	**Selektion**	**Crossover**
Emigration	Truncation	1-Point
Malthusian	Tournament	Uniform
modifizierter Malthusian	Stochastic-Universal	

Tabelle 9: Übersicht der ausgewählten Operatoren eines Genetischen Algorithmus für die Produktlinienoptimierung für die Monte-Carlo-Simulation. (Quelle: Eigene Darstellung)

Alg.	**Selektion**	**Crossover**	**Populationswahrung**	**Alg.**	**Selektion**	**Crossover**	**Populationswahrung**
GA01	Truncation	Uniform	Emigration	**GA10**	Tournament	1-Point	Emigration
GA02	Truncation	Uniform	Malthusian	**GA11**	Tournament	1-Point	Malthusian
GA03	Truncation	Uniform	modif. Malthusian	**GA12**	Tournament	1-Point	modif. Malthusian
GA04	Truncation	1-Point	Emigration	**GA13**	Stochastic-Universal	Uniform	Emigration
GA05	Truncation	1-Point	Malthusian	**GA14**	Stochastic-Universal	Uniform	Malthusian
GA06	Truncation	1-Point	modif. Malthusian	**GA15**	Stochastic-Universal	Uniform	modif. Malthusian
GA07	Tournament	Uniform	Emigration	**GA16**	Stochastic-Universal	1-Point	Emigration
GA08	Tournament	Uniform	Malthusian	**GA17**	Stochastic-Universal	1-Point	Malthusian
GA09	Tournament	Uniform	modif. Malthusian	**GA18**	Stochastic-Universal	1-Point	modif. Malthusian

Tabelle 10: Übersicht über die 18 verwendeten Operatorenkombinationen und Bezeichnung der GA. Legende: modif. ... modifizierter. (Quelle: Eigene Darstellung)

Die Ergebnisse der Monte-Carlo-Simulation sind in Tabelle 11 dargestellt. Dabei werden die Mittelwerte der besten gefundenen Lösung der einzelnen GA für die Zielfunktionen ($\overline{ZF}_{max}$), der Mittelwert über alle Zielfunktionen ($\overline{ZF}_{max}^{mean}$), die Mittelwerte über die gemittelten Zielfunktionswerte für die 10 Wiederholungen aller Probleminstanzen als Anteil an der besten gefundenen Lösung aller GA ($\overline{ZF}_{mean}$) sowie der Mittelwert hierfür über alle Zielfunktionen ($\overline{ZF}_{mean}^{mean}$), die mittleren Standardabweichungen für jede Zielfunktion und die mittlere benötigte Zeit für jede Zielfunktion über alle Probleminstanzen. Die Angabe der Werte in der Tabelle ist als Anteil an der besten gefundenen Lösung aller GA zu verstehen. Die Werte befinden sich somit im Intervall $[0;1]$. Die Berechnung der Werte soll beispielhaft mit zwei fiktiven Algorithmen (Alg1 und Alg2) illustriert werden. Angenommen Alg1 und Alg2 berechnen die folgenden Zielfunktionswerte für eine Probleminstanz der deterministischen Marktanteilsmaximierung M_{det} für 10 Wiederholungen:

- Alg1: $0.05, 0.04, 0.05, 0.06, 0.04, 0.05, 0.04, 0.05, 0.06, 0.03$;
- Alg2: $0.03, 0.04, 0.05, 0.02, 0.05, 0.03, 0.05, 0.04, 0.02, 0.05$.

Zunächst wird der höchste Zielfunktionswert aus beiden Algorithmen für diese Probleminstanz $p, p = 1, ..., P$ bestimmt: $ZF_{max}^{p} = max(ZF^{p}) = 0.06$. Alle Zielfunktionswerte werden durch ZF_{max}^{p} geteilt:

- Alg1: $0.83, 0.67, 0.83, 1.00, 0.67, 0.83, 0.67, 0.83, 1.00, 0.50$;

- Alg2: $0.50, 0.67, 0.83, 0.33, 0.83, 0.50, 0.83, 0.67, 0.33, 0.83$.

Dann ist $ZF_{max}(Alg1) = 1.00$ und $ZF_{max}(Alg2) = 0.83$. Dieses Vorgehen wird für alle Probleminstanzen wiederholt, wobei am Ende der Mittelwert aus ZF_{max} für jede Probleminstanz durch die Anzahl der Probleminstanzen geteilt wird, woraus sich $\overline{ZF}_{max} = \frac{\sum_{p=1}^{P} ZF_{max}^{p}}{P}$ ergibt. Der Wert von $\overline{ZF}_{max}^{mean}$ ergibt sich dann durch Summenbildung über $\overline{ZF}_{max}$ geteilt durch die Anzahl der Zielfunktionen ($= 4$). Somit kann die Fähigkeit eines Algorithmus bestimmt werden, hohe Zielfunktionswerte für eine Probleminstanz zu finden. Für die durchschnittliche Performance eines Algorithmus $\overline{ZF}_{mean}$ werden die Mittelwerte aus den Anteilswerten über die 10 Wiederholungen gebildet, was zu den Werten $ZF_{mean}(Alg1) = 0.783$ für Alg1 und $ZF_{mean}(Alg2) = 0.632$ für Alg2 führt. Für das Zusammenfassen mehrerer Probleminstanzen ergibt sich somit die Bildungsvorschrift $\overline{ZF}_{mean} = \frac{\sum_{p=1}^{P} ZF_{mean}^{p}}{P}$. Summiert man diese Werte wieder über alle Zielfunktionen und teilt sie durch deren Anzahl, erhält man den Wert für $\overline{ZF}_{mean}^{mean}$. Zur Bildung der gemittelten Standardabweichung $\overline{\sigma}_{\mathbf{ZF}}$ für jede Zielfunktion werden die Standardabweichungen jeder Probleminstanz aufsummiert und durch die Anzahl der Probleminstanzen geteilt. Für die Berechnungen der gemittelten Zeit ØT, die von den Algorithmen in Anspruch genommen wurde, werden alle Zeiten aufsummiert und durch die Anzahl der Probleminstanzen multipliziert mit der Anzahl der Wiederholungen für jede Probleminstanz geteilt.

Für die Auswahl eines geeigneten GA für die weiteren Simulationsrechnungen mit anderen Heuristiken werden zunächst die diejenigen GA identifiziert, die für eine genauere Betrachtung infrage kommen. Dies wird zum einen über die fett gedruckten Werte für $\overline{ZF}_{max}^{mean}$ und $\overline{ZF}_{mean}^{mean}$ aus Tabelle 11 und zum anderen mit den p-Werten aus Tabelle 12 realisiert. Die p-Werte stammen aus einem T-Test für gepaarte Stichproben, wobei die 18 GA paarweise miteinander durch die Nullhypothese: das arithmetische Mittel des einen Algorithmus ist gleich dem arithmetischen Mittel des anderen Algorithmus, verglichen werden. Das verwendete Signifikanzniveau liegt bei $\alpha = 0.05$. Ist der p-Wert kleiner als das gewählte Signifikanzniveau wird die Nullhypothese verworfen und die Alternativhypothese: es existiert ein Unterschied in den arithmetischen Mittelwerten zweier Algorithmen, wird angenommen. Dazu werden die beiden Mittelwerte verglichen und sich für den größeren entschieden. Der Stichprobenumfang umfasst alle Zielfunktionswerte für alle Zielfunktionen aus allen Probleminstanzen inklusive der 10 Wiederholungen pro Probleminstanz. Der zu untersuchende Mittelwert entspricht somit $\overline{ZF}_{mean}^{mean}$. Insgesamt ergibt sich eine Stichprobengröße von 720 $(3 \times 3 \times 2 \times 4 \times 10)$, weshalb ein T-Test hier trotz nicht gesicherter Normalverteilung der Grundgesamtheit infrage kommt, da der Stichprobenumfang größer als 30 ist und somit durch den zentralen Grenzwertsatz die Normalverteilungsannahme zumindest approximativ gilt. Als nicht-parametrischer Alternativtest käme der Wilcoxon-Vorzeichen-Rang-Test infrage, welchem der T-Test allerdings vorgezogen wird, da nicht-parametrische Tests häufig über eine geringere Teststärke verfügen. Zunächst werden die gemittelten Zielfunktionswerte der besten gefundenen Lösungen ($\overline{ZF}_{max}^{mean}$) und die gemittelten Werte der gemittelten Zielfunktionswerte ($\overline{ZF}_{mean}^{mean}$) aus Tabelle 11 vergli-

Alg.	ZF	$\overline{\mathbf{ZF}}_{max}$	$\overline{\mathbf{ZF}}_{max}^{mean}$	$\overline{\mathbf{ZF}}_{mean}$	$\overline{\mathbf{ZF}}_{mean}^{mean}$	$\overline{\sigma}_{\mathbf{ZF}}$	ØT	Alg.	ZF	$\overline{\mathbf{ZF}}_{max}$	$\overline{\mathbf{ZF}}_{max}^{mean}$	$\overline{\mathbf{ZF}}_{mean}$	$\overline{\mathbf{ZF}}_{mean}^{mean}$	$\overline{\sigma}_{\mathbf{ZF}}$	ØT
GA01	G_{det}	0.891	0.890	0.623	0.793	0.166	5.59	**GA10**	G_{det}	0.815	0.907	0.699	0.844	0.078	10.69
	M_{det}	0.670		0.551		0.063	2.70		M_{det}	0.813		0.682		0.085	5.30
	G_{prob}	0.999		0.999		0.000	8.37		G_{prob}	0.999		0.995		0.004	20.44
	M_{prob}	0.999		0.999		0.000	6.75		M_{prob}	0.999		0.999		0.000	16.73
GA02	G_{det}	0.921	0.894	0.638	0.798	0.173	8.71	**GA11**	G_{det}	0.915	**0.960**	0.830	**0.907**	0.060	22.66
	M_{det}	0.654		0.554		0.072	3.91		M_{det}	0.925		0.799		0.082	8.65
	G_{prob}	0.999		0.999		0.000	11.98		G_{prob}	0.999		0.999		0.001	25.79
	M_{prob}	0.999		0.999		0.000	9.79		M_{prob}	0.999		0.999		0.000	20.99
GA03	G_{det}	0.898	0.940	0.801	0.879	0.064	29.66	**GA12**	G_{det}	0.894	0.941	0.793	0.884	0.066	27.72
	M_{det}	0.862		0.715		0.101	9.50		M_{det}	0.870		0.744		0.082	8.75
	G_{prob}	0.999		0.999		0.000	14.86		G_{prob}	0.999		0.999		0.000	28.61
	M_{prob}	0.999		0.999		0.000	12.66		M_{prob}	0.999		0.999		0.000	23.68
GA04	G_{det}	0.943	**0.969**	0.849	**0.918**	0.059	14.99	**GA13**	G_{det}	0.592	0.741	0.455	0.664	0.082	3.46
	M_{det}	0.932		0.821		0.084	5.81		M_{det}	0.635		0.511		0.079	2.82
	G_{prob}	0.999		0.999		0.000	15.78		G_{prob}	0.795		0.756		0.024	2.25
	M_{prob}	0.999		0.999		0.000	13.08		M_{prob}	0.940		0.933		0.004	1.38
GA05	G_{det}	0.928	**0.968**	0.845	**0.920**	0.053	22.94	**GA14**	G_{det}	0.695	0.816	0.499	0.743	0.120	5.59
	M_{det}	0.942		0.835		0.070	8.37		M_{det}	0.568		0.473		0.068	3.11
	G_{prob}	0.999		0.999		0.000	24.64		G_{prob}	0.999		0.997		0.001	14.22
	M_{prob}	0.999		0.999		0.000	18.99		M_{prob}	0.999		0.999		0.000	10.72
GA06	G_{det}	0.899	0.949	0.811	0.895	0.057	27.53	**GA15**	G_{det}	0.892	0.935	0.749	0.858	0.108	23.37
	M_{det}	0.898		0.771		0.081	9.01		M_{det}	0.847		0.683		0.107	9.51
	G_{prob}	0.999		0.999		0.000	27.27		G_{prob}	0.999		0.999		0.000	17.56
	M_{prob}	0.999		0.999		0.000	23.13		M_{prob}	0.999		0.999		0.000	14.46
GA07	G_{det}	0.841	0.895	0.640	0.791	0.171	7.18	**GA16**	G_{det}	0.632	0.770	0.519	0.694	0.065	4.25
	M_{det}	0.738		0.525		0.127	2.92		M_{det}	0.716		0.580		0.084	3.41
	G_{prob}	0.999		0.997		0.001	12.28		G_{prob}	0.781		0.741		0.024	2.11
	M_{prob}	0.999		0.999		0.000	9.71		M_{prob}	0.947		0.935		0.006	1.57
GA08	G_{det}	0.783	0.860	0.568	0.773	0.123	6.41	**GA17**	G_{det}	0.889	0.936	0.759	0.869	0.087	13.95
	M_{det}	0.657		0.525		0.094	3.95		M_{det}	0.853		0.723		0.094	7.15
	G_{prob}	0.999		0.998		0.000	13.79		G_{prob}	0.999		0.995		0.006	27.53
	M_{prob}	0.999		0.999		0.000	11.46		M_{prob}	0.999		0.998		0.001	21.28
GA09	G_{det}	0.907	0.935	0.802	0.873	0.070	30.85	**GA18**	G_{det}	0.881	0.934	0.775	0.879	0.069	20.12
	M_{det}	0.834		0.690		0.088	9.08		M_{det}	0.856		0.739		0.078	9.29
	G_{prob}	0.999		0.999		0.000	18.02		G_{prob}	0.999		0.999		0.000	33.75
	M_{prob}	0.999		0.999		0.000	15.41		M_{prob}	0.999		0.999		0.000	27.12

Tabelle 11: Ergebnisse der Monte-Carlo-Simulation für die 18 GA als arithmetische Mittelwerte des (prozentualen) Anteils der besten gefundenen Lösung eines der GA für jede der Probleminstanzen. Legende: **ZF**... Zielfunktion; $\overline{\mathbf{ZF}}_{max}$... arithmetische Mittelwerte der Anteile an der besten gefundenen Lösung unter 10 Wiederholungen im Verhältnis zur besten gefundenen Lösung einer Probleminstanz aus allen GA für jede Zielfunktion; $\overline{\mathbf{ZF}}_{max}^{mean}$ ... arithmetisches Mittel aus $\overline{\mathbf{ZF}}_{max}$ über alle Zielfunktionen; $\overline{\mathbf{ZF}}_{mean}$... arithmetische Mittelwerte über den Mittelwert der 10 Wiederholungen einer Probleminstanz als Anteil an der besten gefunden Lösung über alle GA für jede Zielfunktion; $\overline{\mathbf{ZF}}_{mean}^{mean}$... arithmetisches Mittel von $\overline{\mathbf{ZF}}_{mean}$ aller Zielfunktionen; $\overline{\sigma}_{\mathbf{ZF}}$... arithmetische Mittelwerte der Standardabweichungen für jede Zielfunktion (berechnet aus den als Anteile an der besten Lösung transformierten Zielfunktionswerten); ØT... arithmetisches Mittel der benötigten Zeit über alle Probleminstanzen für jede Zielfunktion. Rundungsfehler können auftreten, da alle Werte aus den Originaldaten berechnet wurden. (Quelle: Eigene Darstellung)

chen. Hierbei stechen drei GA mit hohen Werten > 0.90 in beiden Kategorien heraus: GA04, GA05 und GA11. Des Weiteren fällt auf, dass sich diese drei GA signifikant auf dem 5 %-Niveau von allen anderen GA unterscheiden, außer untereinander, wo sie sich nur zum Teil signifikant unterscheiden (siehe Tabelle 12). Konstatieren lässt sich vorerst, dass sich diese drei GA den anderen überlegen zeigen, weshalb die Beziehungen dieser GA im folgenden näher untersucht werden. Der GA04 unterscheidet sich in seinem Mittelwert signifikant von GA11 (p-Wert $= 0.03$). GA04 erreicht im Mittel den Wert $\overline{ZF}^{mean}_{mean}(GA04) = 0.918$ und GA11 den Wert $\overline{ZF}^{mean}_{mean}(GA11) = 0.907$, ergo ist der GA04 dem GA11 im Mittel überlegen. Nimmt man noch den Mittelwert der besten gefundenen Lösungen als Vergleich hinzu, bestätigt sich der Eindruck weiter ($\overline{ZF}^{mean}_{max}(GA04) = 0.969 > 0.960 = \overline{ZF}^{mean}_{max}(GA11)$, p-Wert $= 0.02$ für $n = 72$ Datenpunkte, da von jeder Probleminstanz lediglich der beste erreichte Zielfunktionswert genommen wurde). GA11 unterscheidet sich nicht signifikant auf dem 5 %-Niveau von GA05 (p-Wert $= 0.06$), womit die Nullhypothese der Gleichheit der Mittelwerte nicht verworfen werden kann. Auch der p-Wert $= 0.43$ für die besten gefundenen Zielfunktionswerte je Probleminstanz und die Standardabweichungen (p-Wert $= 0.07$) bieten keine Entscheidungsunterstützung. Die Entscheidung bringt letztlich ein Blick auf die gemittelte Dauer über alle Zielfunktionen von GA05 ($\varnothing\overline{T} = 18.74$) und GA11 ($\varnothing\overline{T} = 19.52$) mit p-Wert $= 0.02$ zugunsten des GA05. Für die Entscheidung, ob GA04 oder GA05 die geeignetste Operatorenkombination für den Vergleich mit den anderen Algorithmen darstellt, werden zunächst wieder die gemittelten Werte der gemittelten Zielfunktionswerte ($\overline{ZF}^{mean}_{mean}$) herangezogen. Statistisch kann die Nullhypothese der Gleichheit der $\overline{ZF}^{mean}_{mean}(GA04) = 0.918$ „=“ $0.920 = \overline{ZF}^{mean}_{mean}(GA05)$, p-Wert $= 0.57$ sowie $\overline{ZF}^{mean}_{max}(GA04) = 0.969$ „=“ $0.968 = \overline{ZF}^{mean}_{max}(GA05)$, p-Wert $= 0.13$ nicht verworfen werden. GA05 weist eine geringere Standardabweichung (p-Wert $= 0.04$) über alle Zielfunktionen als GA04 aus. GA04 ($\varnothing\overline{T} = 12.42$) besitzt ggü. GA05 ($\varnothing\overline{T} = 18.74$) jedoch einen deutlichen Vorteil in der gemittelten Laufzeit (p-Wert $= 2.9 \cdot 10^{-7}$). Da für die spätere Simulationsstudie mit den anderen Algorithmen ein Zeitvorteil von ca. 33 % entscheidend sein kann – z. B. kann die Populationsgröße höher gewählt werden bei keinem Zeitverlust ggü. GA05, wodurch wiederum bessere Zielfunktionswerte zu erwarten sind – wird sich für den weiteren Verlauf für den **GA04 (Truncation, 1-Point, Emigration)** entschieden, womit die *Forschungsfrage 2* beantwortet werden konnte.

In Abbildung 7 ist beispielhaft das Konvergenzverhalten für die größte untersuchte Probleminstanz für die besten drei GA illustriert. Es zeigt sich, dass sich das Konvergenzverhalten der GA deutlich je nach Zielfunktionen, aber kaum unter den betrachteten GA unterscheidet. Während die Konvergenz für die deterministische Gewinnmaximierung (G_{det}) eher allmählich verläuft, gibt es bei der deterministischen Marktanteilsmaximierung (M_{det}) einen sprunghaften Anstieg der Zielfunktionswerte bis zur Konvergenz (10 Iterationen ohne Verbesserung der besten bisherigen Lösung). Für die Zielfunktionen mit probabilistischer Entscheidungsregel verläuft die Konvergenzkurve deutlich flacher, vor allem für die probabilistische Marktanteilsmaximierung (M_{prob}). Dies deutet auf einen fundamentalen Unterschied in der Beschaffenheit der Lösungs-

	GA01	GA02	GA03	GA04	GA05	GA06	GA07	GA08	GA09	GA10	GA11	GA12	GA13	GA14	GA15	GA16	GA17
GA02	0.49																
GA03	0.00	0.00															
GA04	0.00	0.00	0.00														
GA05	0.00	0.00	0.00	**0.57**													
GA06	0.00	0.00	0.59	0.00	0.00												
GA07	0.74	0.32	0.00	0.00	0.00	0.00											
GA08	0.02	0.00	0.00	0.00	0.00	0.00	0.09										
GA09	0.00	0.00	0.75	0.00	0.00	0.91	0.00	0.00									
GA10	0.32	0.65	0.00	0.00	0.00	0.00	0.07	0.00	0.00								
GA11	0.00	0.00	0.01	**0.03**	**0.06**	0.01	0.00	0.00	0.03	0.00							
GA12	0.00	0.00	0.29	0.00	0.00	0.13	0.00	0.00	0.23	0.00	0.00						
GA13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00					
GA14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00				
GA15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00			
GA16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00		
GA17	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.56	0.00	
GA18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.22	0.00	0.03

Tabelle 12: Übersicht der paarweisen Vergleiche mittels p-Wert eines T-Tests für verbundene Stichproben für alle GA. Nullhypothese: das arithmetische Mittel aller Zielfunktionswerte des einen Algorithmus ist gleich dem arithmetischen Mittel aller Zielfunktionswerte des anderen Algorithmus. Stichprobengröße: 720 Zielfunktionswerte. Gewähltes Signifikanzniveau $\alpha = 0.05$. Die fett dargestellten Werte betreffen diejenigen drei Genetischen Algorithmen mit den durchschnittlich höchsten Zielfunktionswerten. Die Werte der Tabelle sind gerundet und nie genau 0. (Quelle: Eigene Darstellung)

räume hin. In den Ergebnissen aus Tabelle 11 fiel für die probabilistischen Zielfunktionen auf, dass sie für fast jeden GA sowohl was die besten Zielfunktionswerte als auch die mittleren Zielfunktionswerte anbelangt, auffällig häufig Zielfunktionswerte sehr nah an der besten gefunden Lösung liefern. Die Vermutung liegt nah, dass sich die Zielfunktionswerte hier in einem schmaleren Band als bei den deterministischen Zielfunktionen befinden. Der Lösungsraum ist flacher, was der bereits hohe Anfangswert der besten Lösung in der Startpopulation anzeigt.

Ein interessanter Effekt tritt zutage, wenn die Mittelwerte der Zielfunktionswerte über die einzelnen Operatorenlevel paarweise miteinander über einen T-Test verglichen werden (siehe Tabelle 13). Zu erwarten wäre, dass aufgrund des ausgewählten GA04 auch bei den Paarvergleichen Truncation, 1-Point und Emigration als bevorzugte Operatorenlevelkombinationen herauskommen. Für Truncation und den 1-Point-Crossover lässt sich das verifizieren. Sie sind signifikant von den anderen Operatorenleveln verschieden auf dem 5 % Signifikanzniveau und besitzen somit die höchsten Zielfunktionsmittelwerte. Für den Populationswahrungsmechanismus zeigt sich jedoch, dass der modifizierte Malthusian auf dem 5 %-Signifikanzniveau signifikant besser abschneidet als Emigration und Malthusian. Diese Tatsache ist überraschend, da unter den besten drei GA (GA04, GA05, GA11) keiner den modifizierten Malthusian als Populationswahrungsmechanismus besitzt. GA04 ist der ausgewählte GA, obwohl er als Populationswahrungsmechanismus Emigration besitzt, welche sowohl signifikant schlechter als Malthusian als auch der modifizierte Malthusian ist. GA05 und GA11 besitzen als Populationswahrungsmechanismus Malthusian, wobei zumindest für den GA05 gilt, dass er abgesehen von der deutlich längeren Laufzeit, sich nicht signifikant von GA04 bezogen auf die mittleren Zielfunktionswerte unterschied. Das bedeutet, dass es Interaktionseffekte zwischen den Operatorenleveln zu geben scheint, die durch die bloße Betrachtung der einzelnen Effekte nicht aufgedeckt werden können. Verließe man sich auf die einzelnen Effekte, würde der ausgewählte GA eine Kombi-

nation aus Truncation, 1-Point und modifiziertem Malthusian sein, was in der Simulation dem GA06 entspräche, womit sich immer noch eine gute, aber nicht die beste Performance erzielen ließe (siehe Tabelle 11).

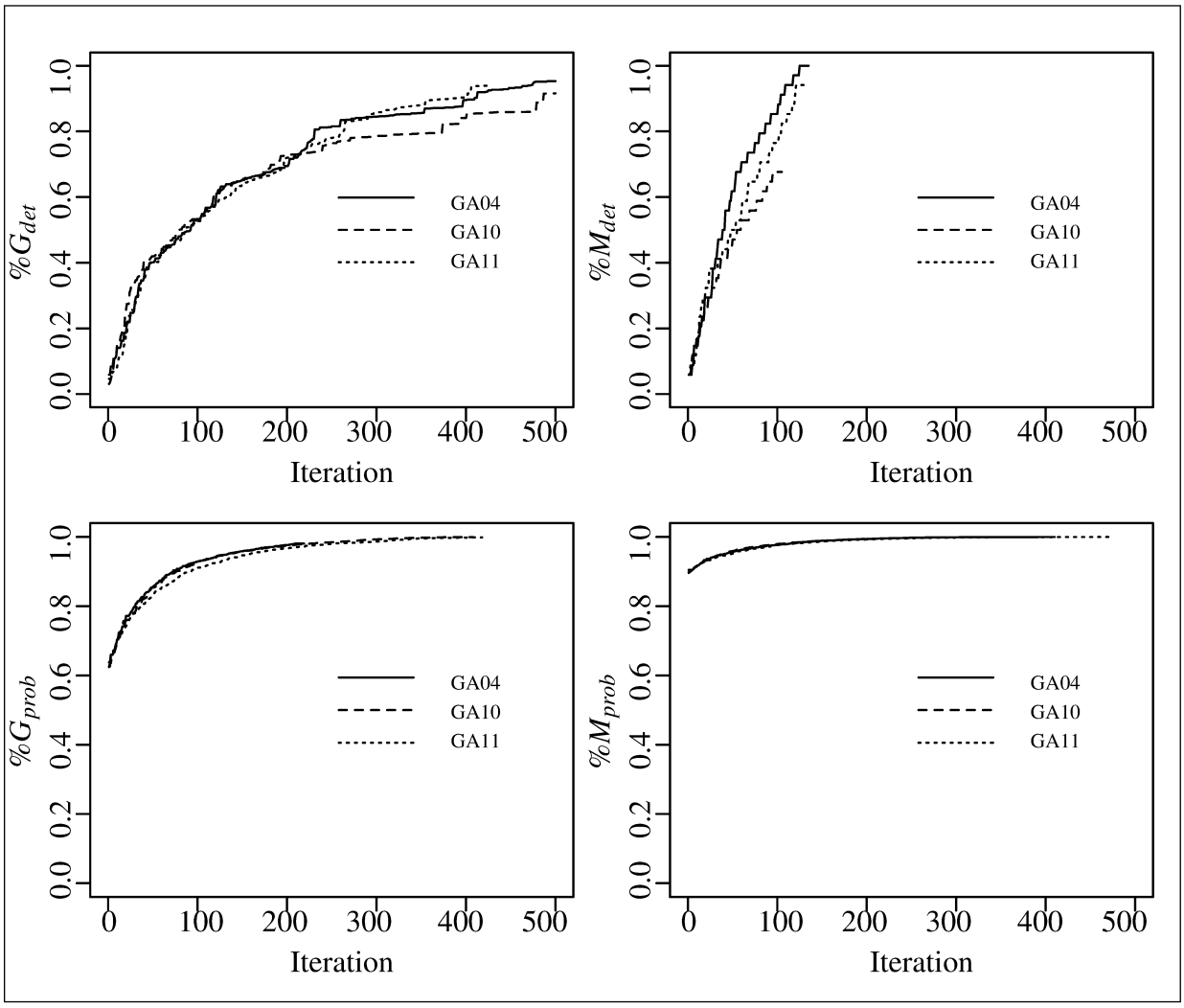

Abbildung 7: Beispielhaftes Konvergenzverhalten der drei besten Genetischen Algorithmen für 15 Eigenschaften mit jeweils 15 Eigenschaftsausprägungen und neun Produkten pro Produktlinie für die vier Zielfunktionen. Die Abzisse ist die Anzahl der Iterationen. Die Ordinate gibt den Verlauf des besten Zielfunktionswerts für jede Iteration als Anteil des maximalen Zielfunktionswert an, der von einem der untersuchten GA für diese Probleminstanz gefunden wurde. (Quelle: Eigene Darstellung)

Generell ist auffällig, dass fast alle GA für die Zielfunktionen mit probabilistischer Entscheidungsregel sehr gute Mittelwerte für die besten und die gemittelten Zielfunktionswerte liefern (außer GA13 und GA16). Die möglichen Gründe hierfür werden in Kapitel 5 diskutiert. Weiterhin fällt auf, dass die verschiedenen GA in den meisten fällen signifikant voneinander unterscheidbar in ihrer Performance bzgl. der gemittelten Zielfunktionswerte sind, wie in Tabelle 12 ersichtlich ist. Dies ist ein weiteres Indiz dafür, dass nicht nur die frei wählbaren Parameter einen Einfluss auf die Lösungsqualität haben, sondern vor allem auch die Kombination aus verschiedenen Operatoren.

Operatoren	**Level**	$\overline{\mathbf{ZF}}^{mean}_{mean}$	**p-Wert**	**Stichprobengröße**
Selektion	Truncation	**0.867**	$2.9 \cdot 10^{-11}$	4320
	Tournament	0.846		
	Truncation	**0.867**	$1.8 \cdot 10^{-57}$	
	Stochastic-Universal	0.785		
	Tournament	**0.846**	$3.3 \cdot 10^{-48}$	
	Stochastic-Universal	0.785		
Crossover	Uniform	0.797	$3.4 \cdot 10^{-39}$	6480
	1-Point	**0.868**		
Populationsw.	Emigration	0.784	$9.8 \cdot 10^{-23}$	4320
	Malthusian	**0.835**		
	Emigration	0.784	$1.9 \cdot 10^{-65}$	
	modifizierter Malthusian	**0.878**		
	Malthusian	0.835	$6.9 \cdot 10^{-22}$	
	modifizierter Malthusian	**0.878**		

Tabelle 13: Paarweise Mittelwertvergleiche mittels der p-Werte eines T-Tests über die Operatorenlevelkombinationen innerhalb der Operatoren. Die Stichprobengrößen ergeben sich aus: 18 GA $\times$ 10 Wiederholung pro Probleminstanz $\times$ 3 Eigenschaften $\times$ 3 Eigenschaftsausprägungen $\times$ 2 Produktliniengrößen $\times$ 4 Zielfunktionen = 1296 Zielfunktionswerte. Die 1296 Zielfunktionswerte werden durch die Anzahl der jeweiligen Operatorenlevel geteilt. Legende: **ZF**... Zielfunktion; $\overline{\mathbf{ZF}}_{max}$... arithmetische Mittelwerte der Anteile an der besten gefundenen Lösung unter 10 Wiederholungen im Verhältnis zur besten gefundenen Lösung einer Probleminstanz aus allen GA für jede Zielfunktion; $\overline{\mathbf{ZF}}^{mean}_{max}$ ... arithmetisches Mittel aus $\overline{\mathbf{ZF}}_{max}$ über alle Zielfunktionen; $\overline{\mathbf{ZF}}_{mean}$... arithmetische Mittelwerte über den Mittelwert der 10 Wiederholungen einer Probleminstanz als Anteil an der besten gefunden Lösung über alle GA für jede Zielfunktion; $\overline{\mathbf{ZF}}^{mean}_{mean}$... arithmetisches Mittel von $\overline{\mathbf{ZF}}_{mean}$ aller Zielfunktionen; $\overline{\sigma}_{\mathbf{ZF}}$... arithmetische Mittelwerte der Standardabweichungen für jede Zielfunktion (berechnet aus den als Anteile an der besten Lösung transformierten Zielfunktionswerten); Ø**T**... arithmetisches Mittel der benötigten Zeit in Sekunden über alle Probleminstanzen für jede Zielfunktion; Populations. ... Populationswahrungsmechanismus. Rundungsfehler können auftreten, da alle Werte aus den Originaldaten berechnet wurden.(Quelle: Eigene Darstellung)

Im folgenden Abschnitt wird eine Methode vorgestellt, die die Performance des GA04 (bzw. generell des GA im Rahmen der Produktlinienoptimierung) durch Ausnutzung der Präferenzstruktur der Konsumenten weiter verbessert.

4.3.3 Verbesserung der Performance durch Maximierung geclusterter Präferenzen

In der Vergangenheit gab es immer wieder Bestrebungen die Performance von GA zu verbessern, indem die Startpopulation mit lokalen Optima, die von anderen Algorithmen erzeugt wurden, geimpft werden. Dieses Vorgehen ist meist nicht von Erfolg gekrönt, da diese lokalen Optima schnell zu einem Genetischen Drift in diese Regionen des Lösungsraums der lokalen Optima führen. So konnten z. B. Balakrishnan et al. (2004) durch die Hybridisierung von GA und Beam Search keine Performanceverbesserungen ggü. dem bloßen GA erzielen. Auch für diese Arbeit wurde versucht, den GA mit anderen Heuristiken zu kombinieren (1-Opt und Simulated Annealing) und die Population während der Iterationen immer wieder mit den lokalen Optima zu impfen. Dieses Vorgehen führte allerdings lediglich zu langen Laufzeiten des GA durch die lokale Suche sowie zur vorzeitigen Konvergenz in ungünstige Lösungsraumregionen. Die Ergebnisse lagen weit unter den Erwartungen, so dass sie hier nicht weiter diskutiert werden. Diese Erkenntnisse lassen die Schlussfolgerungen zu, dass (a) die Startpopulation des GA eine möglichst hohe Diversität aufweisen sollte, damit der GA nicht gegen das lokale Optimum aus der lokalen Suche konvergiert, und (b), falls die Startpopulation mit aussichtsreichen Lösungen geimpft werden soll, müssen diese im Vorhinein durch die Struktur des zugrunde liegenden Optimierungsproblems identifiziert werden. Die in dieser Arbeit betrachteten Zielfunktionen hängen zum einen von den Konsumentenpräferenzen, vor allem die Marktanteilsmaximierungen, die nur auf den Präferenzen aufbauen, ab und zum anderen spielen sie für die Gewinnmaximierungen die relativen Stückdeckungsbeiträgen eine Rolle.

4.3.3.1 Beschreibung des neuen Clusteransatzes

Aus diesen Überlegungen ergibt sich die folgende Idee: Sowohl für die Marktanteilsmaximierung, als auch für die Gewinnoptimierung stehen die Konsumentenpräferenzen im Mittelpunkt der Betrachtung. Sie determinieren die Höhe des Zielfunktionswerts. Eine Produktlinie, die hohe Zielfunktionswerte aufweist, wird den Konsumentenpräferenzen gerecht, da die Anzahl derer, die ein Produkt aus der Produktlinie wählen bzw. eine hohe Wahrscheinlichkeit der Entscheidung für ein Produkt aus der Produktlinie aufweisen (bei den probabilistischen Modellen), entscheidend für dessen Höhe sind. D.h., man sucht Produktlinien, die Produkte beinhalten, die möglichst vielen Konsumenten gerecht werden, da hierfür hohe Zielfunktionswerte zu erwar-

ten sind. Einen weiteren Aspekt bilden die relativen Stückdeckungsbeiträge. Für diese ist die Situation etwas komplizierter, da es hier einen Trade-off zwischen Produkten mit hohen Deckungsbeiträgen gibt, die jedoch nicht zwangsläufig von den Konsumenten präferiert werden müssen. So besitzt ein Produkt mit mittleren Deckungsbeiträgen ggf. eine höhere Konsumentenabdeckung als ein Produkt mit den höchsten Deckungsbeiträgen, was dazu führen kann, dass das Produkt mit den mittleren Deckungsbeiträgen bessere Zielfunktionswerte liefert. Die geringeren Deckungsbeiträge werden durch eine höhere Anzahl an Käufern kompensiert. Ergo, sucht man vor allem nach einer Möglichkeit, die Startpopulation des GA mit vielversprechenden Produkten, die die Konsumentenpräferenzen berücksichtigen zu impfen. Der ideale Fall für die Befriedigung möglichst vieler Konsumentenpräferenzen liegt dann vor, wenn genauso viele Produkte in die Produktlinie aufgenommen werden, wie es Konsumentensegmente gibt, da es in diesem Fall möglich wäre, die Produkte einzeln für jedes Segment zu optimieren, was die Problemgrößen deutlich verringern würde. Dieser Fall liegt in der Praxis selten vor, da die Konsumentenpräferenzen zumeist nicht klar voneinander in Gruppen abgegrenzt werden können (überlappende Präferenzen) und die Anzahl der Produkte in einer Produktlinie häufig von anderen Erwägungen eines Unternehmens abhängig ist, die unabhängig von der Präferenzstruktur sein können, z. B. das Ziel möglichst hoher Marktanteile, auch wenn die Produktlinie dann aus sehr vielen Produkten bestehen kann, um Wettbewerber aus dem Markt zu verdrängen. Die Segmentierung der Konsumentenpräferenzen bietet allerdings interessante Implikationen in Verbindung mit dem Framework zum exakten Lösen des probabilistischen Marktanteilsmaximierungsproblems aus Abschnitt 4.2.2. Zur Erinnerung: die probabilistische Marktanteilsmaximierung ist äquivalent zur Maximierung des Buyers' Welfare (Maximierung des Nutzens der Konsumenten; Kohli/Sukumar 1990, S. 1467), weshalb für die weiteren Erläuterungen von der Maximierung des Nutzens gesprochen wird. Die Startpopulation lässt sich damit, wie im Folgenden beschrieben, impfen.

Ausgehend von der Matrix β der Teilnutzenwerte für die Eigenschaftsausprägungen für jeden Konsumenten:

$$\beta = \begin{pmatrix} \beta_{111} & \cdots & \beta_{1KL_K} \\ \beta_{211} & \cdots & \beta_{2KL_K} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \beta_{I11} & \cdots & \beta_{IKL_K} \end{pmatrix},$$

werden zunächst die individuellen Konsumentenpräferenzen mittels des k-Means-Algorithmus geclustert. Die Anzahl der Cluster bestimmt sich durch die Anzahl der in die Produktlinie aufzunehmenden Produkte R. Deshalb ist k-Means prädestiniert für die Erzeugung der Cluster, da für diesen Algorithmus die Anzahl der Cluster vorher definiert werden kann. K-Means teilt die

Konsumentenpräferenzen in R (Anzahl der Produkte in der Produktlinie) Partitionen auf, indem er die Summe der quadrierten Abweichungen von den Clusterschwerpunkten minimiert, was einem Clustern mit Varianzminimierung entspricht, da die Varianzen als zu minimierende euklidische Distanzen angesehen werden können. Der Algorithmus versucht die Varianz innerhalb der Cluster zu minimieren und die Varianz zwischen den Clustern zu maximieren. Weil k-Means die Startpunkte, aus denen die anfänglichen Schwerpunkte berechnet werden, zufällig initialisiert, ist eine eindeutige Zuordnung der Konsumenten in bestimmte Cluster im Allgemeinen schwer möglich, da sich aus unterschiedlichen Clusterschwerpunkten andere Cluster ergeben. Dieser Nachteil ist jedoch für die zugrunde liegenden Betrachtungen nicht entscheidend. Die Vorteile von k-Means sind, dass gleich große Cluster bevorzugt werden und die Varianz innerhalb der Cluster minimiert wird.

Nachdem R Cluster durch k-Means bestimmt wurden, werden die aufsummierten Teilnutzenwerte der sich in den Clustern befindlichen Konsumenten berechnet, so dass für jedes Cluster ein Vektor mit aufsummierten Teilnutzenwerten wie folgt erzeugt wird:

$$
\begin{aligned}
\overline{\beta}_{kl}^{\,\text{Cluster}_1} &= \frac{1}{|\text{Cluster}_1|}\left(\sum_{i_1\in\text{Cluster}_1}^{|\text{Cluster}_1|}\beta_{i_1 11},\cdots,\sum_{i_1\in\text{Cluster}_1}^{|\text{Cluster}_1|}\beta_{i_1 KL_K}\right)\\
\overline{\beta}_{kl}^{\,\text{Cluster}_2} &= \frac{1}{|\text{Cluster}_2|}\left(\sum_{i_2\in\text{Cluster}_2}^{|\text{Cluster}_2|}\beta_{i_2 11},\cdots,\sum_{i_2\in\text{Cluster}_2}^{|\text{Cluster}_2|}\beta_{i_2 KL_K}\right)\\
&\vdots\\
\overline{\beta}_{kl}^{\,\text{Cluster}_R} &= \frac{1}{|\text{Cluster}_R|}\left(\sum_{i_R\in\text{Cluster}_R}^{|\text{Cluster}_R|}\beta_{i_R 11},\cdots,\sum_{i_R\in\text{Cluster}_R}^{|\text{Cluster}_R|}\beta_{i_R KL_K}\right)
\end{aligned}
$$

Für jedes der R Cluster werden über die Cluster-Mittelwerte mittels des exakten Algorithmus aus Abschnitt 4.2.2 für die probabilistische Marktanteilsmaximierung die J_{best} Produkte mit den J_{best} höchsten Nutzenwerten berechnet. Dabei ist J_{best} ein zu definierender Parameter, der die Anzahl der Produkte für jedes Cluster darstellt, die in die Startpopulation des GA aufgenommen werden sollen. Es ließen sich anstatt der Mittelwerte auch die absoluten, aufsummierten Teilnutzenwerte verwenden. Die Mittelwertbildung erfolgt lediglich, um im ursprünglichen Format der Teilnutzenwerte zu bleiben. Formal lässt es sich mithilfe der Nebenbedingungen 2.18, 2.19, 2.20 der probabilistischen Marktanteilsmaximierung aus Abschnitt 2.2.3.1 darstellen als:

$$\arg\max_x \left(\sum_{j=1}^{J_{best}} \sum_{k=1}^{K} \sum_{l=1}^{L_K} \overline{\beta}_{kl}^{\text{Cluster}_1} x_{jkl} \right) = \left(\text{Produkt}_1^{\text{Cluster}_1}, \cdots, \text{Produkt}_{J_{best}}^{\text{Cluster}_1} \right)$$
$$\arg\max_x \left(\sum_{j=1}^{J_{best}} \sum_{k=1}^{K} \sum_{l=1}^{L_K} \overline{\beta}_{kl}^{\text{Cluster}_2} x_{jkl} \right) = \left(\text{Produkt}_1^{\text{Cluster}_2}, \cdots, \text{Produkt}_{J_{best}}^{\text{Cluster}_2} \right)$$
$$\vdots$$
$$\arg\max_x \left(\sum_{j=1}^{J_{best}} \sum_{k=1}^{K} \sum_{l=1}^{L_K} \overline{\beta}_{kl}^{\text{Cluster}_R} x_{jkl} \right) = \left(\text{Produkt}_1^{\text{Cluster}_R}, \cdots, \text{Produkt}_{J_{best}}^{\text{Cluster}_R} \right)$$

Die Population eines GA besitzt pro Zeile eine Produktlinie mit R Produkten. Das erste Produkt der ersten Produktlinie entspricht dem Produkt mit dem maximalen Nutzen für die Konsumenten im ersten Cluster. Das zweite Produkt der ersten Produktlinie entspricht dem Produkt mit dem maximalen Nutzen für die Konsumenten im zweiten Cluster usw. Somit besteht die erste Produktlinie aus den Produkten mit den maximalen Nutzenwerten für jedes Cluster. Die zweite Produktlinie besteht aus den Produkten mit den zweithöchsten Nutzenwerten pro Cluster. Dieses Vorgehen wird wiederholt bis J_{best} Produktlinien mit den J_{best} besten Produkten für jedes Cluster aufgefüllt wurden. Um die Diversität in der Startpopulation zu erhalten, werden die restlichen Produktlinien der Startpopulation mit zufällig erzeugten Produktlinien aufgefüllt. Damit lässt sich die Startpopulation des GA mit neuem Clusteransatz (GAC) wie folgt darstellen:

$$\text{Startpopulation(GAC)} = \begin{pmatrix} \text{Produkt}_1^{\text{Cluster}_1} & \text{Produkt}_1^{\text{Cluster}_2} & \cdots & \text{Produkt}_1^{\text{Cluster}_R} \\ \text{Produkt}_2^{\text{Cluster}_1} & \text{Produkt}_2^{\text{Cluster}_2} & \cdots & \text{Produkt}_2^{\text{Cluster}_R} \\ \vdots & \vdots & \ddots & \vdots \\ \text{Produkt}_{J_{best}}^{\text{Cluster}_1} & \text{Produkt}_{J_{best}}^{\text{Cluster}_2} & \cdots & \text{Produkt}_{J_{best}}^{\text{Cluster}_R} \\ \text{Produkt}_{J_{best}+1}^{\text{zufällig}_1} & \text{Produkt}_{J_{best}+1}^{\text{zufällig}_2} & \cdots & \text{Produkt}_{J_{best}+1}^{\text{zufällig}_R} \\ \vdots & \vdots & \ddots & \vdots \\ \text{Produkt}_{\text{Pop.gr.}}^{\text{zufällig}_1} & \text{Produkt}_{\text{Pop.gr.}}^{\text{zufällig}_2} & \cdots & \text{Produkt}_{\text{Pop.gr.}}^{\text{zufällig}_R} \end{pmatrix}$$

Die Abkürzung „Pop.gr.“ steht für die Populationsgröße des GA und der Ausdruck $\text{Produkt}_{J_{best}+1}^{\text{zufällig}_1}$ beschreibt ein zufällig erzeugtes Produkt mittels Gleichverteilung, damit die Startpopulation die nötige Diversität erhält. Um diesen neuen Ansatz zu testen, wird der Clusteransatz mit einem

GA verglichen, der mit einer zufällig erzeugten Startpopulation startet. Die genaue Beschreibung erfolgt im nächsten Abschnitt.

4.3.3.2 Ergebnisse des neuen Clusteransatzes

In diesem Abschnitt wird der neue Clusteransatz durch Simulationen getestet. Dabei werden die gleichen Datensätze wie in Abschnitt 4.3.2 verwendet. Die Grundlage des Vergleichs bildet der ausgewählte GA04 aus Abschnitt 4.3.2 mit den Operatoren Truncation, 1-Point und Emigration. Die Populationsgröße bleibt für das Experiment bei 500 Produktlinien. Der GA04 wird mit 500 zufällig ausgewählten Produktlinien initialisiert. Der GAC wird mit $J_{best} = 50$ Produktlinien geimpft, die aus dem Clustering der Konsumentenpräferenzen und der anschließenden Maximierung des Nutzens der Cluster erzeugt werden sowie mit 450 zufällig erzeugten Produktlinien aufgefüllt. Die Zahl der $J_{best} = 50$ entspringt aus Versuchen, eine geeignete Anzahl an Produktlinien zu erzeugen, ohne zum einen die Diversität der Population negativ zu beeinflussen und zum anderen, die J_{best} Produkte für jedes Cluster in angemessener Zeit zu erzeugen. Ab einer bestimmten Anzahl von zu erzeugenden Produkten steigt die benötigte Laufzeit des exakten Optimierungsalgorithmus für die Nutzenmaximierung der Cluster stark an. Da die Lösungsqualität durch eine höhere Anzahl von so erzeugten Produkten nicht weiter ansteigt und somit eine genügend große Anzahl in angemessener Laufzeit von präferierbaren Produkten für jedes Cluster erzeugt wurde, ist ein Wert von $J_{best} = 50$ geeignet.

Die Ergebnisse des Vergleichs sind aus Tabelle 14 zu entnehmen. Dabei ist zu berücksichtigen, dass für den Vergleich der beiden GA die Zielfunktionswerte wieder anteilig an der besten gefundenen Lösung für eine Probleminstanz relativ zueinander bestimmt werden und somit nicht direkt vergleichbar mit den Ergebnissen aus Tabelle 11 sind. Die Überlegenheit des neuen Clusteransatzes bzgl. der maximalen und gemittelten Zielfunktionswerte ist offensichtlich. Für $\overline{\mathbf{ZF}}_{max}^{mean}$, also die beste Lösung über 10 Wiederholungen einer Probleminstanz gemittelt über die Anzahl der Probleminstanzen, ist der Unterschied signifikant auf dem 5 %-Niveau (p-Wert = 0.0005) zugunsten des GAC. Des Weiteren ist der Unterschied in den Mittelwerten aller Zielfunktionswerte $\overline{\mathbf{ZF}}_{mean}^{mean}$ ebenfalls signifikant zugunsten des GAC (p-Wert = $2.66 \cdot 10^{-11}$). Der signifikante Unterschied in der Performance entsteht vor allem durch die besseren Zielfunktionswerte des GAC für die Zielfunktionen mit deterministischer Entscheidungsregel (p-Wert = $1.32 \cdot 10^{-11}$). Für die Zielfunktionen mit probabilistischer Entscheidungsregel ist der Unterschied zwischen den beiden Algorithmen statistisch nicht signifikant auf dem 5 %-Niveau (p-Wert = 0.91). Wie zu erwarten war, zeigt das Impfen der Startpopulation mit Produktlinien, die den GAC in Regionen des Lösungsraum führt, vor allem dort seine Wirkung, wo die Zielfunktionen ausschließlich abhängig von den Konsumentenpräferenzen sind (M_{det} und M_{prob}). Auch für die Zielfunktionen, die Gewinne maximieren (G_{det} und G_{prob}), zeigt sich deutlich, dass es sowohl für die beste gefundene Lösung jeder Probleminstanz, als auch für die mittleren

Algorithmus	**ZF**	$\overline{\mathbf{ZF}}_{max}$	$\overline{\mathbf{ZF}}_{max}^{mean}$	$\overline{\mathbf{ZF}}_{mean}$	$\overline{\mathbf{ZF}}_{mean}^{mean}$	$\overline{\sigma}_{\mathbf{ZF}}$	**ØIterationen**	**ØT**
GA04	G_{det}	0.937	0.945	0.846	0.899	0.120	134.37	14.98
	M_{det}	0.843		0.752		0.204	67.62	5.81
	G_{prob}	0.999		0.999		0.002	131.26	15.78
	M_{prob}	0.999		0.999		0.000	130.65	13.08
GAC	G_{det}	0.985	**0.995**	0.919	**0.961**	0.058	84.61	13.26
	M_{det}	0.997		0.925		0.060	48.52	9.50
	G_{prob}	0.999		0.999		0.001	131.68	19.19
	M_{prob}	0.999		0.999		0.000	102.81	13.99

Tabelle 14: Ergebnisse der Monte-Carlo-Simulation für einen GA mit zufälliger Startpopulation (GA04) und einen GA mit neuem Clusteransatz (GAC) mit den GA-Operatoren Truncation, 1-Point und Emigration für beide GA. Legende: **ZF**... Zielfunktion; $\overline{\mathbf{ZF}}_{max}$... arithmetische Mittelwerte der Anteile an der besten gefundenen Lösung unter 10 Wiederholungen im Verhältnis zur besten gefundenen Lösung einer Probleminstanz aus allen GA für jede Zielfunktion; $\overline{\mathbf{ZF}}_{max}^{mean}$ ... arithmetisches Mittel aus $\overline{\mathbf{ZF}}_{max}$ über alle Zielfunktionen; $\overline{\mathbf{ZF}}_{mean}$... arithmetische Mittelwerte über den Mittelwert der 10 Wiederholungen einer Probleminstanz als Anteil an der besten gefunden Lösung über alle GA für jede Zielfunktion; $\overline{\mathbf{ZF}}_{mean}^{mean}$... arithmetisches Mittel von $\overline{\mathbf{ZF}}_{mean}$ aller Zielfunktionen; $\overline{\sigma}_{\mathbf{ZF}}$... arithmetische Mittelwerte der Standardabweichungen für jede Zielfunktion (berechnet aus den als Anteile an der besten Lösung transformierten Zielfunktionswerten); Ø**T**... arithmetisches Mittel der benötigten Zeit in Sekunden über alle Probleminstanzen für jede Zielfunktion. Rundungsfehler können auftreten, da alle Werte aus den Originaldaten berechnet wurden. (Quelle: Eigene Darstellung)

Zielfunktionswerte einen zusätzlichen Nutzen bringt, die Startpopulation mit den J_{best} besten Produktlinien für jedes Cluster zu versehen. Des Weiteren gelingt durch den neuen Ansatz eine Reduzierung der Standardabweichungen (p-Wert = 0.0008). Auffällig ist, dass die Unterschiede in der Laufzeit beider Algorithmen (p-Wert = 0.016 zugunsten des GA04) anscheinend nicht mit der benötigten mittleren Anzahl an Iterationen bis zur Konvergenz der Algorithmen korrelieren. Der Grund hierfür liegt allerdings in der Berechnung der J_{best} Produkte für jedes Cluster im Vorhinein der Optimierung. Je größer die Lösungsräume der Probleminstanzen, desto länger benötigen die Vorberechnungen. Das Impfen der Startpopulation mit geeigneten Produktlinien führt zu schnellerer Konvergenz des GAC ggü. dem GA04 (p-Wert = $8.52 \cdot 10^{-7}$).

Generell muss angemerkt werden, dass die Trennschärfe der Cluster meist nicht gut ist. Zu erwarten wäre eine gute Trennschärfe, wenn genauso viele Produkte in die Produktlinie aufgenommen würden, wie es tatsächliche Segmente von Konsumenten bzgl. ihrer Präferenzen gäbe. Somit ist die Varianz zwischen den Clustern, vor allem bei einer hohen Zahl von Produkten in der Produktlinie, eher gering. Des Weiteren ist die erklärte Varianz des gesamten Modells mit k-Means ebenfalls eher gering. Nichtsdestotrotz gelingt es hierdurch, von Beginn an des Suchprozesses Produktlinien in günstigen Lösungsraumregionen zu finden, auch wenn die Startlösungen in der Regel noch weit von den (lokalen) Maxima am Ende des Suchpro-

zesses entfernt sind. Die Bedeutung der Zuordnung der einzelnen Konsumenten in bestimmte Cluster ist deshalb auch nicht zentral, da bei ungünstiger Zuordnung einzelner Konsumenten immer noch mindestens ein präferierbares Produkt in der Produktlinie für diese Konsumenten existiert, nämlich das Produkt aus seinem „eigentlichen“ Cluster. Weiterhin werden ungünstige Zuordnungen von Konsumenten abgefedert, indem J_{best} Produktlinien erzeugt werden, was die Diversität innerhalb der günstigen Lösungsraumregionen erhöht. Die Feinjustierung des Optimierungsprozesses obliegt anschließend dem GA, um möglichst die Produktlinie mit dem maximalen Zielfunktionswert zu finden.

Der Clusteransatz funktioniert unter den in dieser Arbeit betrachteten Algorithmen nur für den GA. Experimente mit Simulated Annealing und dem Min-Max Ant-System schlugen fehl. Für Simulated Annealing wurde eine einzelne Produktlinie ($J_{best} = 1$) als Startlösung mit dem Clusteransatz erzeugt. Dies führte allerdings zur vorzeitigen Konvergenz des Algorithmus und somit zu lokalen Maxima, die durch die Standardimplementierung mit zufällig erzeugter Startlösung überboten werden konnte. Beim Min-Max Ant-System wurde der Startvektor mit den Auswahlwahrscheinlichkeiten der Eigenschaftsausprägungen über $J_{best} = 1$ und $J_{best} = 50$ erzeugt, was aber ebenfalls in ungünstigen lokalen Maxima mündete. Insofern ist der GA der einzige Algorithmus dieser Arbeit, mit dem die Performance durch den Clusteransatz gesteigert werden konnte.

Abschließend lässt sich konstatieren, dass der neue Clusteransatz mittels k-Means eine statistisch signifikante Verbesserung der GA-Performance durch Ausnutzung der Struktur in den Präferenzwerten bedeutet. Aus diesem Grund werden sowohl der GA04 als auch der GAC in die Simulationsrechnungen mit den anderen Algorithmen integriert, um zu sehen, ob sich die Vorteile des GAC auch dann zeigen, wenn die Problemgrößen, die Anzahl der Konsumenten sowie die Eigenschaften und Eigenschaftsausprägungen realistischen Conjointproblemen entsprechen. Zum Vergleich wird der „reine“ GA in Form des GA04 weiterhin berücksichtigt.

4.4 Particle-Swarm-Optimization in der Produktlinienoptimierung

Tsafarakis et al. (2011) führten für die Produktlinienoptimierung die Particle-Swarm-Optimization (PSO) ein. Als zu optimierende Zielfunktion verwenden sie die probabilistische Marktanteilsmaximierung basierend auf der BTL-Auswahlregel. Für die Performanceevaluierung wird als Vergleich ein genetischer Algorithmus herangezogen. Im Folgenden werden die verschiedenen Möglichkeiten der Implementierung von Tsafarakis et al. (2011) vorgestellt und diskutiert. Hierzu werden die Ausführungen von Tsafarakis et al. (2011, S. 16-18) beschrieben. Da die

Particle-Swarm-Optimization für stetige Zielfunktionen ersonnen wurde, muss sie für die Produktlinienoptimierung entsprechend in eine diskrete Formulierung überführt werden. Dafür vergleichen die Autoren drei ganzzahlige und zwei binäre Mappings, um herauszufinden, welche Mappings die höchsten Zielfunktionswerte erzeugen.

Mapping	**Kodierung**	**Beschreibung**
Fix	multi-nomial	fallen lassen des Vorzeichens und abschneiden der Nachkommastellen
Trunc	multi-nomial	fallen lassen des Vorzeichens und runden auf die nächste ganze Zahl
Mod	multi-nomial	Modulooperation auf die Partikelposition mittels Anzahl der Eigenschaftsausprägungen für die entsprechende Eigenschaft und abschneiden der Nachkommastellen
Discr	binär	höchster Positionswert innerhalb einer Eigenschaftsausprägung eines Partikels bekommt den Wert 1; alle anderen Eigenschaftsausprägungen der Eigenschaft den Wert 0
SPV	binär	niedrigster Positionswert innerhalb einer Eigenschaftsausprägung eines Partikels bekommt den Wert 1; alle anderen Eigenschaftsausprägungen der Eigenschaft den Wert 0

Tabelle 15: Mappings der reellwertigen Positionswerte s_i eines Partikels in diskrete Entscheidungsvariablen für die Eigenschaften der Produkte in einer Produktlinie gemäß Tsafarakis et al. (2011, S. 17). (Quelle: Eigene Darstellung)

In Tabelle 15 sind fünf verschiedene Mappings sowie deren Beschreibungen dargestellt. Mithilfe dieser Mappings kann der Algorithmus weiterhin, wie in Abschnitt 3.4 beschrieben, in einem mehrdimensionalen reellwertigen Raum operieren, was den Einsatz der wichtigen Gewichtungsfaktoren *intertia weight* und *constriction factor* weiterhin möglich macht. So zeigte sich in den Untersuchungen, dass das Mapping „Discr“ im Mittel die höchsten Zielfunktionswerte lieferte. Unter Berücksichtigung der Gewichtungsfaktoren *intertia weight* und *constriction factor* zeigt sich das Mapping „SPV“ dem Mapping „Discr“ jedoch überlegen (Tsafarakis et al. 2011, S. 22). Die verwendete Parameterkonfiguration für die Simulationen ist Tabelle 16 zu entnehmen.

Tsafarakis et al. (2011) testeten für die Populationsgröße neun verschiedene Werte zwischen 20 und 90 und fanden heraus, dass eine Populationsgröße von 60 Partikeln die höchsten Zielfunktionswerte liefert. Anschließend wurden zehn Kombinationen aus den Nachbarschaften mit unterschiedlicher Partikelanzahl, die in Summe 60 Partikel ergeben, gebildet. Die Kombination aus Nachbarschaftsanzahl und Populationsgröße, die im Mittel die höchsten Zielfunktionswerte lieferte, ist $\{6, 60\}$. Die Parameterwerte w_{min} und w_{max} für das *inertia weight* nahmen die Autoren aus der Literatur. Der Algorithmus wird terminiert, wenn er konvergiert, was bestimmt wird,

Parameter	**Parameterwerte**
Populationsgröße	60
Nachbarschaften	6
Partikelanzahl pro Nachbarschaft	10
w_{min}	0.9
w_{max}	0.1
c_1	2
c_2	2
χ	0.728
Iterationen	1000

Tabelle 16: Stärkste Parameterkonfiguration für die Particle-Swarm-Optimization gemäß Tsafarakis et al. (2011, S. 22). (Quelle: Eigene Darstellung)

wenn nach einer bestimmten Anzahl von Iterationen keine Erhöhung des Zielfunktionswerts erreicht werden kann. Nach spätestens 1000 Iterationen bricht der Algorithmus ab. Kritisch anzumerken ist die Zusammensetzung des *constriction factor* χ in Tabelle 16. Tsafarakis et al. (2011) verwenden für die Parameter c_1 und c_2 jeweils den Wert 2. Die Summe dieser beiden Parameter $c = c_1 + c_2 = 2 + 2 = 4$ widerspricht jedoch der Forderung aus Gleichung 3.8, dass $c > 4$ ist. Für $c = 4$, eingesetzt in Gleichung 3.7, erhält man für den *constriction factor* $\chi = 1$, was wiederum der Particle-Swarm-Optimization ohne *constriction factor* entspräche. Ein Wert von $\chi \approx 0.728$ ergibt sich z.B., wenn $c_1 = 2.05$ und $c_2 = 2.05$ ist. Dann ist $c = c_1 + c_2 = 4.1$ und χ ergibt sich entsprechend (Eberhart/Shi 2000, S. 85). Für die Implementierung des PSO in dieser Arbeit werden die korrekten Werte nach Eberhart/Shi (2000, S. 85) verwendet.

Für PSO wird keine ausführliche Monte-Carlo-Simulation zur Parameterbestimmung durchgeführt. Der Grund hierfür ist, dass sich bereits gezeigt hat, dass der PSO den anderen Algorithmen in dieser Arbeit unterlegen ist (Voekler/Baier 2020). Als Grundlage wurde das Mapping „SPV“ aus Tabelle 16 verwendet, da Tsafarakis et al. (2011) in ihren Simulationen hierfür die im Mittel höchsten Zielfunktionswerte erzielten und dieses Mapping als überlegen in eigenen Simulationen bestätigt werden konnte. Weiterhin konnte die Parameterkonfiguration aus Tabelle 16 als beste Kombination verifiziert werden, mit dem Unterschied, dass $c_1 = 2.05$ und $c_2 = 2.05$ aus oben genannten Gründen gesetzt wurden. Verschiedene Versuche, die PSO für die Produktlinienoptimierung zu verbessern, schlugen fehl. Die Performance blieb für alle vier Zielfunktionen schlecht im Vergleich zu den anderen Heurstiken dieser Arbeit. Die Ergebnisse lassen sich in Kapitel 5 nachvollziehen. Aufgrund des schlechten Abschneidens der PSO und der Tatsache, dass sie in Voekler/Baier (2020) bereits untersucht wurde, wurde in dieser Arbeit keine weiteren Simulationen zur Parameterbestimmung durchgeführt. Die Gründe für die schlechte Performance liegen im Informationsverlust durch die Mappingregeln. Der Informationsverlust kommt zustande, da PSO eigentlich ein Algorithmus für Probleme ist, die stetige Lösungsräume besitzen und sie somit auch auf diesen operiert. Die Umwandlung der Variablen

vom diskreten Lösungsraum in den reellwertigen Lösungsraum und zurück zum diskreten Lösungsraum funktioniert nicht gut. Bei Tsafarakis et al. (2011) erzielt PSO auch nur deswegen vergleichbare Ergebnisse zum GA, weil einerseits die Problemgrößen eher klein sind und andererseits hat sich bereits gezeigt (siehe Abschnitt 4.3), dass die Ergebnisse bei der probabilistischen Marktanteilsmaximierung ohnehin zumeist sehr beieinander liegen, wofür die Struktur des Lösungsraums verantwortlich ist, in welchem die Zielfunktionswerte tendenziell in einem recht schmalen Band zu finden sind. Des Weiteren konnten Voekler/Baier (2020) zeigen, dass die Performance von PSO ab ca. 10^{23} zulässigen Produktlinien deutlich mit der Problemgröße auch für die probabilistische Marktanteilsmaximierung abnimmt.

Nichtsdestotrotz wird PSO der Vollständigkeit halber in die späteren Simulationsrechnungen integriert, da es sich um ein Verfahren handelt, dass noch nicht für alle hier verwendeten Zielfunktionen mit anderen Algorithmen verglichen wurde. Am Ende dieser Arbeit soll Klarheit über die Performance der Algorithmen durch umfassende Simulationen auf dem Gebiet der Produktlinienoptimierung herrschen, weshalb die PSO trotz ihrer Schwächen weiterhin in dieser Arbeit berücksichtigt wird.

4.5 Ant-Systems in der Produktlinienoptimierung

Das allgemeine Prinzip der Ant-Colony-Optimization (ACO) wurde in Abschnitt 3.5 vorgestellt. In diesem Abschnitt wird die ACO von Albritton/McMullen (2007) für den Einproduktfall auf Produktlinien erweitert und dargelegt. Des Weiteren wird das Min-Max Ant-System (MMAS) von Stützle/Hoos (1997) für die Produktlinienoptimierung eingeführt, da es sich als Weiterentwicklung zur ACO beim Problem des Handlungsreisenden durch die Festlegung von minimalen und maximalen Pheromonmengen gegenüber anderen Ameisenalgorithmen als überlegen erwies. Ein weiterer Unterschied des MMAS zur ACO von Albritton/McMullen (2007) ist die Verwendung der Zielfunktionswerte für die Bildung der Pheromonmengen bzw. -pfade. Bei der folgend beschriebenen ACO-Implementierung werden die Zielfunktionswerte lediglich zur Bildung einer Reihenfolge verwendet und die Pheromonmengen der beteiligten Eigenschaftsausprägungen um 1 erhöht, falls ein neuer bester Zielfunktionswert gefunden wurde. Beim MMAS hingegen wird in jeder Iteration von der Ameise mit dem besten Zielfunktionswert ein Update der Pheromonmengen proportional zum Zielfunktionswert für die entsprechenden Eigenschaftsausprägungen durchgeführt. Nachfolgend werden beide Varianten beschrieben und einem Vergleich unterzogen, um die beste Variante gegen die in dieser Arbeit benutzten Algorithmen zu benchmarken. Des Weiteren werden verschiedene Parameterkonfigurationen für das MMAS für die vier Produktlinienoptimierungsmodelle getestet und verglichen.

4.5.1 Modell von Albritton/McMullen (2007)

Albritton/McMullen (2007) modellieren die Präferenzen der Konsumenten über die Zeit. Das bedeutet, dass sie eine Produktlinie suchen, die sich unter den sich verändernden Präferenzen der Konsumenten über eine gewisse Zeitspanne den Marktanteil maximiert. Die sich verändernden Präferenzen werden über einen Diffusionsprozess modelliert, der auf gleichverteilten Teilnutzenwerten basiert. Die Autoren argumentieren, dass eine Variation der Teilnutzenwerte über einen bestimmten Zeitraum von den Entscheidungen und der Erfahrung der Konsumenten aus der Vergangenheit abhängen und somit ein größerer Realitätsbezug besteht (Albritton/McMullen 2007, S. 498). Dem ist entgegenzusetzen, dass Teilnutzenwerte in den seltensten Fällen über alle Eigenschaften hinweg gleichverteilt sind. So wird bspw. ein niedriger Preis meistens einem höheren vorgezogen. Die Parameter für den Diffusionsprozess, der die Variation in den Teilnutzenwerten über die Zeit steuert, sind willkürlich gewählt und deren Wahl wird nicht begründet. Außerdem hieße dieses Vorgehen, dass für verschiedene Zeitpunkte (hier: bis zu 10 Stck.) immer die gleiche Conjointstudie mit den gleichen Teilnehmern und den immer gleichen Eigenschaften und den immer gleichen Eigenschaftsausprägungen durchgeführt werden müsste. Sobald eine neue Eigenschaft oder eine Eigenschaftsausprägung hinzukäme, z. B. die Geschmacksrichtung Mate bei Brauseherstellern, zerfiele das gesamt zeitvariierende Modell der Autoren. Da diese Annahmen unrealistisch sind und diese Arbeit keine über die Zeit variierenden Produktlinienoptimierungsmodelle betrachtet, wird die Zeitkomponente für die folgende Beschreibung des ACO gestrichen und die Gleichungen entsprechend für $t = 1$ angepasst.

Die Bezeichnungen für die Variablen und Parameter werden zum Teil aus Albritton/McMullen (2007, S. 506) übernommen. Die Indizes werden an die in dieser Arbeit verwendeten Indizes angepasst. Variablen und Parameter für die Produktlinienoptimierung bzw. dem Ant-System aus Abschnitt 3.5 werden ebenso an diese Arbeit angepasst. Des Weiteren wird das vorgestellte Modell verkürzt dargestellt, da nach der durchgeführten Monte-Carlo-Simulation für die verwendeten Parameter nicht mehr viel vom ursprünglichen Modell übrig bleibt. Der Grund hierfür liegt darin, dass die Autoren versuchen, die Wahrscheinlichkeitsfunktion für die Auswahlentscheidungen der Ameisen für eine Eigenschaftsausprägung über eine Summe mit gewichteten Summanden für ein lokales Update, ein Update in Abhängigkeit vom Zielfunktionswert und ein globales Update, welches nur durchgeführt wird, falls ein besserer Zielfunktionswert gefunden wird, zu modellieren (Albritton/McMullen 2007, S. 507). In den Simulationen zeigt sich, dass die besten Zielfunktionswerte erzielt werden, wenn nur das globale Update durchgeführt wird. Insofern wird im Folgenden lediglich das stark vereinfachte Modell dargestellt.

Die Teilnutzenwerte für die Konsumenten werden aus einer Gleichverteilung im Intervall $[a,b]$ gezogen. Im nächsten Schritt wird die sog. G-Matrix initialisiert, indem alle Matrixeinträge $g_{kl} = 1, \forall k \in K, \forall l \in L_k$ gesetzt werden. Diese Matrix merkt sich diejenigen Eigenschaftsausprägungen, die zu besseren Zielfunktionswerten geführt haben. Sie besitzt die Zeilenanzahl K,

also so viele Zeilen wie Eigenschaften, und die Spaltenanzahl entspricht der Anzahl der Eigenschaftsausprägungen. Die Bildungsvorschrift lautet:

$$g_{kl} = g_{kl} + 1, \quad \text{wobei } k,l \in Z^{max} \tag{4.5}$$

Sobald ein neuer bester Zielfunktionswert Z^{max} gefunden wurde, werden die Matrixeinträge g_{kl} der daran beteiligten Eigenschaftsausprägungen um den Wert 1 erhöht. Da eine solche Matrix in der Art nur existieren kann, wenn alle Probleminstanzen symmetrisch sind, also jede Eigenschaft gleich viele Eigenschaftsausprägungen besitzt, wird sie in dieser Arbeit als Vektor modelliert. Außerdem wird in der Arbeit von Albritton/McMullen (2007) lediglich der Einproduktfall berücksichtigt. Deshalb ist eine weitere Anpassung des Vektors nötig. Der Vektor G' hat die Länge $R \cdot \sum_{k=1}^{K} L_k$ und besteht aus allen Eigenschaftsausprägungen für alle R in die Produktlinie aufzunehmenden Produkte. Somit ändert sich die Bildungsvorschrift 4.5 für die Pheromonmenge des Vektors G' des globalen Updates mit den Komponenten $g_{jkl}, j = 1,...,R, k = 1,...,K, l = 1,...,L_k$, zu:

$$g_{jkl} = g_{jkl} + 1, \quad \text{wobei } j = 1,..,R; \quad k,l \in Z^{max} \tag{4.6}$$

Daraus leiten sich die folgenden Auswahlwahrscheinlichkeiten p_{jkl} für das j-te Produkt der Produktlinie für die l-te Eigenschaftsausprägung der k-ten Eigenschaften ab:

$$p_{jkl} = \frac{g_{jkl}}{\sum_{l'=1}^{L_k} g_{jkl'}}, \quad j = 1,...,R, k = 1,...,K, l = 1,...,L_k \tag{4.7}$$

Wobei gilt:

$$\sum_{l=1}^{L_k} p_{jkl} = 1 \quad j = 1,...,R, k = 1,...,K \tag{4.8}$$

Beispielhaft wird ein Produkt mit zwei Eigenschaften ($K = 2$), drei und zwei Eigenschaftsausprägungen ($L_1 = 3, L_2 = 2$) und zwei Produkten in der Produktlinie ($R = 2$) gewählt. Der Vektor P für die Auswahlwahrscheinlichkeiten der Eigenschaftsausprägungen mit den Komponenten p_{jkl} für jedes Produkt $j = 1,2$ in der Produktlinie ergibt sich zu:

$$P = \left(\underbrace{p_{111}, p_{112}, p_{113}, p_{121}, p_{122}}_{\text{Produkt 1}}, \underbrace{p_{211}, p_{212}, p_{213}, p_{221}, p_{222}}_{\text{Produkt 2}} \right) \tag{4.9}$$

Die Auswahlwahrscheinlichkeiten der gleichen Eigenschaftsausprägungen für Produkt 1 und Produkt 2 unterscheiden sich und werden somit wie sich voneinander unterscheidende Entscheidungsvariablen modelliert. Würde man für jedes Produkt in der Produktlinie dieselben Auswahlwahrscheinlichkeiten für die Eigenschaftsausprägungen haben, so stiege die Chance, immer die gleichen Produkte zu erzeugen, was zu Dopplungen von Produkten in der Produktlinie führt. Behandelt man die Auswahlwahrscheinlichkeiten der Eigenschaftsausprägungen wie unterschiedliche Entscheidungsvariablen, lässt sich dieses Problem vermeiden: Gibt es zwei oder mehr Produkte in einer Produktlinie die genau gleich sind (gleiche Eigenschaftsausprägung in jeder Eigenschaft), gibt es für die Auswahlwahrscheinlichkeiten der Eigenschaftsausprägungen wegen des schlechten Zielfunktionswerts eine Verringerung. Entweder die Produkte kannibalisieren sich gegenseitig, was zu niedrigeren Zielfunktionswerten führt oder es gibt eine Strafe für Zielfunktionswerte, die durch mehrere gleiche Produkte in der Produktlinie entstanden sind. Der Algorithmus lernt so Regionen im Suchraum zu vermeiden, die aus mehreren gleichen Produkten pro Produktlinie bestehen.

Zur Illustration soll das obige Beispiel kurz weitergeführt werden. Der Vektor G' wird für alle $g_{jkl} = 1$ initialisiert:

$$G' = (1,1,1,1,1,1,1,1,1,1) \tag{4.10}$$

Daraus wird der Vektor P für die Auswahlwahrscheinlichkeiten durch 4.7 berechnet:

$$P = \left(\underbrace{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{2}, \frac{1}{2}}_{\text{Produkt 1}}, \underbrace{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{2}, \frac{1}{2}}_{\text{Produkt 2}} \right) \tag{4.11}$$

Der Zielfunktionswert $Z^{max} = 0$ wird initialisiert. Angenommen es wird algorithmisch eine zulässige Produktlinie mit den Produkten $(0,1,0,1,0)$ und $(0,0,1,1,0)$ mit $Z > Z^{max}$ gefunden. Dann ergibt sich der Vektor G' zu:

$$G' = (1,2,1,2,1,1,1,2,2,1) \tag{4.12}$$

Somit bestimmt sich der Vektor P zu:

$$P = \left(\underbrace{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}, \frac{2}{3}, \frac{1}{3}}_{\text{Produkt 1}}, \underbrace{\frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{2}{3}, \frac{1}{3}}_{\text{Produkt 2}} \right) \tag{4.13}$$

Verbessert sich der Zielfunktionswert in einer Iteration nicht, erfolgt kein Update der Auswahlwahrscheinlichkeiten. Das globale Update für die Auswahlwahrscheinlichkeiten P stärkt also den Pfad in die Region derjenigen Produktlinie, die den bisher besten Zielfunktionswert verbessern kann. In der nächsten Iteration des Algorithmus werden durch die Ameisen neue Produktlinien auf Basis der aktualisierten Auswahlwahrscheinlichkeiten erzeugt. Z.B. steigt für die kommende Iteration die Chance der Auswahl der zweiten Eigenschaftsausprägung der ersten Eigenschaft des ersten Produkts von $p_{112} = \frac{1}{3}$ auf $p_{112} = \frac{1}{2}$ nach der ersten Iteration. Analog gilt das Gleiche für p_{121}, p_{213} und p_{221}. Diejenigen Eigenschaftsausprägungen, die nicht Bestandteil der neuen besten Lösung sind, verringern durch den beschriebenen Mechanismus ihre Auswahlwahrscheinlichkeit. Mit fortschreitenden Iterationen konvergiert der Algorithmus gegen ein (lokales) Optimum, indem die Auswahrscheinlichkeiten sich 1 bzw. 0 nähern.

4.5.2 Min-Max Ant-System (MMAS) für die Produktlinienoptimierung

Das in Abschnitt 3.5 vorgestellte MMAS von Stützle/Hoos (1997) wird für die Produktlinienoptimierung beschrieben. Für einige Parameter des Algorithmus, wie minimale und maximale Pheromonmenge sowie die initialen Pheromonmengen, müssen Anpassungen vorgenommen werden, da es sich hierbei zum einen um Maximierungsmodelle und zum anderen um unterschiedliche Optimierungsprobleme handelt. MMAS wurde zunächst für das Problem des Handlungsreisenden konzipiert, dessen Modellierung jedoch nicht 1:1 übernommen werden kann. Des Weiteren wird ein memetischer Algorithmus beschrieben, der eine Kombination aus MMAS und einer lokalen Suche (MMAS_ls) darstellt.

4.5.2.1 Modellierung des MMAS

Die Initialisierung des MMAS geschieht über eine Gleichverteilung der Pheromonmengen. So ist die Auswahl jeder Eigenschaftsausprägung einer Eigenschaft gleich wahrscheinlich. In der ersten Iteration wird die größte Pheromonmenge, also der höchste Zielfunktionswert verwendet, um die maximale Pheromonmenge τ_{max} zu berechnen, um die Pheromonmenge erneut mit der

maximalen Pheromonmenge τ_{max} zu initialisieren. Ein höherer Startwert führt zu einer langsameren Entwicklung der Wahrscheinlichkeiten gen 0 und 1, wodurch die Exploration des Suchraums gestärkt wird.

Das Pheromonupdate richtet sich zunächst danach, welche Strategie für das Update genutzt wird. Da sich für die Produktlinienoptimierung, wie auch beim Problem des Handlungsreisenden zeigte, werden bessere Zielfunktionswerte erzielt, wenn nur die beste Ameise in jeder Iteration t ein Pheromonupdate durchführen darf. Jeder Eigenschaftsausprägung wird, je nach Beteiligung an der besten Produktlinie in einer Iteration, eine bestimmte Pheromonmenge zugewiesen. Es wird ein Vektor τ mit der Länge der Summe der Anzahl der Eigenschaftsausprägungen multipliziert mit der Anzahl der in die Produktlinie aufzunehmenden Produkte $j = 1,...,R$, erzeugt, auf welchem die entsprechenden Pheromone abgelegt werden. Hieraus ergibt sich folgende Bildungsvorschrift für die Pheromonmenge $\tau_{jkl}, j = 1,...,R; k = 1,...,K; l = 1,...,L_k$ (Stützle/Hoos 1997, S. 310):

$$\tau_{jkl}(t+1) = \rho\,\tau_{jkl}(t) + \Delta\tau_{jkl}^{best}(t) \tag{4.14}$$

mit

$$\Delta\tau_{jkl}^{best}(t) = \begin{cases} Z^{max}(t), & \text{falls } (j,k,l) \in Z^{max}(t) \\ 0, & \text{sonst,} \end{cases} \tag{4.15}$$

wobei $\Delta\tau_{jkl}^{best}(t)$ die Pheromonmenge auf der l-ten Eigenschaftsausprägung der k-ten Eigenschaft des j-ten Produkts der Ameise mit dem besten Zielfunktionswert repräsentiert. $Z^{max}(t)$ stellt den erreichten Zielfunktionswert der besten Ameise dar, der auf die entsprechenden Elemente des Vektors τ addiert wird. Der Evaporisationskoeffizient $\rho \in (0,1]$ legt fest, wie schnell die Pheromonmengen aus den vorhergehenden Iterationen vergessen werden, so dass neue, vielversprechende Regionen des Suchraums gefunden und ältere ungünstigere Regionen aufgegeben werden können. In der Praxis haben sich Werte von nah an 1 bewährt, z. B. $\rho = 0.98$, (Stützle/Hoos 1997, S. 310), weshalb dieser Evaporisationskoeffizient auch für die Simulationen in dieser Arbeit verwendet wird. Für die Produktlinienoptimierung könnte ebenso eine Sensitivitätsanalyse für den Evaporisationskoeffizienten durchgeführt werden. Da das Prinzip und die Wirkungsweise für jedes Optimierungsproblem gleich sind, nämlich zum einen die Konvergenz des Algorithmus sicherzustellen und zum anderen ungünstige Lösungsraumregionen wieder zu verlassen, gibt es keinen Grund anzunehmen, dass der Evaporisationskoeffizient $\rho = 0.98$ für die Produktlinienoptimierung ungeeignet wäre, was sich in den anschließenden Simulationsrechnungen bestätigen wird.

Stützle/Hoos (1997, S. 310) schlagen für die Anzahl der Ameisen einen Wert vor, der der Anzahl der Entscheidungsvariablen entspricht. Dieser Wert würde für die Produktlinienoptimierung für größere Probleminstanzen zu groß werden, so dass in jeder Iteration zu viele Zielfunktionswerte berechnet werden müssten, was die Laufzeit erheblich verlängert. Aus diesem Grund wird lediglich die Summe der Eigenschaftsausprägungen eines Produkts als Anzahl für die Ameisen verwendet und nicht die Summe der Eigenschaftsausprägungen für die gesamte Produktlinie. Die Anzahl der verwendeten Ameisen ist also abhängig von der jeweiligen Probleminstanz:

$$\#\text{Ameisen} = \sum_{k=1}^{K} L_k \ .$$

Für die minimale und maximale Pheromonmenge τ_{min} und τ_{max} wird von den Bildungsvorschriften 3.16 und 3.16 aus Abschnitt 3.5.2 abgewichen, da diese sich nicht als geeignet herausstellten. Eine bessere Wahl für die beiden Parameter ist in Anlehnung an Stützle/Hoos (2000, S. 899):

$$\tau_{max}(t) = \frac{1}{1-\rho} \cdot Z^{max}(t) \ ,$$

wobei ρ der Evaporisationskoeffizient ist und $Z^{max}(t)$ der aktuell beste Zielfunktionswert in Iteration t, der in allen vorherigen und aktuellen Iterationen gefunden wurde. In Abhängigkeit von der maximalen Pheromonmenge ergibt sich der Wert für die minimale Pheromonmenge:

$$\tau_{min}(t) = \frac{\tau_{max}(t)}{2 \cdot \#\text{Ameisen}} \ ,$$

als Quotient aus der maximalen Pheromonmenge und der doppelten Anzahl der Ameisen. Die minimale Pheromonmenge wurde dabei durch Simulationsrechnungen empirisch austariert und die beste gefundene Bildungsvorschrift wird hier berichtet.

Als Abbruchbedingung werden von Stützle/Hoos (1997, S. 311/312) verschiedene Anzahlen von Iterationen verwendet, die vor allem empirisch ausprobiert wurden. In dieser Arbeit werden mehrere Abbruchbedingungen genutzt, wobei diejenige greift, die als erste erfüllt ist. Die drei Abbruchbedingungen sind (1) eine vorher definierte Anzahl an Iterationen (abhängig von der Anzahl der Ameisen: $\frac{250000}{\#\text{Ameisen}}$), (2) eine vorher definierte maximale Laufzeit (max. 300 Sekunden) und (3) eine maximale Anzahl von Iterationen (= 500) ohne Verbesserung des bisherigen Maximums. Der Wert $\frac{250000}{\#\text{Ameisen}}$ wurde für die Simulationsrechnungen zur Auswahl eines geeigneten MMAS verwendet, damit die Anzahl an maximalen Zielfunktionswertevaluationen

für alle verglichenen Ameisenalgorithmen konstant ist, um mehr Kontrolle über diesen Parameter zu erlangen und aussagekräftigere Ergebnisse zu erhalten. Abbruchbedingung (3) trägt der Stagnation bzw. der Konvergenz des Algorithmus Rechnung und wurde empirisch ermittelt.

4.5.2.2 Modellierung des MMAS mit lokaler Suche

Eine Möglichkeit zur Verbesserung der Performance des MMAS liegt in der Kombination des ursprünglichen MMAS mit einem lokalen Suchalgorithmus (MMAS_ls). Stützle/Hoos (1997) zeigen, dass diese Kombination die Performance des MMAS erheblich verbessern kann. Die Autoren nutzen hierfür einen Algorithmus für das Problem des Handlungsreisenden (Lin- Kernighan Algorithmus), der speziell für dieses Problem erdacht wurde, aber nicht für die Produktlinienoptimierung infrage kommt, da dieser Algorithmus problemspezifische Strukturen ausnutzt. Die lokale Suche findet dabei in jeder Iteration des MMAS statt. Hierfür wird die beste Produktlinie, die vom MMAS gefunden wurde für die lokale Suche verwendet. Anschließend findet das Pheromonupdate auf Basis dieser Produktlinie statt, da diese den besten Zielfunktionswert besitzt, weil die lokale Suche nicht zu geringeren Zielfunktionswerten führen kann. Eine Ausnahme würde hier z. B. Simulated Annealing bilden, wo durchaus schlechtere Produktlinien akzeptiert werden könnten. Für die Produktlinienoptimierung gibt es mehrere lokale Suchalgorithmen, die geeignet sein könnten. Getestet wurden für diese Arbeit z. B. eine Kombination aus MMAS und Simulated Annealing. Die Idee war, dass Simulated Annealing geeignet sein könnte, das MMAS von lokalen Optima zu bewahren. Es zeigte sich jedoch, dass die Laufzeit dieser Kombination nicht akzeptabel war, da Simulated Annealing nach jeder MMAS-Iteration ausgeführt wurde und zu lange für die Konvergenz benötigte. Bei kürzeren Cooling-Schedules mündete die Optimierung in schwächeren lokalen Optima als der reine MMAS. Eine der einfachsten lokalen Suchalgorithmen ist die *Coordinate Ascent*-Methode von Green et al. (1989). Der Optimierungsprozess wird mit einer zufällig erzeugten Produktlinie gestartet, deren Zielfunktionswert berechnet wird. Anschließend durchläuft der Algorithmus alle Eigenschaften der Produkte der Produktlinie in zufälliger Reihenfolge, berechnet dabei die Zielfunktionswerte der entstehenden Produktlinien und akzeptiert nur Produktlinien als neue beste Lösung, die einen höheren Zielfunktionswert als die bisherige beste Produktlinie liefern. Dieser Prozess endet, sobald keine Zielfunktionswertverbesserungen mehr erzielt werden. Belloni et al. (2008b) nennen die *Coordinate Ascent*-Methode, für die alle Zielfunktionswerte einer Eigenschaft eines Produkts in der Produktlinie berechnet wird *One-Opt*. Diese Bezeichnung wird übernommen.

Für die Verwendung von *One-Opt* im MMAS müssen Anpassungen vorgenommen werden. Es zeigte sich in den Testsimulationen, dass die Verwendung von der oben beschriebenen One-Opt-Methode zu frühzeitiger Konvergenz des MMAS_ls führt und somit zu schwächeren lokalen Optima. Außerdem verlängerte sich die Laufzeit von MMAS_ls inakzeptabel. Gewinnbringend für die Qualität der Zielfunktionswerte war allerdings die Anpassung, dass in jeder Iteration

des MMAS_ls nicht solange eine lokale Suche durchgeführt wird bis keine Verbesserung in den Zielfunktionswerten mehr erzielt wird, sondern dass jede Eigenschaft genau einmal in zufälliger Reihenfolge besucht wird. Damit wird zwar nicht unbedingt das lokale Optimum der verwendeten Produktlinie gefunden, aber der gesamte Algorithmus wird vor frühzeitiger Konvergenz bewahrt und die Laufzeit ist vertretbar. An dieser Stelle sei erwähnt, dass Stützle/Hoos (1997) eine konstante Anzahl von Ameisen in jeder Iteration verwenden, wobei jede Ameise eine lokale Suche erfährt. Dies ist für die Produktlinienoptimierung, zumindest bei einer Implementierung mit GNU R nicht möglich, da die Laufzeit des Algorithmus nicht mehr annehmbar wäre. Aus diesem Grund wird nur die beste Lösung in jeder Iteration einer lokalen Suche unterzogen.

Die Initialisierung, die minimalen und maximalen Pheromonmengen, die Anzahl der Ameisen sowie die Abbruchbedingungen entsprechen denen des MMAS. Der einzige Unterschied ist die Verwendung eines kleineren Evaporisationskoeffizient $\rho = 0.80$ (Stützle/Hoos 2000, S. 905). Mit diesem kleineren Wert wird die Explorationsfähigkeit gestärkt, was wichtig ist, um einer vorzeitigen Konvergenz entgegenzuwirken und die Diversität zu erhalten, um aus lokalen Optima zu entkommen. Als gegensätzlicher Mechanismus wirkt die lokale Suche, One-Opt, um nach besseren Zielfunktionswerten in der Nachbarschaft zu suchen. Die übrigen Parameter entsprechen denen des MMAS und sind in Tabelle 17 illustriert.

4.5.3 Monte-Carlo-Simulation zur Auswahl eines geeigneten Ameisenalgorithmus

In diesem Abschnitt werden ACO, MMAS und MMAS_ls miteinander verglichen. Zusammenfassend werden in Tabelle 17 die Parameterkonfigurationen der Algorithmen dargestellt. Für die Monte-Carlo-Simulation wurden die gleichen Datensätze wie in Abschnitt 4.3.2 und Tabelle 8 für den GA verwendet. Für alle Ameisenalgorithmen wird die Anzahl der Ameisen (#Ameisen) in Abhängigkeit von der Problemgröße bestimmt. Sie ergeben sich als Summe der Eigenschaftsausprägungen. Stützle/Hoos (1997) empfehlen, dass die Anzahl der Ameisen der Anzahl der Entscheidungsvariablen entspricht. Für die Produktlinienoptimierung steigt jedoch die Laufzeit der Algorithmen zu schnell, da zu viele Zielfunktionswertevaluationen bis zur Konvergenz des Algorithmus durchgeführt werden müssten. Die maximale Anzahl an Iterationen, die ein Ameisenalgorithmus benötigen darf, bestimmt sich aus dem abgerundeten Quotienten aus einer festen Zahl 250000 geteilt durch die Anzahl der Ameisen. Die Zahl 250000 wurde gewählt, um mehr Kontrolle über die Laufzeit zu haben. Umso größer die Probleminstanz ist, desto weniger Iterationen sind möglich. Diese Bedingung wird in der späteren Simulation mit den anderen Algorithmen dieser Arbeit fallen gelassen. Für die Zwischensimulation ist es zweckdienlich. Alle Ameisenalgorithmen werden nach spätestens 300 Sekunden terminiert, falls keine der anderen

Abbruchbedingungen eher eintrifft.

	ACO	**MMAS**	**MMAS_ls**
Initialisierung	$\tau_{kl}(1) = 1 \; \forall k,l$	$\tau_{kl}(1) = \tau_{max}(0) \; \forall k,l$	$\tau_{kl}(1) = \tau_{max}(0) \; \forall k,l$
#Ameisen = ...	$\sum_{k=1}^{K} L_k$		
$\rho = ...$	1.00	0.98	0.80
$\tau_{kl}(t+1) = \rho\,\tau_{kl}(t)...$	$+1(k,l) \in Z^{max}(t) \forall t$	$+ Z^{max}(t) \; \forall t$	$+ Z^{max}(t) \; \forall t$
$\tau_{max}(t) = ...$	–	$\frac{1}{1-\rho} \cdot Z^{max}(t) \; \forall t$	$\frac{1}{1-\rho} \cdot Z^{max}(t) \; \forall t$
$\tau_{min}(t) = ...$	–	$\frac{\tau_{max}(t)}{2 \cdot \#\text{Ameisen}}$	$\frac{\tau_{max}(t)}{2 \cdot \#\text{Ameisen}}$
Abbruchbedingung	$Iter = \left\lfloor \frac{250000}{\#\text{Ameisen}} \right\rfloor \; \vee \; \text{Time} = 300s \; \vee \; Z^{max}(t) = Z^{max}(t-500)$		

Tabelle 17: Übersicht über die verwendeten Parameterkonfigurationen der Ameisenalgorithmen. (Quelle: Eigene Darstellung)

Die Ergebnisse der Monte-Carlo-Simulation aus Tabelle 18 sind eindeutig. Zunächst fällt das schlechte Abschneiden des ACO nach dem Modell von Albritton/McMullen (2007) für alle Zielfunktionen ins Auge. Die Ursache ist vor allem das ungeeignete Pheromonupdate, das nur stattfindet, falls eine neue beste Lösung gefunden wurde. Dadurch wird die Konvergenz des Algorithmus ab einer bestimmten Problemgröße nahezu verhindert, da zu wenige Updates stattfinden, um die Auswahlwahrscheinlichkeiten der Eigenschaftsausprägungen gegen 1 oder 0 gehen zu lassen. Dieses Verhalten zeigt sich in Abbildung 8 (oben links), wo die Auswahlwahrscheinlichkeiten nach wenigen Iterationen Werte weit entfernt von 0 oder 1 annehmen. Die Suche gleicht über viele Iterationen eher einer Zufallssuche, da die Pheromonmengen gleichverteilt mit dem Wert 1 initialisiert werden. Ein weiteres elementares Problem dieser Implementierung ist ein Evaporisationskoeffizient von $\rho = 1.0$, wodurch ungünstige Eigenschaftsausprägungen über die Iterationen nur sehr langsam aus dem „Gedächtnis“ des ACO verschwinden. In Kombination mit zu wenigen Pheromonupdates verstärkt sich dieser Nachteil noch. Die Zufallssuche erklärt auch die kurzen Laufzeiten des ACO. Wenn zu lange keine Zielfunktionswertverbesserung erreicht werden konnte, terminiert der Algorithmus.

Im Gegensatz dazu führen MMAS und MMAS_ls ein Pheromonupdate in jeder Iteration mit der bisherigen besten Lösung durch, was sich in den deutlich verbesserten Zielfunktionswerten und der Konvergenz der Auswahlwahrscheinlichkeiten gegen 1 und 0 zeigt. In Abbildung 8 ist beispielhaft das Konvergenzverhalten für MMAS (oben rechts) und MMAS_ls illustriert. So lässt sich erkennen, dass der MMAS langsamer als der MMAS_ls konvergiert. Interessant ist, dass sich während des Suchprozesses des MMAS mehrfach die Auswahlwahrscheinlichkeiten in unterschiedliche Richtungen bewegen, sobald neue beste Zielfunktionswerte gefunden wurden und schließlich gegen 0 oder 1 konvergieren. Dieser Effekt fällt beim MMAS_ls durch die lokale Suche deutlich schwächer aus, findet aber dennoch statt (siehe Shift zwischen zwei Eigenschaftsausprägungen zwischen den Iterationen 300 und 400). Des Weiteren konvergieren die Auswahlwahrscheinlichkeiten für die meisten Eigenschaftsausprägungen, wie erwartet, er-

Algorithmus	ZF	$\overline{\mathbf{ZF}}_{max}$	$\overline{\mathbf{ZF}}_{max}^{mean}$	$\overline{\mathbf{ZF}}_{mean}$	$\overline{\mathbf{ZF}}_{mean}^{mean}$	$\overline{\sigma}_{\mathbf{ZF}}$	ØT
ACO	G_{det}	0.365	0.632	0.307	0.595	0.219	8.96
	M_{det}	0.339		0.291		0.214	6.46
	G_{prob}	0.860		0.825		0.102	11.32
	M_{prob}	0.966		0.958		0.023	8.81
MMAS	G_{det}	0.934	0.972	0.858	0.939	0.079	27.56
	M_{det}	0.955		0.899		0.079	26.13
	G_{prob}	0.999		0.999		0.002	26.65
	M_{prob}	0.999		0.999		0.000	20.46
MMAS_ls	G_{det}	0.998	**0.999**	0.926	**0.962**	0.079	109.32
	M_{det}	0.999		0.921		0.061	102.55
	G_{prob}	0.999		0.999		0.000	97.52
	M_{prob}	0.999		0.999		0.000	80.27

Tabelle 18: Ergebnisse der Monte-Carlo-Simulation für die Ameisenalgorithmen. Legende: **ZF**... Zielfunktion; $\overline{\mathbf{ZF}}_{max}$... arithmetische Mittelwerte der Anteile an der besten gefundenen Lösung unter 10 Wiederholungen im Verhältnis zur besten gefundenen Lösung einer Probleminstanz aus allen Ameisenalgorithmen für jede Zielfunktion; $\overline{\mathbf{ZF}}_{max}^{mean}$ ... arithmetisches Mittel aus $\overline{\mathbf{ZF}}_{max}$ über alle Zielfunktionen; $\overline{\mathbf{ZF}}_{mean}$... arithmetische Mittelwerte über den Mittelwert der 10 Wiederholungen einer Probleminstanz als Anteil an der besten gefunden Lösung über alle Ameisenalgorithmen für jede Zielfunktion; $\overline{\mathbf{ZF}}_{mean}^{mean}$... arithmetisches Mittel von $\overline{\mathbf{ZF}}_{mean}$ aller Zielfunktionen; $\overline{\sigma}_{\mathbf{ZF}}$... arithmetische Mittelwerte der Standardabweichungen für jede Zielfunktion (berechnet aus den als Anteile an der besten Lösung transformierten Zielfunktionswerten); **ØT**... arithmetisches Mittel der benötigten Zeit in Sekunden über alle Probleminstanzen für jede Zielfunktion. Rundungsfehler können auftreten, da alle Werte aus den Originaldaten berechnet wurden. (Quelle: Eigene Darstellung)

heblich schneller durch die lokale Suche. Dies zeigt sich nochmals beispielhaft in der Grafik rechts unten (Abbildung 8), in der die gemittelten Zielfunktionswerte der Ameisenpopulation jeder Iteration der deterministischen Gewinnfunktion für MMAS_ls schneller (hier im Sinne von weniger Iterationen, nicht der Laufzeit) bessere Zielfunktionswerte ggü. MMAS annehmen (Hinweis: die Grafik rechts unten zeigt wegen illustrativen Zwecken einen anderen Durchlauf für die gleiche Probleminstanz als die drei anderen Grafiken mit mehr Iterationen für den ACO).

Der ACO verbessert seine Zielfunktionswerte zwar, man sieht jedoch, dass er über die Iterationen hinweg in ungünstigen Regionen des Lösungsraums, aufgrund der fast zufälligen Suche, „gefangen“ bleibt. Aufgrund des offensichtlich schlechten Abschneidens des ACO für jede Zielfunktion mit größeren Standardabweichungen wird sich auf den Vergleich zwischen MMAS und MMAS_ls konzentriert. An den Ergebnissen lässt sich ablesen, dass die Hinzunahme einer lokalen Suche (One-Opt) im Mittel zu signifikant verbesserten Zielfunktionswerten ($\overline{ZF}_{mean}^{mean}$) auf dem 5 %-Signifikanzniveau führt (siehe Tabelle 19; p-Wert = $2.94 \cdot 10^{-10}$). Auch für die bes-

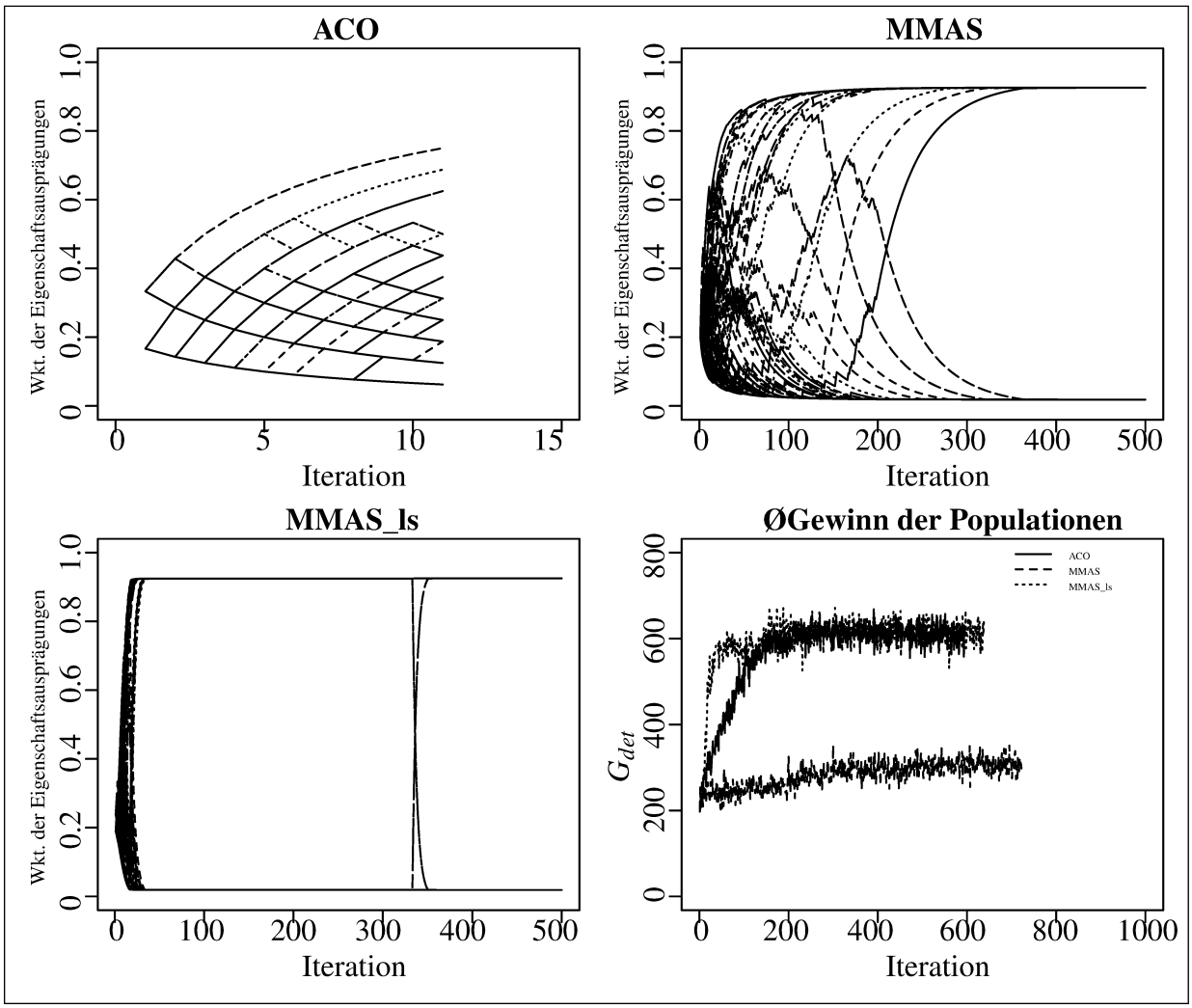

Abbildung 8: Beispielhafter Plot der Auswahlwahrscheinlichkeiten für ACO/MMAS/MMAS_ls der Eigenschaftsausprägungen und durchschnittliche Gewinne jeder Iteration. Die verwendete Zielfunktion ist G_{det} für fünf Eigenschaften mit je fünf Eigenschaftsausprägungen und drei Produkten in der Produktlinie. Hinweis: die Grafik rechts unten zeigt wegen illustrativen Zwecken einen anderen Durchlauf für die gleiche Probleminstanz als die drei anderen Grafiken. (Quelle: Eigene Darstellung)

ten gefundenen Zielfunktionswerte für jedes Problem ($\overline{ZF}_{max}^{mean}$) ist der Unterschied statistisch signifikant (p-Wert = 0.023). Der Unterschied zwischen den deterministischen Entscheidungsregeln ist sowohl für $\overline{ZF}_{max}(G_{det}, M_{det})$ (p-Wert = 0.022), als auch für $\overline{ZF}_{mean}(G_{det}, M_{det})$ (p-Wert = 0.003) statistisch signifikant. Kein signifikanter Unterschied konnte zwischen den Zielfunktionswerten der probabilistischen Entscheidungsregeln festgestellt werden ($\overline{ZF}_{max}(G_{prob}, M_{prob})$: p-Wert = 0.373 und $\overline{ZF}_{mean}(G_{prob}, M_{prob})$: p-Wert = 0.232). Der Unterschied in den Laufzeiten zwischen MMAS und MMAS_ls ist statistisch signifikant (p-Wert = $3.68 \cdot 10^{-11}$). Die verbesserte Performance des MMAS_ls geht somit auf die Kosten der Laufzeit.

	ACO	**MMAS**
MMAS	$2.32 \cdot 10^{-32}$	
MMAS_ls	$1.16 \cdot 10^{-31}$	$2.94 \cdot 10^{-10}$

Tabelle 19: Übersicht der paarweisen p-Werte eines T-Tests für verbundene Stichproben für alle Ameisenalgorithmen. Nullhypothese: das arithmetische Mittel des einen Algorithmus ist gleich dem arithmetischen Mittel des anderen Algorithmus. Stichprobengröße: 720 Zielfunktionswerte. Gewähltes Signifikanzniveau $\alpha = 0.05$. (Quelle: Eigene Darstellung)

Ähnlich wie beim GA werden dennoch der MMAS sowie der MMAS_ls für die weiteren Simulationsrechnungen gegen die anderen Algorithmen berücksichtigt, um die Performance der „reinen“ Implementierungen miteinander vergleichen zu können.

4.6 Simulated Annealing in der Produktlinienoptimierung

Simulated Annealing (SA) wird in der Publikation von Belloni et al. (2008b) verwendet. SA schnitt in dieser Publikation bzgl. der besten und der durchschnittlichen Zielfunktionswerte am besten ab. Da es sich bei der Produktlinienoptimierung um Maximierungsprobleme handelt, muss die Akzeptanzwahrscheinlichkeit aus Abschnitt 3.6 in der Art angepasst werden, dass die Zielfunktionswerte Z_t und Z_{t+1} vertauscht werden müssen bzw., dass die Differenz dieser Zielfunktionswerte mit -1 multipliziert werden muss. Daraus ergibt sich für die Akzeptanzwahrscheinlichkeit für die Produktlinienoptimierungsprobleme:

$$P\{\text{akzeptiere Zustand } t+1\} = \begin{cases} 1 & \text{wenn } Z_{t+1} \geq Z_t \\ \exp\left(\frac{Z_{t+1}-Z_t}{c}\right) & \text{wenn } Z_{t+1} < Z_t \end{cases} \tag{4.16}$$

Belloni et al. (2008a, S. 2) nutzen zwei verschiedene Cooling-Schedule, die sie durch Probieren für die simulierten Conjointdatensätze und den realen Conjointdatensatz austariert haben. Für den Parameter α aus 3.24 zur Steuerung des Kontrollparameters c wird für die simulierten Datensätze ein Wert von $\alpha = 0.9$ und für den realen Conjointdatensatz ein Wert $\alpha = 0.8$ verwendet. Die beiden Cooling-Schedules lauten somit, wobei der Endwert jeweils bei 0.1 gesetzt wurde:

- Für die simulierten Datensätze: 21.0, 18.9, 17.0, 15.3, 13.8, 12.4, 11.2, 10.0, 9.0, 8.1, 7.3, 6.6, 5.9, 5.3, 4.8, 4.3, 3.9, 3.5, 3.2, 2.8, 2.6, 2.3, 2.1, 1.9, 1.7, 1.5, 1.4, 1.2, 1.1, 0.1 und

- für den realen Conjointdatensatz: 1443, 1154, 923, 739, 591, 473, 378, 303, 242, 194, 155, 124, 99, 79, 63, 51, 41, 32, 26, 21, 17, 13, 11, 9, 7, 5, 4, 3, 0.1.

Für jeden Wert von c werden 10000 Änderungen in den Eigenschaftsausprägungen durchgeführt, was bei den vorliegenden Cooling-Schedules der Länge 29 in der Berechnung zu insgesamt 290000 Zielfunktionsevaluationen führt. Hier zeigt sich bereits, was das Problem bei SA ist: der Cooling-Schedule hängt von der Probleminstanz ab, spezieller: von den Zielfunktionswerten. Deshalb muss für die vier Produktlinienoptimierungszielfunktionen dieser Arbeit eine allgemeingültigere Lösung dieses Problems gefunden werden. Dies wird über zwei Mechanismen versucht: zum einen soll ein akzeptabler Startwert des Cooling-Schedules bestimmt werden, von welchem aus das Abkühlen linear und exponentiell mit vorher definierter Länge des Cooling-Schedules stattfindet, bis ein vorher definierter Endwert nahe 0 erreicht ist. Zum anderen wird ein adaptives Vorgehen nach den Gesetzen der statistischen Mechanik von Aarts/-Korst (1989) aus Abschnitt 3.6 verfolgt. Die Startlösung für den Suchprozess wird in allen Fällen zufällig erzeugt.

4.6.1 Anpassungen der Cooling-Schedules für die Produktlinienoptimierung

In diesem Abschnitt werden die nötigen Anpassungen der Cooling-Schedules für das Produktlinienoptimierungsproblem beschrieben. Ursprünglich sollten ein linearer, exponentieller und adaptiver Cooling-Schedule miteinander verglichen werden. In den Vorsimulationen zeigte sich jedoch, dass ein linearer Cooling-Schedule zu sehr schlechten Zielfunktionswerten führt, da er zu langsam konvergiert. Die Suche gleicht zu lange einer Zufallssuche, in der häufige Sprünge in andere Lösungsraumregionen durchgeführt werden, ohne die Suche in vielversprechenden Regionen intensivieren zu können. Aufgrund der langen Laufzeiten wird daher auf den linearen Cooling-Schedule für den Vergleich verzichtet. Eine genau Beschreibung des Vorgehens erfolgt in den kommenden Abschnitten, da verhältnismäßig viele Anpassungen an die Cooling-Schedules für die Produktlinienoptimierung gemacht werden müssen.

4.6.1.1 Exponentieller Cooling-Schedule

Beim exponentiellen Cooling-Schedule kommt es auf mehrere Parameter an: Start- und Endwert, Länge und die Anzahl der Zielfunktionsevaluationen pro Temperatur. Um die Vergleichbarkeit mit dem adaptiven Cooling-Schedule zu verbessern, wird eine feste Länge des Cooling-Schedules von $C = 29$ mit fester Anzahl von Zielfunktionswertevaluationen i.H.v. 10000 pro Temperatur gemäß Belloni et al. (2008b) verwendet, was in insgesamt 290000 Zielfunktionswertevaluationen mündet. Die Start- und Endwerte für den Cooling-Schedule müssen zumindest an die Art der Zielfunktion (Gewinn und Marktanteil) angepasst werden, um einen geeigneten Cooling-Schedule zu erzeugen. So wird für den Gewinn ein Endwert $c_{end} = 0.1$ wie in Belloni et al. (2008b) und für den Marktanteil ein Endwert $c_{end} = 0.0001$ verwendet. Die Anpassung für die Marktanteilszielfunktionen ist nötig, da der Cooling-Schedule theoretisch gegen 0 konvergieren soll. Ein Endwert von $c_{end} = 0.1$ entspräche allerdings im Rahmen eines Marktanteils einem sehr großen Cooling-Schedule-Endwert, wodurch der Quotient im Exponenten in 4.16 gegen Ende der Suche nicht klein genug wird, um schlechtere Zustände nicht mehr zu akzeptieren. Der Quotient im Exponenten ist immer negativ (falls positiv, wird Z_{t+1} immer akzeptiert, da es eine Zielfunktionswertverbesserung gab) und sollte zum Ende der Suche hin möglichst klein werden, da die Akzeptanzwahrscheinlichkeit des schlechteren Zustands dann gegen 0 geht, was am Ende der Suche nicht nur erwünscht, sondern elementar für die lokale Suche von SA ist. Für die Wahl des Startwerts ist es wichtig, eine geeignete Höhe der Zielfunktionswerte zu bestimmen. Ein geeigneter Startwert c_0 ist ein Wert, der zu Beginn der Suche viele neue Lösungen akzeptiert, also ein Wert, bei dem die Akzeptanzwahrscheinlichkeit relativ hoch ist. Dies wird erreicht, wenn der Quotient $\frac{Z_{t+1}-Z_t}{c_0}$, falls $Z_{t+1} < Z_t$, nah genug an 0 liegt. Wird c_0 zu groß gewählt, gleicht die Suche für viele Iterationen einer Zufallssuche, weil

zu viele neue Zustände akzeptiert werden. Ist c_0 zu klein, wird zu zeitig die lokale Suche gestärkt, was dazu führt, dass der SA zu schnell gegen lokale Optima konvergiert. Deshalb wird c_0 bestimmt, indem eine einfache Heuristik, One-Opt, dem SA vorgeschaltet wird. Der Startwert c_0 entspricht dann dem Mittelwert der von One-Opt gefundenen Zielfunktionswerte. Die Mittelwertbildung hat zum Ziel einen geeigneten Startwert zu ermitteln, der weder zu groß, noch zu klein für einen effektiven Suchprozess ist. Es wird eine angemessene Balance zwischen Exploration und lokaler Suche des Suchraums erzielt. Die Anzahl der berücksichtigten Lösungen richtet sich nach der Problemgröße. So wird ausgehend von einer zufällig erzeugten Startlösung jede Eigenschaftsausprägung jeder Eigenschaft jedes Produkts in der Produktlinie genau einmal berücksichtigt, wodurch für den Mittelwert $R \cdot \sum_{k=1}^{K} L_k, k = 1,2,...,K$, mögliche Lösungen mit zugehörigen Zielfunktionswerten verwendet werden, wobei R für die Anzahl der Produkte in einer Produktlinie steht und L_k die Anzahl der Eigenschaftsausprägungen einer Eigenschaft k darstellt. Somit hat man problemabhängig einen geeigneten Startwert bestimmt, der zur jeweiligen Probleminstanz passt und einfach zu bestimmen ist. Die Temperaturen eines exponentiellen Cooling-Schedules ergeben sich zu (siehe Abschnitt 3.24):

$$c_{m+1} = \alpha \cdot c_m, \quad m = 1,2,..., \tag{4.17}$$

Bei vorgegebener Länge des Cooling-Schedules und vorgegebenen Start- und Endwerten bestimmt sich der Wert für α zu:

$$\alpha = \left(\frac{c_{end}}{c_0} \right)^{\frac{1}{C-1}} , \tag{4.18}$$

was sich aus dem Umstellen der Gleichung $c_{end} = c_0 \cdot \alpha^{C-1}$ ergibt. Somit werden die Temperaturen des Cooling-Schedules, die zwischen c_0 und c_{end} liegen, über folgende Bildungsvorschrift berechnet:

$$c_m = c_0 \cdot \alpha^{m-1} \, , m = 1,2,...29, \tag{4.19}$$

wobei dann $c_1 = c_0$ und $c_{29} = c_{end}$ entsprechen.

Diese Überlegungen erzeugen einen problemabhängigen, geeigneten, empirischen Cooling-Schedule, der für jede Zielfunktion zu brauchbaren Zielfunktionswerten führt. Der Algorithmus wird beendet, wenn entweder 290000 Zielfunktionsevaluationen stattgefunden haben, für 150000 aufeinanderfolgende Zielfunktionsevaluationen keine Verbesserung des bisher besten

Zielfunktionswert erreicht werden konnte oder wenn SA die Zeitschranke von 300 Sekunden überschreitet.

4.6.1.2 Adaptiver Cooling-Schedule

Für den adaptiven Cooling-Schedule aus Abschnitt 3.6.1 müssen einige Anpassungen für die Produktlinienoptimierung vorgenommen werden. Die Anpassung der Bildungsvorschriften für die Parameter für Maximierungsprobleme wurde bereits in Abschnitt 3.6.2 durchgeführt und kann daher direkt übernommen werden. Beim adaptiven Cooling-Schedule passen sich die Temperaturen während des Suchprozesses in Abhängigkeit der gefundenen Lösungen mit deren Zielfunktionswerten dynamisch an. Der Startwert c_0 wird wie in Abschnitt 3.6.2 nach Bildungsvorschrift 3.23 mit initialer Akzeptanzrate $\chi_0 = 0.95$ gebildet. Die weiteren Temperaturen werden gemäß Bildungsvorschrift 3.24 ausgehend von c_0 adaptiv während des Suchprozesses in Abhängigkeit eines Distanzparameters δ und der Standardabweichungen der Zielfunktionswerte σ_{c_m} bestimmt. Die benötigten Zielfunktionswerte werden in einem One-Opt-Durchlauf erzeugt, womit die benötigten Parameter zur Bestimmung von c_0 nach 3.23 bestimmt werden können. Der Parameter $\delta = 0.1$ wurde gemäß Aarts/Korst (1989, S. 72) gewählt, welcher die Geschwindigkeit des Abkühlens des Kontrollparameters c_m determiniert. Für die Standardabweichung der bisherigen Zielfunktionswerte σ_{c_m} wird von Aarts/Korst (1989, S. 64) empfohlen, nicht alle Zielfunktionswerte des bisherigen Suchprozesses zu verwenden, da es sonst zu einer vorzeitigen Konvergenz von SA führen kann. Deshalb wird nur eine bestimmte Anzahl von Zielfunktionswerten ausgehend vom aktuellen Zielfunktionswert verwendet. Da in der Literatur hierzu keine Angaben gemacht wurden, wird für die Anzahl der Zielfunktionswerte die Anzahl der Entscheidungsvariablen einer Probleminstanz genutzt, die bei $R \cdot \sum_{k=1}^{K} L_k, k = 1,2,...,K$, liegt. Dieser Wert hat sich in empirischen Voruntersuchungen als geeignet erwiesen und kann aus der vorliegenden Probleminstanz berechnet werden. Für die Länge der Markov-Kette einer Temperatur schlagen Aarts/Korst (1989, S. 65) die Anzahl der Entscheidungsvariablen vor. Für die Produktlinienoptimierung hat sich herausgestellt, dass es vorteilhafter ist, für jede Temperatur die Anzahl der Entscheidungsvariablen *eines* Produkts zu verwenden und dafür eine größere Anzahl von Temperaturen zu verwenden, um schlussendlich kleinere Temperaturwerte näher an 0 zu erhalten. Denn wie sich in Abbildung 9 zeigt, verläuft die Abnahme der Temperatur zunächst sehr steil und anschließend sehr flach. Die Voraussetzung hierfür ist allerdings eine endliche Anzahl von Zielfunktionswertevaluationen, die, um eine bessere Vergleichbarkeit herzustellen, insgesamt bei maximal 290000 liegt. Eine wichtige Anpassung an die Produktlinienoptimierung benötigt das vorgeschlagene Abbruchkriterium 3.29 aus Abschnitt 3.6.3. Es zeigte sich, dass dieses Abbruchkriterium für die Produktlinienoptimierung nicht geeignet ist, da es nie erfüllt wird. Der Grund liegt darin, dass der Quotient $\frac{\sigma_{cm}^2}{\frac{c_m}{|\mathbb{S}|}\sum_{t\in\mathbb{S}} Z(t)}$ nie kleiner als die geforderte Schranke ε wird, da die Nachbarschaftslösungen zu stark oszillieren. In vielen Fällen reicht bereits die Veränderung einer Eigenschaft eines einzelnen Produkts in der Produktlinie, um

den Zielfunktionswert stark zu verändern, wodurch sich keine Konvergenz der Standardabweichung der Zielfunktionswerte gegen einen so kleinen Wert einstellt, wie er für das Abbruchkriterium benötigt wird. Daher werden vordefinierte Abbruchkriterien verwendet. SA terminiert, wenn entweder 290000 Zielfunktionsevaluationen stattgefunden haben, für 150000 aufeinanderfolgende Zielfunktionsevaluationen keine Verbesserung des bisher besten Zielfunktionswert erreicht werden konnte oder wenn SA die Zeitschranke von 300 Sekunden überschreitet.

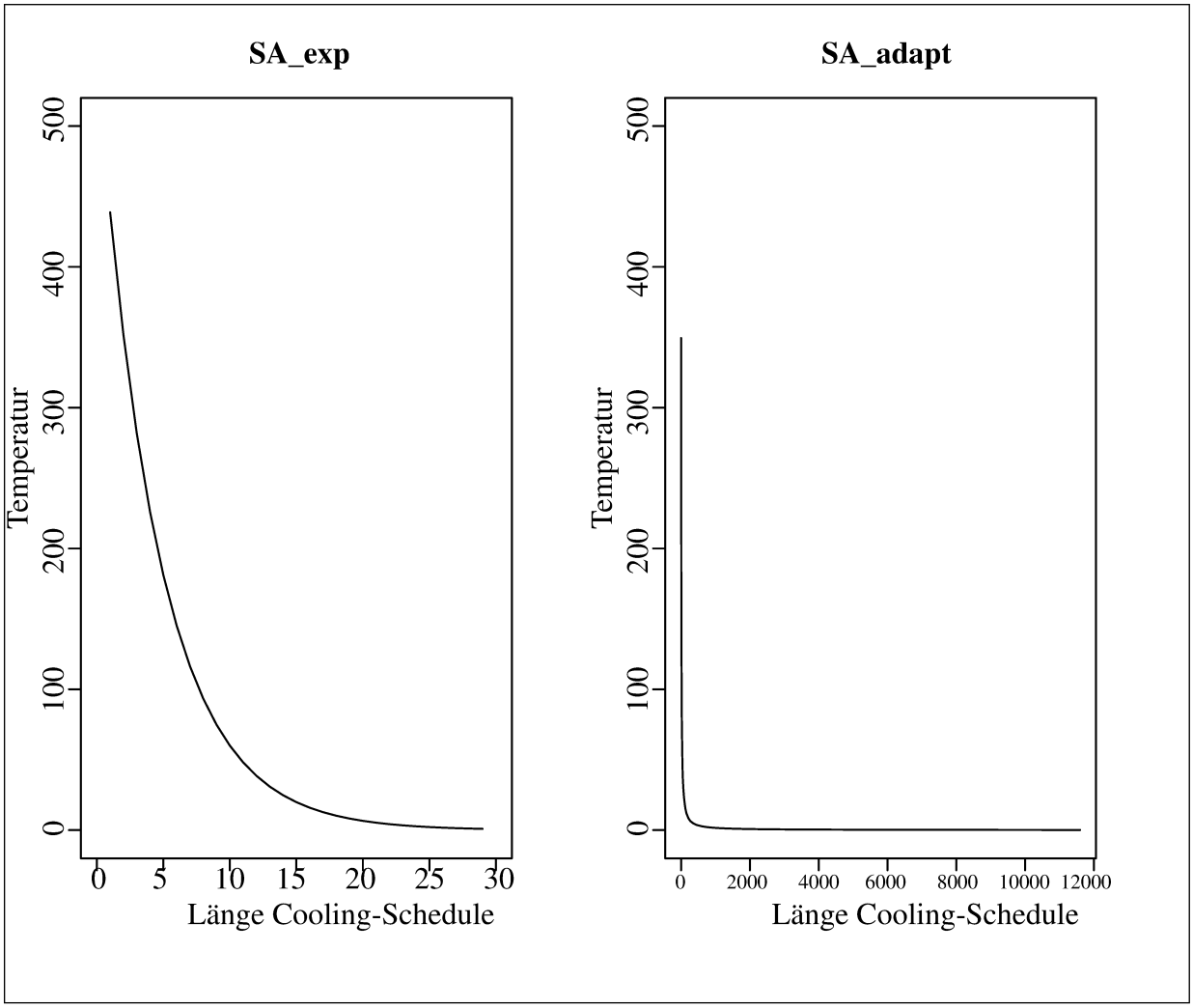

Abbildung 9: Beispielhafte Cooling-Schedules für SA mit exponentiellem Cooling-Schedule der vordefinierten Länge 29 mit Temperaturendwert $c_{end} = 0.1$ und für SA mit adaptivem Cooling-Schedule der Länge 11601 und Temperaturendwert nach Konvergenz von $c_{end} = 0.127$. Die verwendete Zielfunktion ist G_{det} für fünf Eigenschaften mit je fünf Eigenschaftsausprägungen und drei Produkten in der Produktlinie. (Quelle: Eigene Darstellung)

4.6.2 Monte-Carlo-Simulation zur Auswahl des Cooling-Schedules

In diesem Abschnitt werden die beiden Cooling-Schedules miteinander verglichen und schlussendlich derjenige ausgewählt, der für die weiteren Simulationen verwendet wird. In Abbildung

9 zeigt sich der Verlauf der Cooling-Schedules. Der adaptive Cooling-Schedule verläuft anfangs sehr steil und fällt im weiteren Verlauf immer flacher ab. Des Weiteren besitzt der adaptive Cooling-Schedule erheblich mehr Temperaturen als der exponentielle. Außerdem weisen beiden Cooling-Schedules einen ähnlichen Endwert für die Temperaturen auf, wobei der exponentielle Cooling-Schedule mit einem höheren Startwert beginnt. In Tabelle 20 sind die vormals beschriebenen Parameter aufgeführt. Das Signifikanzniveau für die statistischen Untersuchungen wird auf 5 % gesetzt. Für die Monte-Carlo-Simulation wurden die gleichen Datensätze wie in Abschnitt 4.3.2 und Tabelle 8 für den GA verwendet.

	SA_exp	**SA_adapt**
Kontrollparameter c_{m+1}	$\alpha \cdot c_m$	$\frac{c_m}{1+\frac{c_m \cdot \ln(1+\delta)}{3\sigma_{c_m}}}$
Startwert c_0 (mit Z aus One-Opt)	$\overline{Z}$	$\frac{\overline{\Delta Z}^{(+)}}{\ln(\frac{s_2}{s_2 \cdot \chi_0 - s_1(1-\chi_0)})}$
Endwert c_{end}	$= \begin{cases} 0.1000 & \text{Gewinn} \\ 0.0001 & \text{Marktanteil} \end{cases}$	adaptiv
Distanzparameter δ	-	0.1
initiale Akzeptanzwahrscheinlichkeit χ_0	-	0.95
Länge des Cooling-Schedules	29	adaptiv
Anzahl Evaluationen pro Temperatur	10000	$\sum_{k=1}^{K} L_k$
maximale Anzahl Evaluationen	290000	290000
Abbruchkriterien	maximale Anzahl Evaluationen $\vee$ Time $= 300s$ $\vee \nexists Z^{max}(t, t+1, ..., t+150000) \geq Z^{max}$	

Tabelle 20: Übersicht über die verwendeten Parameterkonfigurationen von Simulated Annealing. (Quelle: Eigene Darstellung)

Die in Tabelle 21 aufgeführten Ergebnisse sind nicht ganz eindeutig. Über alle Zielfunktionen hinweg, $\overline{ZF}^{mean}_{mean}$, erzielt SA_adapt bessere Zielfunktionswerte (p-Wert $= 4.10 \cdot 10^{-4}$). Für die im Mittel besten Zielfunktionswerte der Durchläufe, $\overline{ZF}^{mean}_{max}$, erzeugt SA_adapt ebenfalls bessere Ergebnisse als SA_exp (p-Wert $= 6.70 \cdot 10^{-3}$). Es fällt auf, dass SA_adapt bezogen auf $\overline{ZF}^{mean}_{mean}(G_{prob}, M_{prob})$ besser für die probabilistische Entscheidungsregel geeignet ist (p-Wert $= 4.49 \cdot 10^{-25}$) und SA_exp bezogen auf $\overline{ZF}^{mean}_{mean}(G_{det}, M_{det})$ besser für die deterministische Entscheidungsregel (p-Wert $= 2.30 \cdot 10^{-6}$). Bezüglich der mittleren besten Zielfunktionswerte, gibt es statistisch keinen signifikanten Unterschied zwischen den Cooling-Schedules für die deterministische Entscheidungsregel ($\overline{ZF}^{mean}_{max}(G_{det}, M_{det})$, p-Wert $= 0.29$) und für die probabilistische Entscheidungsregel, $\overline{ZF}^{mean}_{max}(G_{prob}, M_{prob})$, gibt es einen statistisch signifikanten Unterschied zugunsten von SA_adapt (p-Wert $= 8.30 \cdot 10^{-4}$). Die Unterschiede lassen sich dadurch erklären, dass SA mit exponentiellem Cooling-Schedule durch die höheren Startwerte für c_0 und dem längeren Verharren in den Temperaturen stärker in der Exploration des Suchraums ist, was für die deterministische Entscheidungsregel, wenn auch nicht bei den maximalen Zielfunktionswerten, aber zumindest im Mittel vorteilhaft zu sein scheint. Die besseren Zielfunktionswerte für die probabilistische Entscheidungsregel bei dem adaptiven Cooling-Schedule erklären sich

durch ein langsameres Abflachen der Temperaturen c, was vorteilhaft für die lokale Suche zu sein scheint und offenbar wichtig für das Finden guter Lösungen in Suchräumen ist, in denen die Höhe der Zielfunktionswerte näher beieinander liegt.

Algorithmus	**ZF**	$\overline{\mathbf{ZF}}_{max}$	$\overline{\mathbf{ZF}}_{max}^{mean}$	$\overline{\mathbf{ZF}}_{mean}$	$\overline{\mathbf{ZF}}_{mean}^{mean}$	$\overline{\sigma}_{\mathbf{ZF}}$	**ØT**
SA_exp	G_{det}	0.993	0.965	0.946	0.938	0.047	234.15
	M_{det}	0.986		0.939		0.054	229.60
	G_{prob}	0.894		0.884		0.127	230.44
	M_{prob}	0.986		0.985		0.023	222.31
SA_adapt	G_{det}	0.980	**0.992**	0.910	**0.959**	0.063	235.99
	M_{det}	0.986		0.925		0.056	231.34
	G_{prob}	1.000		0.999		0.000	234.55
	M_{prob}	1.000		0.999		0.000	223.16

Tabelle 21: Übersicht der Ergebnisse für den exponentiellen (SA_exp) und den adaptiven (SA_adapt) Cooling-Schedule für Simulated Annealing. Legende: **ZF**... Zielfunktion; $\overline{\mathbf{ZF}}_{max}$... arithmetische Mittelwerte der Anteile an der besten gefundenen Lösung unter 10 Wiederholungen im Verhältnis zur besten gefundenen Lösung einer Probleminstanz aus allen SA für jede Zielfunktion; $\overline{\mathbf{ZF}}_{max}^{mean}$ ... arithmetisches Mittel aus $\overline{\mathbf{ZF}}_{max}$ über alle Zielfunktionen; $\overline{\mathbf{ZF}}_{mean}$... arithmetische Mittelwerte über den Mittelwert der 10 Wiederholungen einer Probleminstanz als Anteil an der besten gefundenen Lösung über alle SA für jede Zielfunktion; $\overline{\mathbf{ZF}}_{mean}^{mean}$... arithmetisches Mittel von $\overline{\mathbf{ZF}}_{mean}$ aller Zielfunktionen; $\overline{\sigma}_{\mathbf{ZF}}$... arithmetische Mittelwerte der Standardabweichungen für jede Zielfunktion (berechnet aus den als Anteile an der besten Lösung transformierten Zielfunktionswerten); **ØT**... arithmetisches Mittel der benötigten Zeit in Sekunden über alle Probleminstanzen für jede Zielfunktion. Rundungsfehler können auftreten, da alle Werte aus den Originaldaten berechnet wurden. (Quelle: Eigene Darstellung)

Die Standardabweichungen unterscheiden sich nicht signifikant, wenn alle Zielfunktionswerte über alle Zielfunktionen betrachtet werden (p-Wert $= 0.06$). Betrachtet man jedoch die Entscheidungsregeln getrennt, zeigen sich signifikante Unterschiede für die probabilistische Entscheidungsregel zugunsten des SA_adapt (p-Wert $= 4.18 \cdot 10^{-5}$) und signifikante Unterschiede für die deterministische Entscheidungsregel zugunsten von SA_exp (p-Wert $= 9.90 \cdot 10^{-4}$). Die Laufzeiten der SA-Implementierungen unterscheiden sich kaum, aber statistisch signifikant zugunsten von SA_exp (p-Wert $= 8.12 \cdot 10^{-20}$). Dies liegt vor allem daran, dass meistens die beiden Abbruchkriterien Laufzeit und die maximale Anzahl von Zielfunktionsevaluationen gegriffen haben, wodurch sich die Laufzeiten kaum unterscheiden. Daraus kann man schließen, dass das Konvergenzverhalten beider Cooling-Schedules ähnlich ist.

Für die weiteren Simulationen in dieser Arbeit wird SA_adapt verwendet. Die Gründe hierfür sind theoretischer und empirischer Natur. Der adaptive Cooling-Schedule bietet ein starkes theoretisches Fundament auf Basis der statistischen Mechanik aus dem Bereich Physik, während der hier verwendete exponentielle Cooling-Schedule empirisch „erraten“ werden muss. Die Vorteilhaftigkeit eines adaptiven Cooling-Schedules zeigt sich vor allem, wenn alle Ziel-

funktionen gemeinsam betrachtet werden. Das ist das Ziel dieser Arbeit: einen Algorithmus einer Gattung zu finden, der für alle Zielfunktionen gute Ergebnisse liefert. Der exponentielle Cooling-Schedule ist lediglich für die mittleren Zielfunktionswerte über alle Zielfunktionen mit deterministischer Entscheidungsregel signifikant. Bei den mittleren maximalen Zielfunktionswerten gibt es allerdings keinen Unterschied mehr. Der Unterschied in den mittleren Laufzeiten ist aufgrund der geringen Differenz zu vernachlässigen. Die Höhe der Standardabweichungen ist keine Entscheidungshilfe, da sie keine statistisch signifikanten Unterschiede über alle Zielfunktionen aufdecken konnte. Insofern ist es gelungen einen adaptiven Cooling-Schedule in die Produktlinienoptimierung mit starkem theoretischen Fundament einzuführen und seine Überlegenheit und höhere Flexibilität gegenüber dem bisher genutzten Cooling-Schedule von Belloni et al. (2008b) zu beweisen.

Am Ende dieses Kapitels lässt sich die *Forschungsfrage 3* positiv beantworten. Ein adaptives Vorgehen für die Verfahrensparameter bringt deutliche Vorteile gegenüber fest definierten Parameterwerten.

5 Simulationsstudie

In diesem Kapitel wird der Aufbau und der Ablauf der Simulationsstudie beschrieben. Anschließend werden die Simulationsergebnisse einer statistischen Analyse unterzogen und interpretiert. Es werden sechs Verfahren für vier Zielfunktionen auf ihre Performance hin untersucht. Performance bedeutet in diesem Zusammenhang die Qualität des Zielfunktionswertes. Für die direkte Vergleichbarkeit der Verfahren käme z. B. die Laufzeit der Verfahren infrage. Die Laufzeit eines Verfahrens, also die Dauer bis eine approximativ optimale Produktlinie erreicht wurde, wird nicht nur von ihrer allgemeinen Funktionsweise beeinflusst, sondern z. B. auch durch die verwendete Programmiersprache oder bestimmte Implementierungstechniken wie die Vektorisierung bestimmter Operatoren. Eine weitere Möglichkeit, um eine bessere Vergleichbarkeit der Verfahren herzustellen, könnte die Anzahl der untersuchten Produktlinien jedes Verfahrens sein. Im Vorhinein ist jedoch kaum möglich abzuschätzen, bei welcher Anzahl untersuchter Produktlinien ein Verfahren konvergiert. Die Konvergenz eines Verfahrens ist jedoch entscheidend für dessen Performance, weshalb die Konvergenz eines Verfahrens als wichtiger für die Simulationsstudie angesehen wird, als eine ohnehin kaum mögliche Vergleichbarkeit über die Anzahl der untersuchten Produktlinien. Ferner bewegt sich die vorliegende Arbeit damit im Rahmen des in der Literatur üblichen Vorgehens im Bereich der Verfahren, nicht nur für die Produktlinienoptimierung. Alle Verfahren wurden mit der Programmiersprache *R* implementiert.

5.1 Conjoint-Parameter für die Simulationsstudie

Die Auswahl der Parameter für die Monte-Carlo-Simulation erfolgt in der Literatur zumeist willkürlich. In der vorliegenden Arbeit werden die Parameter aus 2089 durchgeführten Conjointstudien hergeleitet, die von einem großen Marktforschungsunternehmen in den Jahren von 1996–2011 durchgeführt wurden. Die herzuleitenden Parameter sind die folgenden:

- Die Anzahl der Eigenschaften,
- die Anzahl der Eigenschaftsausprägungen,

- die Anzahl der Konsumenten,
- die Anzahl der Produkte in einer Produktlinie,
- die Verteilung der individuellen Teilnutzenwerte und
- die Größe des Status-quo-Markts.

Die Häufigkeit des Vorkommens der Anzahl der Eigenschaften kann aus Abbildung 10 entnommen werden. Ca. 76 % der untersuchten Conjointstudien besitzen zwischen 3-9 Eigenschaften. 354 Studien, das entspricht ca. 17 %, haben sieben Eigenschaften, was in dieser Stichprobe die größte Häufigkeit dargestellt. Für die Anzahl der Eigenschaften der Monte-Carlo-Simulation wird die Häufigkeitsverteilung der Eigenschaften aus den 2089 Conjointstudien zugrunde gelegt. Die Häufigkeit des Vorkommens einer Eigenschaft entspricht also der Auswahlwahrscheinlichkeit dieser Eigenschaft für die Monte-Carlo-Simulation.

Aus Abbildung 11 geht hervor, dass die meisten Eigenschaften zwischen 2–7 Eigenschaftsausprägungen besitzen. Nachdem aus der Wahrscheinlichkeitsverteilung der Eigenschaften eine bestimmte Anzahl von Eigenschaften einer simulierten Conjointstudie erzeugt wurde, wird mit der aus den Häufigkeiten der Anzahl der Eigenschaftsausprägungen die Anzahl dieser über ihre Wahrscheinlichkeitsverteilung gezogen. Die höchste Anzahl von Eigenschaften, die eine Conjointstudie im untersuchten Datensatz hatte, war 51. Keine Conjointstudie besaß weniger als zwei Eigenschaften. Sei n_k die Anzahl der Conjointstudien mit k Eigenschaften. Dann sieht die empirische Wahrscheinlichkeitsverteilung für die Eigenschaften wie folgt aus:

$$P(K = k') = \frac{n_{k'}}{\sum_{k^*=2}^{51} n_{k^*}} \qquad k' = 2, ..., 51. \tag{5.1}$$

Die empirische Wahrscheinlichkeitsverteilung für die Eigenschaftsausprägungen ergibt sich ähnlich. Dabei sei n_L die Anzahl der Eigenschaftsausprägungen in den untersuchten Conjointstudien. Die höchste Anzahl von Eigenschaftsausprägungen war 47 und die geringste Anzahl war 2.

$$P(L_k = L) = \frac{n_L}{\sum_{L^*=2}^{47} n_{L^*}} \qquad \forall k = 1, ..., K; L = 2, ..., 47. \tag{5.2}$$

Die benötigte Anzahl von Konsumenten wird über eine lineare Regression bestimmt. Hierbei

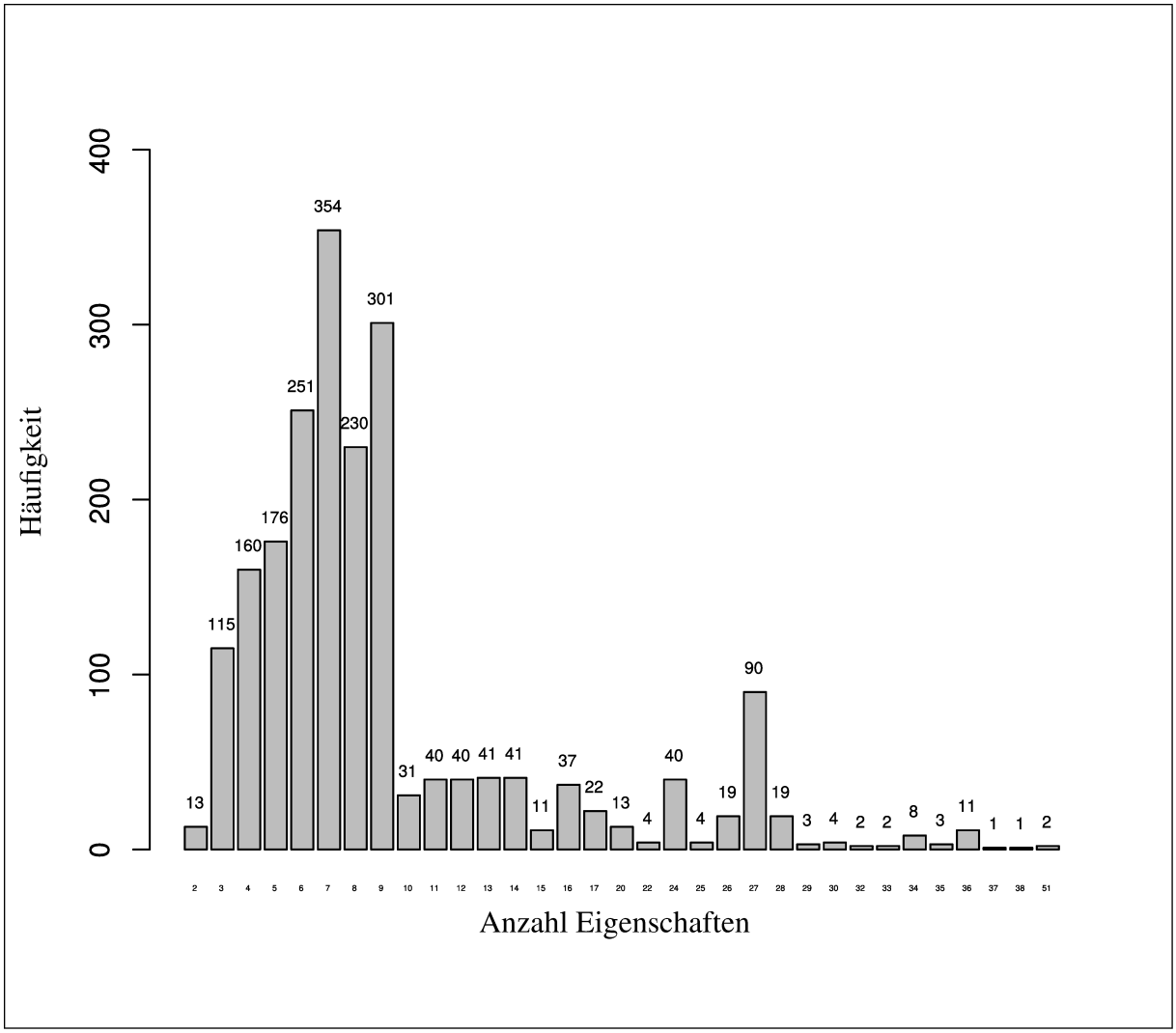

Abbildung 10: Balkendiagramm der Häufigkeit des Vorkommens der Eigenschaften in den untersuchten Conjointstudien. (Quelle: Eigene Darstellung)

ist die abhängige Variable die Anzahl der durchschnittlichen Konsumenten, die durch Bildung des arithmetischen Mittels über alle Konsumenten, die die gleiche Summe von Eigenschaftsausprägungen besitzen, beschrieben wird. In Abbildung 12 lässt sich der lineare Zusammenhang erkennen. Das lineare Modell hat ein korrigiertes Bestimmtheitsmaß von $R^2 = 0.129$, was auf eine recht große Streuung in der Anzahl der Konsumenten hindeutet. Die Regressionskoeffizienten für die Konstante und die abhängige Variable sind statistisch signifikant für $\alpha = 0.001$, womit angenommen werden kann, dass sie verschieden von 0 sind und einen steigenden Trend beschreiben. Somit wird die geschätzte Anzahl der Konsumenten $\hat{I}$ für die simulierten Conjointstudien über das folgende lineare Modell bestimmt, wobei die Nachkommastellen „abgeschnitten“ werden, sprich, es wird auf die nächste natürliche Zahl abgerundet, was durch die Abrundungsfunktion $\lfloor \cdot \rfloor$ angezeigt wird:

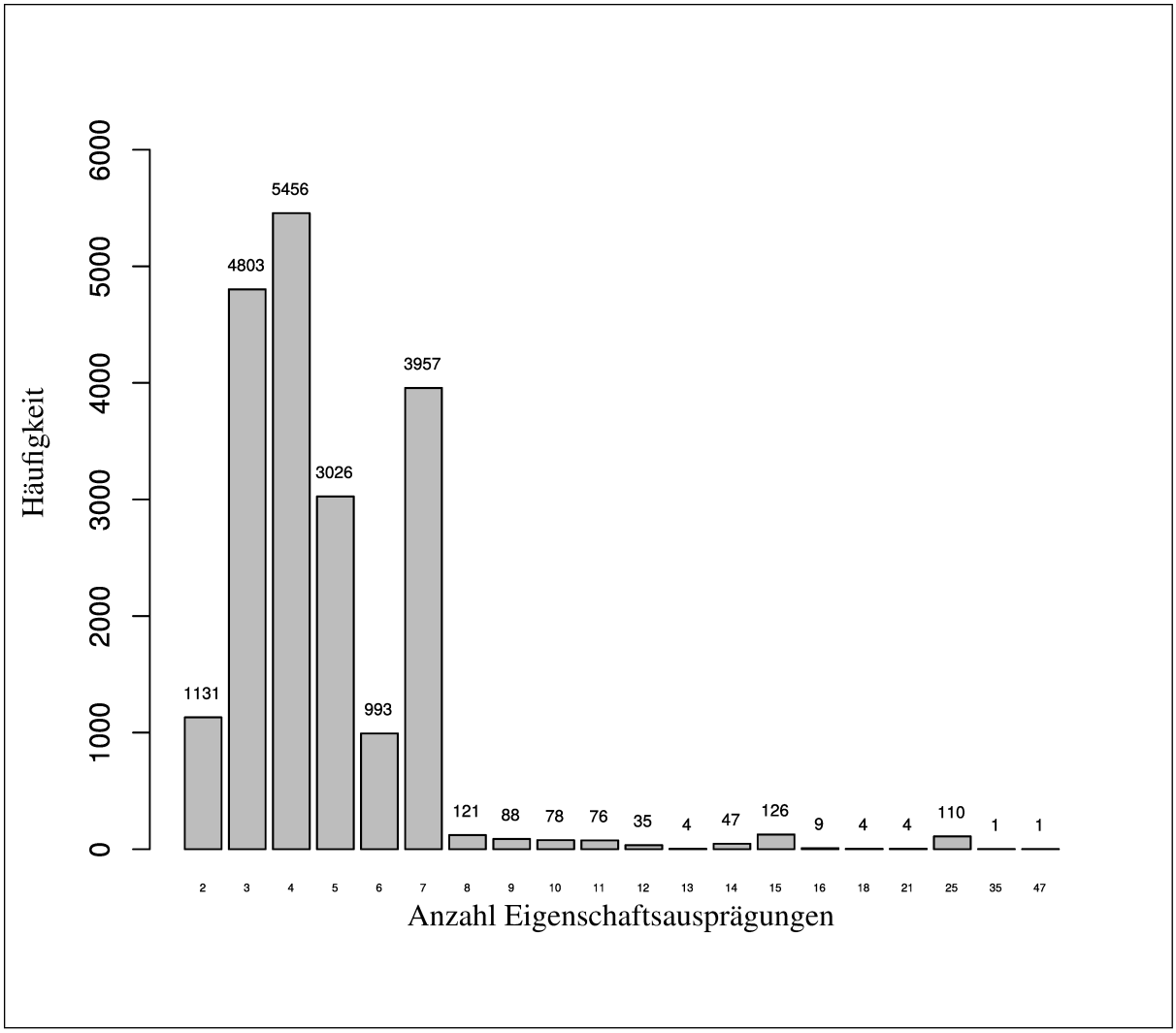

Abbildung 11: Balkendiagramm der Häufigkeit des Vorkommens der Eigenschaftsausprägungen in den untersuchten Conjointstudien. (Quelle: Eigene Darstellung)

$$\hat{I} = \lfloor 502{,}0188 + 2{,}3968 \cdot \sum_{k=1}^{K} L_k \rfloor \tag{5.3}$$

Die meisten Publikationen im Bereich der Produktlinienoptimierung, die eine Monte-Carlo-Simulation durchführten, nutzten symmetrische Eigenschaften. Symmetrisch bedeutet in diesem Zusammenhang, dass alle Eigenschaften einer simulierten Conjointstudie die gleiche Anzahl Eigenschaftsausprägungen haben. Aus den vorliegenden 2089 Conjointstudien geht jedoch hervor, dass diese symmetrischen Eigenschaften kaum existieren. Lediglich sechs Studien verwenden symmetrische Eigenschaften, weshalb in dieser Simulation auf diese Art simulierter Conjointstudien verzichtet wird.

Die vorliegenden Conjointstudien geben keine Auskunft über die Anzahl der Produkte pro Produktlinie, falls eine Produktlinienoptimierung durchgeführt wurde. In der Literatur werden hier-

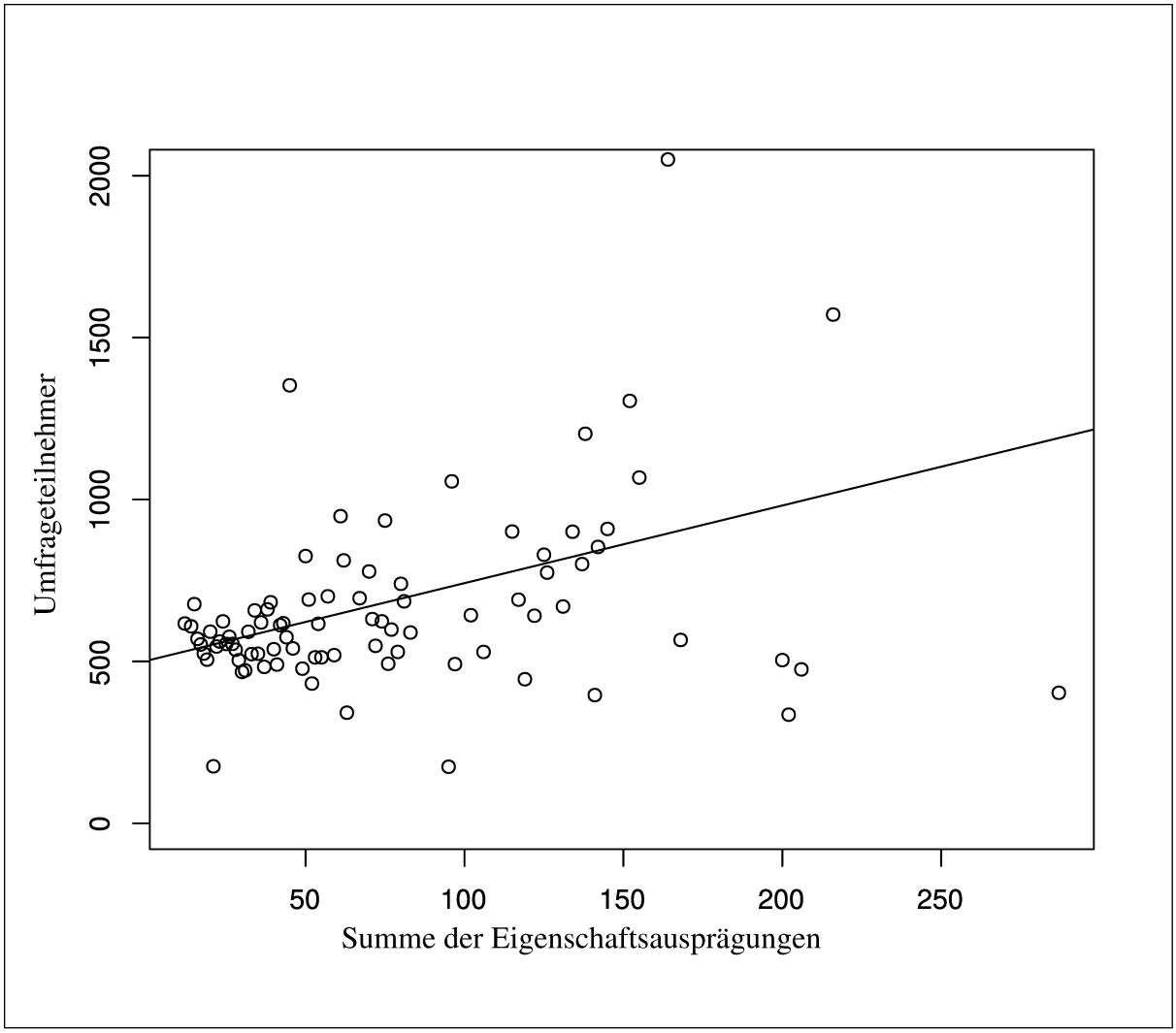

Abbildung 12: Regressionsgerade für die Vorhersage der Anzahl der Konsumenten in einer Umfrage in Abhängigkeit der Summe der Eigenschaftsausprägungen. (Quelle: Eigene Darstellung)

zu keine Angaben gemacht, weshalb wie viele Produkte in einer Produktlinie verwendet werden. Aus diesem Grund wird sich für die maximale Anzahl von Produkten in einer Produktlinie an der oberen Grenze der in der Literatur verwendeten Produkte pro Produktlinie orientiert, die bei neun Produkten pro Produktlinie von Tsafarakis et al. (2011) liegt. Siehe hierzu auch Tabelle 5. Des Weiteren werden ergänzend drei Produkte pro Produktlinie für die Simulationen verwendet. Zusammenfassend: $R = \{3, 9\}$.

Für die Erzeugung der simulierten Teilnutzenwerte wird sich an Baier (2014, S. 110) gehalten. Der beste Weg für eine realitätsnahe Bestimmung von individuellen Teilnutzenwerten wäre die empirische Verteilung aus den Hierarchischen Bayes-Schätzungen der vorliegenden Conjointstudien zu bestimmen. Die hierfür nötigen, aus den Schätzungen stammenden Teilnutzenwerte liegen jedoch nicht vor. In der Hierarchischen Bayes-Schätzung wird für die Priori-Verteilung der Eigenschaftsausprägungen über die Konsumenten von einer multivariaten Normalverteilung ausgegangen (Allenby 1998, S. 71), weshalb die Annahme für die Verteilung der individuellen

Teilnutzenwerte aus einer multivariaten Normalverteilung für die simulierten Conjointstudien legitim ist. Da die hierarchische Bayes-Schätzung das State-of-the-Art-Schätzverfahren für Choice-based Conjointanalysen ist, die eine Einsatzhäufigkeit von 94 % aufweisen (Selka/Baier 2014, S. 59), werden die aus den in den bisherigen Publikationen verwendeten Verteilungsannahmen (Gleich- oder Betaverteilung, siehe Tabelle 4) vernachlässigt. Für die individuellen Teilnutzenwerte werden zunächst die Mittelwerte für die Teilnutzenwerte durch eine Gleichverteilung mit Werten zwischen $[-1,1]$ bestimmt.

Eigenschaften	$P(K=k') = \frac{n_{k'}}{\sum_{k^*=2}^{51} n_{k^*}}$
Eigenschaftsausprägungen	$P(L_k = L) = \frac{n_L}{\sum_{L^*=2}^{47} n_{L^*}}$
Konsumenten	$\hat{I} = \lfloor 502,0188 + 2,3968 \cdot \sum_{k=1}^{K} L_k \rfloor$
Produkte pro Produktlinie	$\{3,9\}$
Teilnutzenwerte	$\beta_i \sim \text{MVN}\left(\overline{\beta}, \Sigma\right)$
Status-quo-Markt	$n_{SQP} = \left\lfloor \log_{10}\left[\binom{\prod_{k=1}^{K} L_k}{R}\right]\right\rfloor$

Tabelle 22: Zusammenfassung der Parameter für die Monte-Carlo-Simulation. (Quelle: Eigene Darstellung)

Um eine relativ hohe Heterogenität innerhalb der Konsumenten zu erzeugen, wird die Standardabweichung für die verwendeten Mittelwerte der Teilnutzenwerte auf $\sigma = 0.5$ bzw. $\sigma^2 = 0.25$ gesetzt. Als Modellformulierung für einen simulierten Konsumenten werden die individuellen Teilnutzenwerte, wie folgt, erzeugt:

$$\beta_i = \begin{pmatrix} \beta_{i11} \\ \vdots \\ \beta_{iKL_K} \end{pmatrix}_{\sum_{k=1}^{K} L_k} \sim \text{MVN}\left(\overline{\beta}, \Sigma\right) \quad \text{mit}$$

$$\overline{\beta} = \begin{pmatrix} \text{Gl}[-1,1] \\ \vdots \\ \text{Gl}[-1,1] \end{pmatrix}_{\sum_{k=1}^{K} L_k} \quad \text{und}$$

$$\Sigma = \begin{pmatrix} 0.25 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 0.25 \end{pmatrix}_{(\sum_{k=1}^{K} L_k) \times (\sum_{k=1}^{K} L_k)}$$

Der Status-quo-Markt wird für die Simulationen zufällig erzeugt. Dabei wird die Anzahl der Status-quo-Produkte mit der Anzahl der zulässigen Lösungen vergrößert, damit sichergestellt wird, dass die Status-quo-Produkte wenigstens von einigen Konsumenten ggü. den Produkten in der Produktlinie bevorzugt werden. Damit werden bspw. Marktanteile von 100 % verhindert, aber vor allem auch Optimierungsprobleme vermieden, gerade für die deterministischen Modelle, die eine hohe Anzahl von optimalen Lösungen haben. Deshalb wird die Anzahl der Status-quo-Produkte adaptiv für jede Problemgröße angepasst, indem der Exponent zur Basis 10 für die Anzahl der zulässigen Lösungen des zugrunde liegenden Optimierungsproblems verwendet wird. Sei n_{SQP} die Anzahl der Status-quo-Produkte:

$$n_{SQP} = \left\lfloor \log_{10} \left[\binom{\prod_{k=1}^{K} L_k}{R} \right] \right\rfloor \tag{5.4}$$

Somit wurden fast alle für die simulierten Conjointdaten und die Produktlinien relevanten Parameter aus einem repräsentativen Datensatz aus realen Conjointstudien hergeleitet. Lediglich die Anzahl der Produkte pro Produktlinie konnte daraus nicht bestimmt werden, weshalb sich an den Empfehlungen aus der Literatur orientiert wird.

5.2 Simulationsparameter

Im Folgenden werden die für die Monte-Carlo-Simulation (MCS) relevanten Parameter definiert und diejenigen Objekte aufgeführt, die den Umfang der MCS determinieren. Zu diesen Parametern und Objekten gehören:

- Die vier Zielfunktionen,
- die Anzahl der simulierten Conjointdatensätze für jede Zielfunktion,
- die untersuchte Anzahl der Produkte in einer Produktlinie und
- die Anzahl der Wiederholungen, die ein bestimmtes Produktlinienoptimierungsproblem von jedem Verfahren gelöst wird.

Für die Durchführung der MCS werden für jede der vier Zielfunktionen drei Blöcke gebildet. Die Einteilung der simulierten Conjointdatensätze in die einzelnen Blöcke geschieht über die Anzahl der zulässigen Produktlinien, die sich über die Anzahl der Eigenschaften, die Anzahl der Eigenschaftsausprägungen und der Anzahl der Produkte pro Produktlinie zusammensetzen.

Die Anzahl der Wiederholungen, die ein bestimmtes Produktlinienoptimierungsproblem von jedem Verfahren gelöst wird, wird auf, wie in der Literatur üblich, 10 gesetzt, um die Ergebnisse der Optimierungen zu stabilisieren. Würde jedes Verfahren jedes Problem lediglich einmal lösen, könnten keine aussagekräftigen Feststellungen über deren Performance getätigt werden, da aufgrund der stochastischen Natur der Verfahren Ausreißer für die statistische Analyse einen geringeren Einfluss auf das Gesamtergebnis ausüben können. Die Einteilung in drei Blöcke mit der entsprechenden Anzahl an möglichen Lösungen hat den Hintergrund, dass sich in früheren Simulationsrechnungen herausgestellt hat, dass sich die Performance der Algorithmen bis zu einer Anzahl möglicher Produktlinien bis 10^{40} kaum unterscheidet (Voekler/Baier 2020). Aus diesem Grund werden diese Problemgrößen in diesem Zusammenhang als „klein“ definiert. Sie werden dennoch der Vollständigkeit halber in die Betrachtung integriert, um einen repräsentativen Datensatz zu erhalten. „Mittlere“ Produktlinienoptimierungsprobleme liegen zwischen 10^{41} bis einschließlich 10^{80} möglichen Lösungen. Ab 10^{81} möglichen Produktlinien wird in dieser Arbeit von „großen“ Produktlinienoptimierungsproblemen gesprochen. Die Einteilung in diese Blöcke ermöglicht eine gleichmäßigere Verteilung der Datensätze auf verschiedene Problemgrößen. Würde man die Datensätze, wie in Abschnitt 5.1, mit einer ähnlichen Verteilung der Problemgrößen erzeugen, würden kleinere Probleme häufiger und größere Probleme deutlich seltener vorkommen, womit die Stichprobe zugunsten kleinerer Problemgrößen verzerrt wäre. In der Endbetrachtung sollen diese Datensätze jedoch relativ homogen über die Anzahl der möglichen Produktlinien verteilt sein. Ab mittleren Problemgrößen zeigen sich verstärkt Unterschiede in der Performance der Verfahren. Für die Auswertung werden die drei Blöcke gemeinsam betrachtet. Ziel dieser Arbeit ist es u.a., ein Verfahren zu identifizieren, dass unabhängig von der Problemgröße eine gute Performance zeigt und nicht notwendigerweise lediglich in einem der Blöcke. Insofern ist die gemeinsame Betrachtung über die drei Blöcke hinweg zielführend und führt zu einer besseren Verallgemeinerbarkeit. Die Blockgrößen werden definiert als:

- **Block 1:** kleine Produktlinienoptimierungsprobleme mit weniger als 10^{41} zulässigen Produktlinien,

- **Block 2:** mittlere Produktlinienoptimierungsprobleme mit gleich vielen oder mehr als 10^{41} oder weniger als 10^{81} zulässigen Produktlinien,

- **Block 3:** große Produktlinienoptimierungsprobleme mit $\geq 10^{81}$ zulässigen Produktlinien.

In Tabelle 23 wird die Blockeinteilung und der Aufbau der MCS tabellarisch dargestellt. Für jeden Block werden unter Berücksichtigung der Anzahl von Produkten pro Produktlinie und der Anzahl der Eigenschaften und Eigenschaftsausprägungen, stammend aus den Verteilungen von 5.1 und 5.2, 20 simulierte Conjointdatensätze erzeugt. Anschließend werden für die simulierten Conjointdatensätze die Anzahl der Konsumenten über 5.3 sowie die Anzahl der Status-quo-

Produkte über 5.4 bestimmt. An dieser Stelle beginnt die eigentliche Produktlinienoptimierung, indem jeder Datensatz von jedem Verfahren zehnmal gelöst wird.

∀ **Zielfunktionen**	**Block 1**		**Block 2**		**Block 3**	
Anzahl zulässiger Produktlinien	$< 10^{41}$		$[10^{41}; 10^{81})$		$\geq 10^{81}$	
Produkte pro Produktlinie	3	9	3	9	3	9
Anzahl simulierte CA-Datensätze	10	10	10	10	10	10
Konsumenten	siehe Bildungsvorschrift 5.3					
Wiederholungen pro CA-Datensatz	10		10		10	
Datensätze inkl. Wiederholungen	100	100	100	100	100	100
Datensätze pro Block	200		200		200	
Datensätze für jede Zielfunktion	600					

Tabelle 23: Bildung der Datengrundlage für die Monte-Carlo-Simulation für die Produktlinienoptimierung pro Zielfunktion. (Quelle: Eigene Darstellung)

Daraus resultieren für jeden Block 200 (20×10) Datensätze und somit 600 (3×200) Datensätze für jede Zielfunktion. Zusammengenommen über die drei Blöcke und die vier Zielfunktionen ergeben sich hieraus 2400 (4×600) Optimierungsdurchläufe für jedes der sechs Verfahren.

Die erzeugten Datensätze besitzen $6.9552 \cdot 10^5$ bis $7.328679 \cdot 10^{200}$ mögliche Produktlinien. Eine detaillierte Auflistung dieser Datensätze, geordnet nach der Anzahl der möglichen Produktlinien, befindet sich im Anhang (siehe S. XII).

Für die folgende Simulationsstudie sind alle verwendeten Parameter der Verfahren in Tabelle 24 nochmals aufgeführt. Dabei wurden die Abbruchbedingungen zum Teil angepasst. Die Laufzeiten wurden für jeden der drei Blöcke vorgegeben. Je größer die Anzahl der möglichen Produktlinien, desto mehr Laufzeit bekommen die Verfahren ($T = \{300s, 450s, 600s\}$), um eine vorzeitige Konvergenz zu vermeiden. Für MMAS und MMAS_ls wurde die Anzahl der Iterationen, in denen sich der maximale Zielfunktionswert nicht verbessert hat, von 500 auf 5000 hochgesetzt, um die vorgegebenen Laufzeiten für die einzelnen Blöcke besser auszureizen. Die lokale Suche im MMAS_ls entspricht dem One-Opt-Verfahren aus Abschnitt 4.5.2.2. Simulated Annealing hingegen wurde die Anzahl der Iterationen, die nicht zu einem besseren Zielfunktionswert führen von 150000 auf 50000 heruntergesetzt, da sich zeigte, dass die Zielfunktionswerte trotz dieser Reduzierung nicht in ihrer Höhe abnehmen. Der verwendete GA entspricht dem GA04 aus Abschnitt 4.3.2. GA in Kombination mit dem Clusteransatz entspricht dem aus Abschnitt 4.3.3. Für die PSO wurde sich an den Parameterwerten aus Tsafarakis et al. (2011, S. 22) mit den beschriebenen Änderungen in Abschnitt 4.4 orientiert. Dabei wurde die Populationsgröße durch Beibehalten des Verhältnisses von der Nachbarschaftsgröße und Anzahl der Partikel je Nachbarschaft auf 540 vergrößert.

Somit können die Verfahren, die durch vorhergehende Simulationen ermittelt wurden, gegen-

Verfahren	Parameter	Abbruchbed.
GA	Pop.gr. = 2000, parents_rate = 0.5, #Nachkommen = 1000, Pop.wahrm. = Emigration, Selektion = Truncation, Crossover = 1-Point, Crossoverwahrsch. = 1.0, Mutationsrate = $\frac{1}{\#\text{Eigenschaften}\cdot\#\text{Produktliniengröße}}$	keine Verbesserung des ZF in 50 Generationen oder $T = \{300s, 450s, 600s\}$
GAC	wie **GA** + $J_{\text{best}} = 50$ erzeugte Produkte pro Cluster (siehe Abschnitt 4.3.3 für Details)	wie **GA**
PSO	Pop.gr. = 540, Nachbarschaftsgr. = 30, Partikel = 18, $w_{min} = 0.1$, $w_{max} = 0.9$, $c_1 = c_2 = 2.05$, $\chi = 0.728$	keine Verbesserung des ZF in 500 Generationen oder $T = \{300s, 450s, 600s\}$
MMAS	#Ameisen $= \sum_{k=1}^{K} L_k$, $\tau_{kl}(1) = \tau_{max}(0)\ \forall k,l$, $\rho = 0.98$, $\tau_{kl}(t+1) = \rho\,\tau_{kl}(t) + Z^{max}(t)\ \ \forall t$, $\tau_{max}(t) = \frac{1}{1-\rho} \cdot Z^{max}(t)\ \forall t$, $\tau_{min}(t) = \frac{\tau_{max}(t)}{2\cdot\#\text{Ameisen}}$	$Z^{max}(t) = Z^{max}(t - 5000)$ oder $T = \{300s, 450s, 600s\}$
MMAS_ls	wie **MMAS**, außer $\rho = 0.80$ + lokale Suche nach jeder Iteration (siehe Abschnitt 4.5.2.2 für Details)	wie **MMAS**
SA	$c_{m+1} = \frac{c_m}{1+\frac{c_m\cdot\ln(1+\delta)}{3\sigma c_m}}$, $c_0 = \frac{\overline{\Delta Z}^{(+)}}{\ln(\frac{s_2}{s_2\cdot\chi_0 - s_1(1-\chi_0)})}$, $c_{end} =$ adaptiv, $\delta = 0.1$, $\chi_0 = 0.95$, Anzahl Evaluationen pro Temperatur = $\sum_{k=1}^{K} L_k$	$\nexists Z^{max}(t, t+1, ..., t+50000) \geq Z^{max}$ oder $T = \{300s, 450s, 600s\}$

Tabelle 24: Zusammenfassung und Übersicht der verwendeten Parameter der einzelnen Verfahren. In den Abbruchbedingungen bedeutet $T = \{300s, 450s, 600s\}$, dass für Block 1, Block 2 und Block 3 eine maximale Laufzeit von $300s$, $450s$ und $600s$ definiert wurde. (Quelle: Eigene Darstellung)

einander unter den gegebenen Voraussetzungen im nächsten Abschnitt gebenchmarkt werden.

5.3 Ergebnisse der Simulationsstudie

In diesem Abschnitt werden die Ergebnisse der Monte-Carlo-Simulation vorgestellt. In Tabelle 25 findet sich die Ergebnisübersicht für alle Verfahren und Zielfunktionen. Zusätzlich sind die Standardabweichungen sowie die Laufzeit der Verfahren bzgl. der Zielfunktionen dargestellt. Es wird nochmals darauf hingewiesen, dass es sich bei den gemittelten Zielfunktionswerten um prozentuale Angaben handelt. Dabei wurde für jeden Conjointdatensatz der maximale Zielfunktionswert unter den zehn Optimierungsdurchläufen über alle Verfahren bestimmt und sodann alle Zielfunktionswerte aller Verfahren des gleichen Conjointdatensatzes durch den maximalen Zielfunktionswert geteilt. Eine Ausnahme besteht in der probabilistischen Marktanteilsmaxi-

mierung M_{prob}, da hierfür das globale Maximum über den exakten Algorithmus aus Abschnitt 4.2.2 bestimmt werden konnte.

Verfahren	**ZF**	$\overline{\mathbf{ZF}}_{max}$	$\overline{\mathbf{ZF}}_{max}^{Ent}$	$\overline{\mathbf{ZF}}_{max}^{mean}$	$\overline{\mathbf{ZF}}_{mean}$	$\overline{\mathbf{ZF}}_{mean}^{Ent}$	$\overline{\mathbf{ZF}}_{mean}^{mean}$	$\overline{\sigma}_{\mathbf{ZF}}$	ØT in s
GA	G_{det}	0.94053	0.94672	0.97279	0.91147	0.91856	0.95850	0.01889	388.65
	M_{det}	0.95290			0.92565			0.01926	375.33
	G_{prob}	**0.99802**	0.99885		0.99731	0.99844		0.00049	376.90
	M_{prob}	**0.99968**			0.99957			0.00006	402.19
GAC	G_{det}	0.97268	0.98042	0.98912	0.94836	0.95775	0.97734	0.01639	401.76
	M_{det}	**0.98817**			**0.96714**			0.01376	390.95
	G_{prob}	0.99570	0.99781		0.99395	0.99692		0.00122	392.04
	M_{prob}	**0.99992**			0.99989			0.00002	411.85
PSO	G_{det}	0.58429	0.59971	0.76776	0.52999	0.54682	0.73403	0.03174	255.06
	M_{det}	0.61513			0.56366			0.03028	264.95
	G_{prob}	0.89301	0.93580		0.86852	0.92122		0.01612	278.19
	M_{prob}	0.97860			0.97393			0.00320	281.55
MMAS	G_{det}	0.92841	0.95245	0.97622	0.88142	0.91262	0.95625	0.03112	293.61
	M_{det}	0.97649			0.94381			0.02270	303.98
	G_{prob}	**0.99997**	**0.99998**		**0.99978**	**0.99988**		0.00021	298.45
	M_{prob}	**0.99999**			**0.99997**			0.00002	321.15
MMAS_ls	G_{det}	**0.99799**	**0.99523**	**0.99756**	**0.97082**	**0.96492**	**0.98233**	0.01971	412.46
	M_{det}	**0.99247**			0.95902			0.02325	415.51
	G_{prob}	0.99984	0.99988		0.99956	0.99972		0.00018	419.61
	M_{prob}	0.99992			0.99988			0.00003	424.07
SA	G_{det}	0.95665	0.96322	0.98144	0.91915	0.92630	0.96277	0.02828	286.05
	M_{det}	0.96978			0.93344			0.02488	292.75
	G_{prob}	**0.99940**	0.99966		0.99874	0.99924		0.00055	276.70
	M_{prob}	0.99991			0.99973			0.00011	311.28

Tabelle 25: Ergebnisse der Monte-Carlo-Simulation für alle verwendeten Verfahren. Die fett markierten Werte stellen die höchsten gemessenen Mittelwerte dar, die sich aus einem paarweisen T-Test zwischen der Verfahren ergeben. Ein Unterschied in den Mittelwerten ist signifikant, wenn die Nullhypothese auf dem 5 %-Signifikanzniveau abgelehnt werden kann. Legende: **ZF**... Zielfunktion; $\overline{\mathbf{ZF}}_{max}$... arithmetische Mittelwerte der Anteile an der besten gefundenen Lösung unter 10 Wiederholungen im Verhältnis zur besten gefundenen Lösung einer Probleminstanz aus allen Verfahren für jede Zielfunktion; $\overline{\mathbf{ZF}}_{max}^{mean}$ ... arithmetisches Mittel aus $\overline{\mathbf{ZF}}_{max}$ über alle Zielfunktionen; $\overline{\mathbf{ZF}}_{mean}$... arithmetische Mittelwerte über den Mittelwert der 10 Wiederholungen einer Probleminstanz als Anteil an der besten gefunden Lösung über alle Verfahren für jede Zielfunktion; $\overline{\mathbf{ZF}}_{max}^{Ent}$... Mittelwerte über $\overline{\mathbf{ZF}}_{max}$ für die deterministischen und probabilistischen Auswahlregeln; $\overline{\mathbf{ZF}}_{mean}^{Ent}$... Mittelwerte über $\overline{\mathbf{ZF}}_{mean}$ für die deterministischen und probabilistischen Auswahlregeln; $\overline{\mathbf{ZF}}_{mean}^{mean}$... arithmetisches Mittel von $\overline{\mathbf{ZF}}_{mean}$ aller Zielfunktionen; $\overline{\sigma}_{\mathbf{ZF}}$... arithmetische Mittelwerte der Standardabweichungen für jede Zielfunktion (berechnet aus den als Anteile an der besten Lösung transformierten Zielfunktionswerten); ØT... arithmetisches Mittel der benötigten Zeit über alle Probleminstanzen für jede Zielfunktion. (Quelle: Eigene Darstellung)

Die Spalte $\overline{\mathbf{ZF}}_{max}$ repräsentiert die (arithmetisch) gemittelten besten Zielfunktionswerte je zehn

Optimierungsdurchläufen pro Conjointdatensatz jedes Verfahrens für jede Zielfunktion mit einer Stichprobengröße von $n = 60$ Conjointdatensätzen. Diese $4 \times 60 = 240$ Zielfunktionswerte werden gemittelt und in der Spalte $\overline{\mathbf{ZF}}_{max}^{mean}$ zusammengefasst. Für die Spalte $\overline{\mathbf{ZF}}_{max}^{Ent}$ mit einer Stichprobengröße von $n = 120$ (60 Conjointdatensätzen $\times$ 2 Zielfunktionen mit deterministischen/probabilistischen Entscheidungsregeln) und $\overline{\mathbf{ZF}}_{mean}^{Ent}$ mit einer Stichprobengröße von $n = 1200$ (60 Conjointdatensätzen $\times$ 10 Optimierungsdurchläufe $\times$ 2 Zielfunktionen mit deterministischen/probabilistischen Entscheidungsregeln) werden die transformierten Zielfunktionswerte bzgl. ihrer Entscheidungsregeln verwendet. Die Spalte $\overline{\mathbf{ZF}}_{mean}$ mittelt für jede Zielfunktion $n = 600$ Zielfunktionswerte, die sich aus den 60 Conjointdatensätzen multipliziert mit je zehn Optimierungsdurchläufen ergeben. Ergo, besteht die Spalte $\overline{\mathbf{ZF}}_{mean}^{mean}$ aus $n = 2400$ gemittelten Zielfunktionswerten für jedes Verfahren (60 Conjointdatensätze $\times$ 10 Optimierungsdurchläufe $\times$ 4 Zielfunktionen). Für die Darstellung der gemittelten Standardabweichung $\overline{\sigma}_{\mathbf{ZF}}$ werden die transformierten Zielfunktionswerte verwendet. Insgesamt gibt es $n = 60$ Standardabweichungen für jede Zielfunktion, die gemittelt werden. In der letzten Spalte **ØT** werden die Laufzeiten der Verfahren für alle Optimierungsdurchläufe gemittelt ($n = 600$ für jede Zielfunktion).

	Rangfolge der Verfahren bezogen auf ...													
	$\overline{\mathbf{ZF}}_{...}^{mean}$		$\overline{\mathbf{ZF}}_{...}^{Ent}$				$\overline{\mathbf{ZF}}_{...}$							
	∀ Zielfunkt.		det.		probab.		G_{det}		M_{det}		G_{prob}		M_{prob}	
	max	*mean*	*max*	*mean*	*max*	*mean*	*max*	*mean*	*max*	*mean*	*max*	*mean*	*max*	*mean*
GA	4.	4.	4.	4.	2.	4.	3.	3.	4.	4.	1./2.	4.	1./2.	5.
GAC	2.	2.	2.	2.	5.	5.	2.	2.	1.	1.	5.	5.	1./2.	2.
PSO	6.	6.	6.	6.	6.	6.	6.	6.	6.	6.	6.	6.	6.	6.
MMAS	4.	4.	4.	5.	1.	1.	3.	5.	3.	3.	1.	1.	1.	1.
MMAS_ls	1.	1.	1.	1.	2.	2.	1.	1.	1.	2.	2.	2.	2.	2.
SA	3.	3.	3.	3.	1./2.	3.	3.	3.	3./4.	4.	1./2.	3.	2.	4.

Tabelle 26: Rangfolge der Verfahren auf aggregierter (*max*: $n = 240$, *mean*: $n = 2400$), Entscheidungsregel- (*max*: $n = 120$, *mean*: $n = 1200$) und Zielfunktionsebene (*max*: $n = 60$, *mean*: $n = 600$) basierend auf paarweisen T-Tests für verbundene Stichproben mit 5 %-igem Signifikanzniveau. Allgemeine Nullhypothese: das arithmetische Mittel $\overline{\mathbf{ZF}}_{...}^{mean}$ bzw. $\overline{\mathbf{ZF}}_{...}^{Ent}$ bzw. $\overline{\mathbf{ZF}}_{...}$ des einen Verfahrens ist gleich dem entsprechenden arithmetischen Mittel des anderen Verfahrens. Anmerkung: Die Rangfolge ist aufgrund der T-Tests nicht notwendigerweise transitiv, vor allem dann nicht, wenn die Stichprobengrößen verhältnismäßig klein sind. Deshalb kann ein Rang in diesem Fall doppelt vergeben werden, da sich ein erstes Verfahren von einem zweiten nicht statistisch signifikant unterscheiden kann, von einem dritten aber schon, das sich allerdings nicht vom zweiten statistisch signifikant unterscheidet. (Quelle: Eigene Darstellung)

Die fett markierten Werte in Tabelle 25 zeigen für die jeweiligen Spalten die höchsten gemittelten Zielfunktionswerte auf. Hierfür wurden paarweise T-Tests für verbundene Stichproben für die untersuchten transformierten Zielfunktionsmittelwerte durchgeführt. Ein Unterschied zwischen zwei Verfahren wird festgestellt, falls der p-Wert unter einem Signifikanzniveau von 5 % liegt. Existieren für zwei Zielfunktionsmittelwerte zwei oder mehr Verfahren, die fett markiert sind, bedeutet dies, dass für diese Zielfunktionsmittelwerte keine statistisch signifikanten Unterschiede zwischen den besten Verfahren festgestellt werden konnten. Siehe bspw. $\overline{\mathbf{ZF}}_{max}$ für die probabilistische Gewinnmaximierung G_{prob}, für die zwischen GA, MMAS und SA kein

statistisch signifikanter Unterschied auf dem 5 %-Signifikanzniveau besteht.

Eine weiterführende Darstellung für die Rangfolge der Verfahren bietet die Tabelle 26. Sie basiert auf den paarweisen Vergleichen eines zweiseitigen T-Tests für verbundene Stichproben auf dem 5 %-Signifikanzniveau für die Auswertung auf aggregierter ($\overline{\mathbf{ZF}}_{...}^{mean}$), Entscheidungsregel- ($\overline{\mathbf{ZF}}_{...}^{Ent}$) und Zielfunktionsebene ($\overline{\mathbf{ZF}}_{...}$). Auf die entsprechenden Rangfolgen wird in den folgenden Abschnitten genauer eingegangen.

5.3.1 Auswertung auf aggregierter Ebene

In Tabelle 27 sind die paarweisen Vergleiche über Vergleichsrelationen mittels p-Wert eines zweiseitigen T-Tests für verbundene Stichproben für die aufgeführten Verfahren bezogen auf die maximalen und gemittelten transformierten Zielfunktionswerte für alle verwendeten Zielfunktionen dargestellt. Diese Tabelle korrespondiert mit den Einträgen in Tabelle 26 für die Spalten $\overline{\mathbf{ZF}}_{max}^{mean}$ und $\overline{\mathbf{ZF}}_{mean}^{mean}$.

		Vergleichsrelationen für $\overline{\mathbf{ZF}}_{max}^{mean}$				
	GA	**GAC**	**PSO**	**MMAS**	**MMAS_ls**	**SA**
GA		$<$	$>$	$=$	$<$	$<$
GAC	$>$		$>$	$>$	$<$	$>$
PSO	$<$	$<$		$<$	$<$	$<$
MMAS	$=$	$<$	$>$		$<$	$<$
MMAS_ls	$>$	$>$	$>$	$>$		$>$
SA	$>$	$<$	$>$	$>$	$<$	
	Vergleichsrelationen für $\overline{\mathbf{ZF}}_{mean}^{mean}$					

Tabelle 27: Übersicht der paarweisen Vergleiche mittels p-Wert ('=' ≥ 0.05, '$<,>$' < 0.05) eines zweiseitigen T-Tests für verbundene Stichproben für die aufgeführten Verfahren bezogen auf alle transformierten Zielfunktionswerte (untere Dreiecksmatrix; bezogen auf $\overline{\mathbf{ZF}}_{mean}^{mean}$) sowie die transformierten maximalen Zielfunktionswerte (obere Dreiecksmatrix; bezogen auf $\overline{\mathbf{ZF}}_{max}^{mean}$) für alle verwendeten Zielfunktionen. Allgemeine Nullhypothese: das arithmetische Mittel $\overline{\mathbf{ZF}}_{mean}^{mean}$ bzw. $\overline{\mathbf{ZF}}_{max}^{mean}$ des einen Verfahrens ist gleich dem entsprechenden arithmetischen Mittel des anderen Verfahrens. Stichprobengröße für $\overline{\mathbf{ZF}}_{mean}^{mean}$: $n = 2400$ Zielfunktionswerte (60 Conjointdatensätze $\times$ 4 Zielfunktionen $\times$ 10 Optimierungsdurchläufe). Stichprobengröße für $\overline{\mathbf{ZF}}_{max}^{mean}$: $n = 240$ Zielfunktionswerte (60 Conjointdatensätze $\times$ 4 Zielfunktionen). **Lesart:** Von links (Zeilenbezeichnung) nach rechts (Spaltenbezeichnung). **Lesebeispiel:** Der Mittelwert $\overline{\mathbf{ZF}}_{mean}^{mean}$ des GAC ist größer als der des GA, aber kleiner als der des MMAS_ls. Der Mittelwert $\overline{\mathbf{ZF}}_{max}^{mean}$ des GA ist kleiner als der des GAC, aber größer als der des PSO. (Quelle: Eigene Darstellung)

Für die Berechnung der p-Werte wurden die transformierten Zielfunktionswerte und nicht die originalen Zielfunktionswerte verwendet. Der Grund hierfür liegt darin, dass die Daten aus derselben Grundgesamtheit stammen müssen, um für die T-Tests nutzbar zu sein. Dies wird

durch die Transformation erreicht (siehe Abschnitt 4.3.2 für Details). Würde man die originalen Zielfunktionswerte aus der Gewinn- und Marktanteilsmaximierung gemeinsam in einem T-Test auswerten, könnten unerwünschte Effekte auftreten, da die Marktanteile zwischen $[0;1]$, die Gewinne jedoch in den Tausendern liegen. Somit entstünde eine Gewichtung zugunsten der Gewinnmaximierungsmodelle, die nicht gewollt ist, da sie die Ergebnisse bzw. die Mittelwerte verzerren kann. Insofern wird durch die Transformation der originalen Zielfunktionswerte ein Ausgleich in der Gewichtung zwischen den verschiedenen Modellen erreicht, wodurch eine gemeinsame Betrachtung ermöglicht wird.

Wie die Tabelle 27 zeigt, unterscheidet sich die Performance der Verfahren unter den gegebenen Voraussetzungen zumeist signifikant auf dem 5 %-Signifikanzniveau. Da zweiseitige T-Tests durchgeführt wurden, muss zur Entscheidungsfindung Tabelle 25 herangezogen werden. Ist $\overline{\mathbf{ZF}}_{max}^{mean}$ oder $\overline{\mathbf{ZF}}_{mean}^{mean}$ eines Verfahrens kleiner oder größer als die zugehörigen Zielfunktionsmittelwerte eines anderen Verfahrens und der zugehörige p-Wert kleiner als 0.05, so existiert ein signifikanter Unterschied in der Performance zwischen den Verfahren unter den gegebenen Voraussetzungen. Für die gemittelten, besten Zielfunktionswerte $\overline{\mathbf{ZF}}_{max}^{mean}$ und die gemittelten Zielfunktionswerte aller Optimierungsdurchläufe $\overline{\mathbf{ZF}}_{mean}^{mean}$ lässt sich die Nullhypothese, dass die Mittelwerte übereinstimmen lediglich für den GA vs. MMAS ($p-\text{Wert}(\text{GA vs. MMAS}; \overline{\mathbf{ZF}}_{max}^{mean}) = 0.35$ und $p-\text{Wert}(\text{GA vs. MMAS}; \overline{\mathbf{ZF}}_{mean}^{mean}) = 0.09$) nicht ablehnen, womit angenommen wird, dass sich die Performance der beiden Verfahren nicht signifikant unterscheidet. Für alle anderen Verfahrensvergleiche existieren signifikante Unterschiede sowohl für $\overline{\mathbf{ZF}}_{max}^{mean}$ als auch für $\overline{\mathbf{ZF}}_{mean}^{mean}$. Somit lässt sich für diese beiden Mittelwerte die Rangfolge der Verfahren für Tabelle 26 aus Tabelle 27 unter den gegebenen Voraussetzungen ableiten. Es lässt sich erkennen, dass MMAS_ls über alle, sowie die maximalen Zielfunktionswerte hinweg die beste Performance zeigt. Das zweitbeste Verfahren in diesem Zusammenhang ist GAC, gefolgt von SA. GA und MMAS unterscheiden sich statistisch nicht signifikant und belegen somit den vierten Platz. Das Verfahren mit dem schlechtesten Ergebnis ist PSO.

Hervorzuheben ist die Konsistenz dieser Reihenfolge bezogen auf $\overline{\mathbf{ZF}}_{max}^{mean}$ und $\overline{\mathbf{ZF}}_{mean}^{mean}$. Sowohl in den gemittelten, maximalen als auch in den gemittelten Zielfunktionswerten aller Optimierungsdurchläufe ist die Reihenfolge gleich. Das bedeutet, dass es kein Verfahren in dieser Untersuchung gibt, das große Ausreißer zu sehr guten oder schlechten Zielfunktionen im Verhältnis zur durchschnittlichen Performance hat. Dies kann als Indiz für eine hohe Stabilität beim Finden von approximativen Optima gedeutet werden. Stärker wird die Evidenz dieser Beobachtung, wenn die gemittelten Standardabweichungen $\overline{\sigma}_{\mathbf{ZF}}$ für die einzelnen Zielfunktionen hinzugezogen werden. Diese beziehen sich auf die transformierten Zielfunktionswerte und bewegen sich für alle Verfahren zwischen gerundeten 0.00002 und maximal 0.03174, was sehr geringe durchschnittliche Abweichungen von den Mittelwerten darstellt. Die lokale Suche mittels One-Opt-Verfahren MMAS_ls verbessert die Performance des MMAS auch für die hier verwendeten Probleminstanzen. Die Gründe hierfür sind zweierlei: One-Opt verbessert die lokale Suche nach

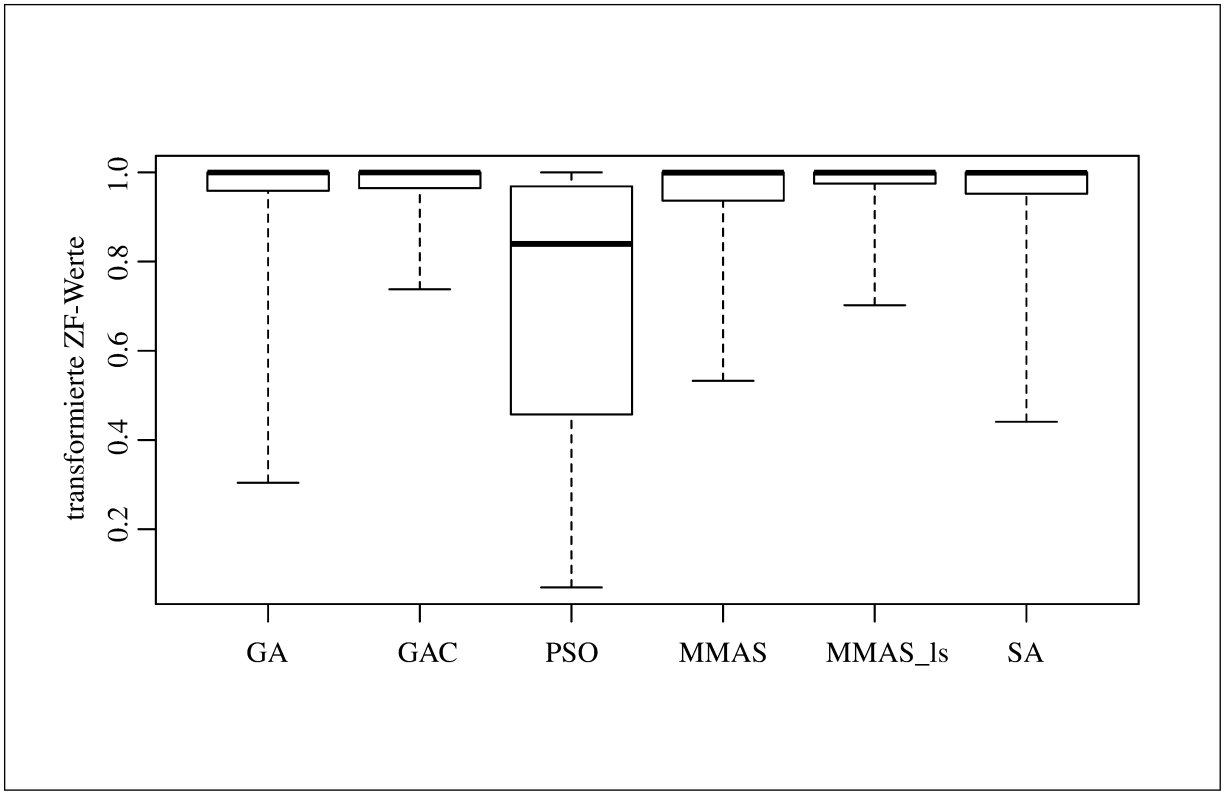

Abbildung 13: Boxplot der transformierten Zielfunktionswerte aller Zielfunktionen für jedes Verfahren. Stichprobengröße für jedes Verfahren: $n = 2400$. Die Boxplot-Whiskers beziehen sich auf den minimalen und maximalen transformierten Zielfunktionswert jeder Stichprobe. (Quelle: Eigene Darstellung)

jeder Iteration und passt die Pheromone entsprechend an. Zeitgleich führt der Evaporisationskoeffizient $\rho = 0.80$, der kleiner als der des MMAS ist ($\rho = 0.98$), zu einer besseren Exploration des Suchraums. Durch One-Opt wird verhindert, dass das Verfahren durch den kleinen Evaporisationskoeffizient nicht oder zu spät konvergiert. Für den Clusteransatz beim GAC bestätigen sich die Ergebnisse aus den Vorsimulationen ebenfalls. Die Berücksichtigung von Konsumentpräferenzen über den exakten Algorithmus aus Abschnitt 4.2.2 ggü. einer zufällig gewählten Startpopulation erweist sich als sehr erfolgreich und übertrumpft damit SA. Die Verwendung einer lokalen Suche beim GA brachte keine Verbesserung in der Qualität der Zielfunktionswerte. Im Gegenteil, One-Opt führte dazu, dass der GA vorzeitig gegen lokale Optima konvergiert. Der Clusteransatz funktionierte gleichfalls lediglich für ein Verfahren, den GA. Das erzeugen einer Startlösung über diesen Ansatz, um es für SA nutzbar zu machen, führte auch hier zur vorzeitigen Konvergenz von SA und somit zu schlechteren Zielfunktionswerten. Das Gleiche gilt für den MMAS. Die hiesige Implementierung von SA erzeugt sehr gute annähernde Optima, was aufgrund der Vorarbeit von Belloni et al. (2008b), in der SA am besten abschnitt, zu erwarten war und belegt somit den dritten Platz in dieser Untersuchung.

Die Laufzeiten unterscheiden sich erwartungsgemäß. Dies begründet sich in der unterschiedlichen Funktionsweise, aber auch durch Implementierungsdetails und die gewählte Programmiersprache *R*. Die Vergrößerung der Populationen des GA und des GAC führten ebenso zu

längeren Laufzeiten als in den vorherigen Simulationen. Die Laufzeiten des GAC liegen dabei über denen des GA durch die Berechnungen der $J_{\text{best}} = 50$ Produkte für jedes Cluster. Das beste Verfahren MMAS_ls über alle Zielfunktionen hinweg liefert die längsten Laufzeiten, was aufgrund der lokalen Suche nicht überrascht. Die Laufzeiten übersteigen diejenigen des MMAS deutlich. Die geringsten Laufzeiten weist die PSO auf, was vor allem damit zu begründen ist, dass dieses Verfahren durch das verwendete Mapping von stetigen zu diskreten Entscheidungsvariablen vorzeitig konvergiert, da es durch das Verstärken von lokalen Variablen kaum noch aus den lokalen Optima herauskommt. Dieses Problem zeigt sich besonders deutlich an den niedrigen Werten für $\overline{\mathbf{ZF}}_{max}^{mean}$ und $\overline{\mathbf{ZF}}_{mean}^{mean}$. Zum einen geht das Mapping mit einem starken Informationsverlust einher, da es im stetigen Lösungsraum operiert und anschließend in diskrete Entscheidungsvariablen umgewandelt wird. Zum anderen ist dieses Verfahren nicht gut für Kombinatorische Optimierungen geeignet, da es nicht dafür gemacht ist, diskrete Optimierungsprobleme zu lösen. Der Vollständigkeit halber und zur Demonstration der Ungeeignetheit wurde dieses Verfahren dennoch mit in die Betrachtung einbezogen.

Der Boxplot in Abbildung 13 gibt einen Eindruck für die erreichten Zielfunktionswerte. Aufgrund der kleineren Datensätze, in denen zumeist eine Performance von 1.00 erreicht wurde, liegt der Median für alle Verfahren bis auf PSO nahe 1.00. Generell zeigt sich eine relativ große Min-Max-Ausdehnung der Stichproben, die durch die Boxplot-Whiskers erkennbar wird. Andererseits zeigt sich jedoch eine gewisse Stabilität der Verfahren (außer PSO), da die Spannweite der Stichprobe, in der 50 % aller transformierten Zielfunktionswerte liegen, relativ klein ist.

5.3.2 Auswertung auf Entscheidungsregelebene

Auf Entscheidungsregelebene lässt sich die Rangfolge aus den Spalten für $\overline{\mathbf{ZF}}_{\cdots}^{Ent}$ aus Tabelle 26 ablesen. Die Mittelwerte der transformierten Zielfunktionswerte lassen sich aus Tabelle 25 entnehmen. Für die Untersuchung auf Entscheidungsregelebene wurden für die gemittelten, maximalen Zielfunktionswerte $n = 120$ und für die gemittelten Zielfunktionswerte $n = 1200$ transformierte Zielfunktionswerte berücksichtigt. Durch die Analyse der Performance der Verfahren auf Basis der Entscheidungsregeln treten einige Auffälligkeiten zutage, die bei rein aggregierter Betrachtung im Verborgenen geblieben wären. Für die probabilistische Entscheidungsregel zeigt sich wie in den Voruntersuchungen, dass der Lösungsraum deutlich flacher ist, als der der deterministischen Entscheidungsregel. D.h., die Zielfunktionswerte unterscheiden sich in ihrer Differenz weniger, wodurch die durchschnittlichen Zielfunktionswerte im Mittel deutlich näher an 1 liegen, sprich, an der besten gefundenen Lösung eines Conjointdatensatzes. Die Schlussfolgerung, dass die Verfahren daher für die probabilistische Entscheidungsregel somit besser geeignet wären als für die deterministische Entscheidungsregel, ist jedoch unzulässig, da die Qualität der Lösungen aufgrund der geringeren Varianz anders zu bewerten ist. Die unterschiedliche Beschaffenheit der Lösungsräume lässt sich u.a. aus dem Boxplot in Abbildung

14 ableiten. Für die deterministische Entscheidungsregel liegen die Mediane zwischen 93.68 % für den MMAS und 97.5 % für den MMAS_ls (ausgenommen PSO), wohingegen die Mediane für die probabilistische Entscheidungsregel (bis auf PSO) bei 1.00 liegen. Die Spannweiten unterscheiden sich ebenfalls deutlich. Während die ersten Quantile (ausgenommen PSO) für die deterministische Entscheidungsregel zwischen 88.30 % für SA und 94.93 % für MMAS_ls, sowie die dritten Quantile zwischen 96.60 % für MMAS und 100 % für GA und GAC liegen, existiert nur eine sehr schmale Spannweite für die probabilistische Entscheidungsregel. Hier liegen die ersten Quantile (außer PSO) zwischen 99.97 % für den MMAS_ls und 99.99 % für GA, GAC und MMAS. Die dritten Quantile liegen bei 100 % für alle Verfahren außer PSO.

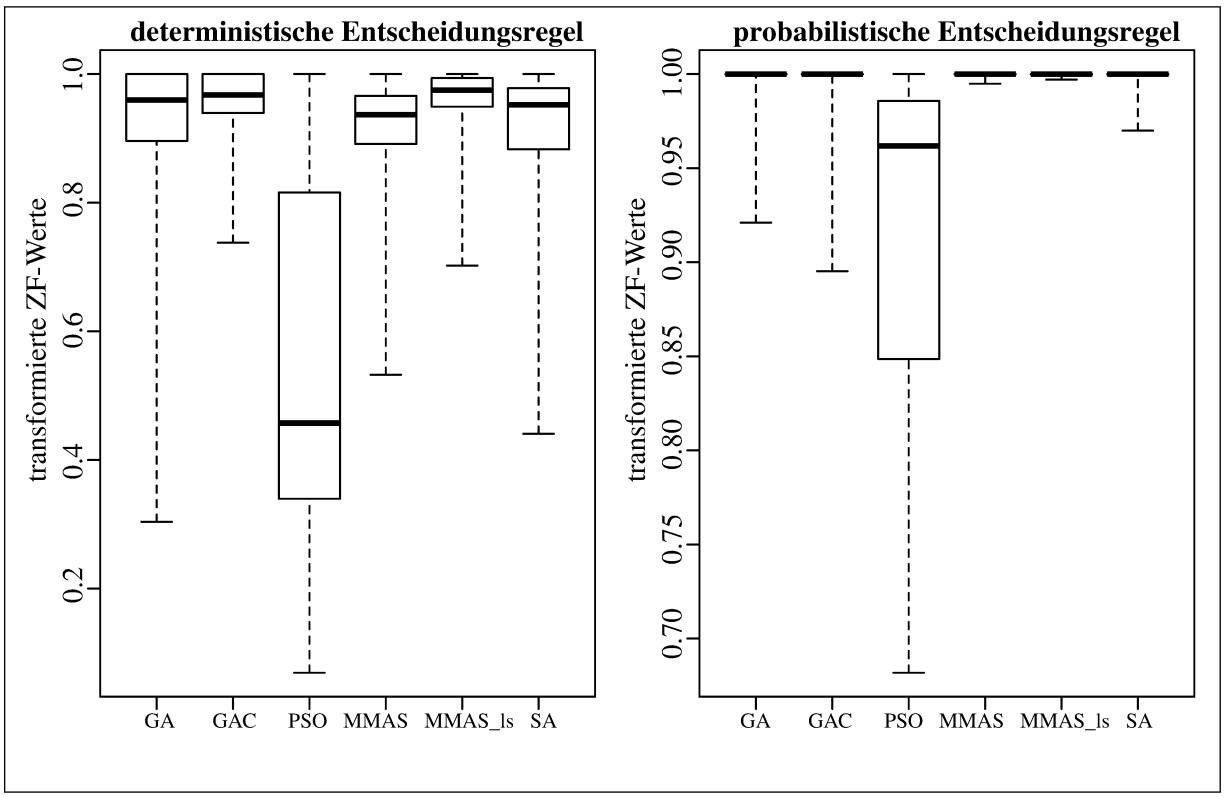

Abbildung 14: Boxplot der transformierten Zielfunktionswerte der Zielfunktionen für die Entscheidungsregeln für jedes Verfahren. Stichprobengröße für jedes Verfahren pro Entscheidungsregel: $n = 1200$. Die Boxplot-Whiskers beziehen sich auf den minimalen und maximalen transformierten Zielfunktionswert jeder Stichprobe. (Quelle: Eigene Darstellung)

Die Rangfolge der Verfahren für die deterministische Entscheidungsregel zeichnet ein klares Bild. Sowohl für $\overline{\mathbf{ZF}}_{max}^{Ent=det.}$ als auch für $\overline{\mathbf{ZF}}_{mean}^{Ent=det.}$ liefert MMAS_ls ($\overline{\mathbf{ZF}}_{max}^{Ent=det.} = 0.99523$; $\overline{\mathbf{ZF}}_{mean}^{Ent=det.} = 0.96492$) die beste Performance, gefolgt von GAC ($\overline{\mathbf{ZF}}_{max}^{Ent=det.} = 0.98042$; $\overline{\mathbf{ZF}}_{mean}^{Ent=det.} = 0.95775$) und SA ($\overline{\mathbf{ZF}}_{max}^{Ent=det.} = 0.96322$; $\overline{\mathbf{ZF}}_{mean}^{Ent=det.} = 0.92630$). Zwischen GA und MMAS existiert für ($\overline{\mathbf{ZF}}_{max}^{Ent=det.}(GA) = 0.94672$) bzw. ($\overline{\mathbf{ZF}}_{max}^{Ent=det.}(MMAS) = 0.95245$) kein statistisch signifikanter Unterschied auf dem 5 %-Niveau, wohingegen für $\overline{\mathbf{ZF}}_{mean}^{Ent=det.}$ der GA ($\overline{\mathbf{ZF}}_{mean}^{Ent=det.}(GA) = 0.91856$) dem MMAS ($\overline{\mathbf{ZF}}_{mean}^{Ent=det.}(MMAS) = 0.91262$) überlegen ist. PSO ($\overline{\mathbf{ZF}}_{max}^{Ent=det.} = 0.59971$; $\overline{\mathbf{ZF}}_{mean}^{Ent=det.} = 0.54682$) schneidet am schlechtesten ab.

Im Gegensatz zu der deterministischen Entscheidungsregel ergibt sich für die probabilistische Entscheidungsregel eine uneindeutigere Situation. Während sich die Rangfolge für $\overline{\mathbf{ZF}}_{mean}^{Ent=probab.}$ transitiv ausdifferenziert, existieren für $\overline{\mathbf{ZF}}_{max}^{Ent=probab.}$ zum Teil geringere bzw. keine signifikanten Unterschiede zwischen den Verfahren. Auffällig ist die Veränderung in der Rangfolge für die einzelnen Verfahren ggü. der deterministischen Entscheidungsregel. Am besten schneidet MMAS für $\overline{\mathbf{ZF}}_{mean}^{Ent=probab.}$ mit einem Wert von 0.99988 ab. Für $\overline{\mathbf{ZF}}_{max}^{Ent=probab.}$ sind sowohl MMAS ($\overline{\mathbf{ZF}}_{max}^{Ent=probab.} = 0.99998$) als auch SA ($\overline{\mathbf{ZF}}_{max}^{Ent=probab.} = 0.99966$) die besten Verfahren. Bezogen auf die maximalen Zielfunktionswerte gibt es allerdings keinen statistisch signifikanten Unterschied zwischen SA, MMAS_ls ($\overline{\mathbf{ZF}}_{max}^{Ent=probab.} = 0.99988$) und GA ($\overline{\mathbf{ZF}}_{max}^{Ent=probab.} = 0.99885$). Aufgrund der relativ geringen Stichprobengröße in Bezug auf die sehr kleinen zu messenden Unterschiede ($n = 120$) wird hier jedoch die eigentlich erwünschte Transitivität verletzt. Obwohl $\overline{\mathbf{ZF}}_{max}^{Ent=probab.}$ für MMAS_ls größer ist als derjenige von SA, gibt es einen statistisch signifikanten Unterschied auf dem 5 %-Niveau zwischen MMAS_ls und MMAS, aber zwischen SA und MMAS nicht. Insofern sind die Ergebnisse für diesen Fall vorsichtig zu bewerten. Zumal SA für $\overline{\mathbf{ZF}}_{mean}^{Ent=probab.}$ bei größerer Stichprobe ($n = 1200$) nur noch den dritten Rang erreicht. Insgesamt liegt MMAS_ls auf dem zweiten Rang. Der GA erreicht für $\overline{\mathbf{ZF}}_{mean}^{Ent=probab.}$ den vierten Rang. Ein deutlicher Unterschied existiert insgesamt für die Performance des GAC im Vergleich zur deterministischen Entscheidungsregel, wo er den zweiten Rang im Gegensatz zum fünften Rang für die deterministische Entscheidungsregel erreicht. PSO bildet weiterhin das Schlusslicht.

5.3.3 Auswertung auf Zielfunktionsebene

Die Rangfolge der Verfahren für die einzelnen Zielfunktionen lässt sich aus Tabelle 26 ablesen. Die transformierten, mittleren Zielfunktionswerte können aus Tabelle 25 entnommen werden. Auf Zielfunktionsebene werden für die gemittelten, maximalen Zielfunktionswerte $n = 60$ und für die gemittelten Zielfunktionswerte $n = 600$ transformierte Zielfunktionswerte berücksichtigt. Die Ausdifferenzierung der Analyse auf Zielfunktionsebene fördert im Gegensatz zur aggregierten Betrachtung weitere interessante Effekte zutage. So zeigt sich bspw. für M_{det}, dass der neue Clusteransatz für den GAC die besten gemittelten Zielfunktionswerte ($\overline{\mathbf{ZF}}_{mean}$) erzeugt. Für die maximalen, gemittelten Zielfunktionswerte ($\overline{\mathbf{ZF}}_{max}$) teilen sich GAC und MMAS_ls den ersten Rang, da es hier keinen statistisch signifikanten Unterschied gibt. Ein weiterer interessanter Punkt ist, dass MMAS den GA für M_{det} outperformt, aber GA den MMAS für G_{det}. Der Grund für diesen Effekt ist unklar. Man könnte vermuten, dass die alleinige Abhängigkeit von Nutzerpräferenzen MMAS zugute kommt. Dagegen spricht allerdings die Diskrepanz zwischen dem Abschneiden für G_{det} (4. bzw. 5. Rang) und G_{prob} (1. bzw. 1. Rang), wo nicht nur die Teilnutzenwerte in die Berechnungen eingehen, sondern auch die Deckungsbeiträge. Generell fällt für die Zielfunktionen auf, dass die Rangfolge für $\overline{\mathbf{ZF}}_{max}$ verschwimmt und es kaum statistisch signifikante Unterschiede zwischen einigen verschiedenen Verfahren

gibt. Dieser Effekt ist eine Folge des geringen Stichprobenumfangs, welcher sich bei $\overline{\mathbf{ZF}}_{mean}$ mit deutlich größerer Stichprobe nicht mehr zeigt. Da die Effekte im Allgemeinen sehr klein sind, benötigt der T-Test eine gewisse Stichprobengröße, um auch kleinere Effekte erkennen zu können. Aus diesem Grund wird sich für die weitere Beurteilung vor allem auf die Werte von $\overline{\mathbf{ZF}}_{mean}$ zur Beurteilung gestützt.

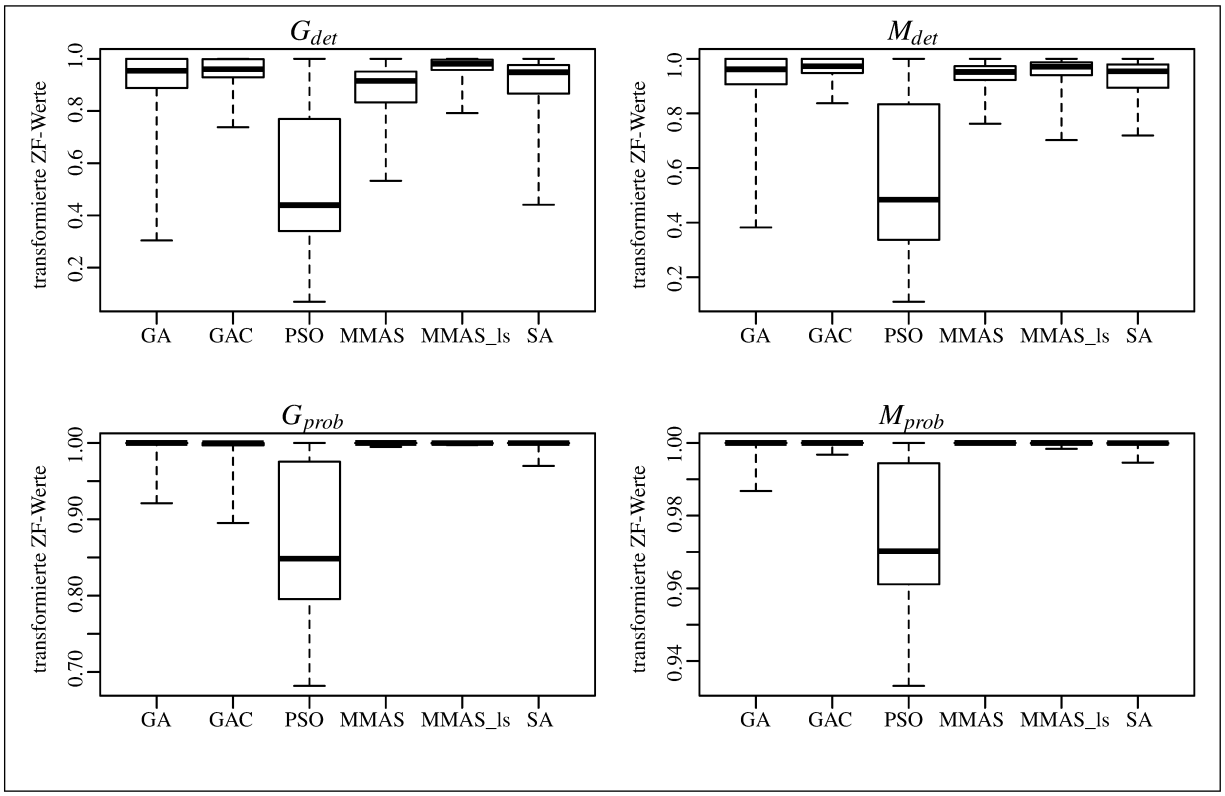

Abbildung 15: Boxplot der transformierten Zielfunktionswerte (ZF-Werte) jeder Zielfunktion für jedes Verfahren. Stichprobengröße für jedes Verfahren pro Zielfunktion: $n = 600$. Die Boxplot-Whiskers beziehen sich auf den minimalen und maximalen transformierten Zielfunktionswert jeder Stichprobe. (Quelle: Eigene Darstellung)

Für G_{det} ist MMAS_ls das stärkste Verfahren. Auf dem zweiten Rang folgt GAC. Diese Tatsache ist zunächst überraschend, sie hängt aber von dem zugrunde liegenden Conjointdatensatz und den Deckungsbeiträgen ab. Häufig erzielen diejenigen Produktlinien einen hohen Gewinn, die von vielen Konsumenten gekauft werden. Da GAC die Population durch den Clusteransatz in die Richtung derjenigen Produktlinien zieht, deren Produkte viele Konsumenten befriedigen, erklärt sich die gute Platzierung. Besitzen Produkte mit schwächeren Nutzenpräferenzen sehr hohe Deckungsbeiträge, kann es passieren, dass die gewinnmaximale Produktlinie aus wenigen Käufern, aber hochpreisigen Produkten besteht. In diesem Fall würde man eine schlechte Performance des GAC erwarten. Dieser Effekt des „Zusichziehens“ der Population des GAC zugunsten von vielversprechenden Konsumentenpräferenzen zeigt sich zusätzlich in der Performance für M_{det}, wo GAC noch vor MMAS_ls den ersten Rang belegt. Für beide Verfahren lässt sich dieses Bild in Abbildung 15 ebenfalls bestätigen, wo beide Verfahren die höchsten Mediane und die geringste Auslenkung in Richtung extremer Ausreißer aufweisen. GA und SA

tauschen für G_{det} (GA: 3. Rang; SA: 4. Rang) und M_{det} (GA: 4. Rang; SA: 3. Rang) jeweils die Plätze.

Für G_{prob} und M_{prob} belegt MMAS den ersten Rang. Das zweitbeste Verfahren für G_{prob} ist MMAS_ls. Bei M_{prob} teilen sich MMAS_ls und GAC den zweiten Rang. Dies ist wenig verwunderlich: während MMAS_ls durchgehend zu den stärksten Verfahren gehört, spielt GAC seine Stärke durch Einbeziehung der Konsumentenpräferenzen in die Startpopulation wieder aus. Für G_{prob} schneidet GAC jedoch mit dem fünften Rang eher schwach ab. Bei dieser Zielfunktion sind die Deckungsbeiträge für die Performance offensichtlich ein entscheidenderer Aspekt als die Konsumentenpräferenzen. SA belegt für G_{prob} den dritten und für M_{prob} den vierten Rang. GA belegt für G_{prob} den vierten und für M_{prob} den fünften Rang.

Die Boxplots für G_{prob} und M_{prob} zeigen wieder deutlich die unterschiedliche Struktur der Lösungsräume. Ein Hereinzoomen in die transformierten Zielfunktionswerte bringt hier keinen Mehrwert, da die Spannweiten zwischen den Quantilen entsprechend klein sind. Dennoch genügt es, um einen Eindruck für die Daten zu bekommen.

5.3.4 Diskussion und Fazit

Ein wichtiger Aspekt, der in diesem Abschnitt kurz diskutiert werden soll, ist die Benutzung des T-Tests über die Gesamtheit der Zielfunktionen sowie über die Entscheidungsregeln. Ausgehend von den originalen Zielfunktionswerten wurden die Zielfunktionswerte derart transformiert, dass sie im Intervall $[0,1]$ liegen, um eine gemeinsame Grundgesamtheit herzustellen, damit ein statistischer Test für gepaarte Stichproben durchgeführt werden kann. Dieses Vorgehen allein kann problematisch sein, da hierbei ein Informationsverlust droht. Die transformierten Zielfunktionswerte der probabilistischen Zielfunktionen liegen sehr viel näher an 1 als diejenigen der deterministischen Zielfunktionen und besitzen eine sehr viel geringere Standardabweichung (siehe Tabelle 25). Die Höhe der Mittelwerte sowie die Standardabweichung sind bei konstanter Stichprobengröße die determinierenden Größen für die Teststatistik. Werden nun Mittelwerte und Standardabweichungen von vier Zielfunktionen ($= 4$ Grundgesamtheiten) trotz Transformation zusammen als eine neue Grundgesamtheit betrachtet, kommt es zu Informationsverlusten für die probabilistischen Zielfunktionen, da für diese Werte eine deutlich geringere Standardabweichung und Differenz der transformierten Zielfunktionswerte in die Teststatistik eingeht als bei den Werten für die deterministischen Zielfunktionen. Die transformierten Zielfunktionswerte der probabilistischen Zielfunktionen werden somit von denjenigen der deterministischen Zielfunktionen dominiert. Dieser Effekt zeigt sich z. B. in Tabelle 26 für MMAS, der für die probabilistischen Zielfunktionen am besten abschneidet, insgesamt aber nur auf Rang 4 liegt. Des Weiteren wird die Normalverteilungsannahme des T-Tests verletzt, wenn mehrere Grundgesamtheiten miteinander vermischt werden. Man könnte argumentieren, dass durch die

Mittelwertbildung in der Teststatistik des T-Tests zwar der Zentrale Grenzwertsatz ab $n > 30$ zum Tragen kommt, aber der Zentrale Grenzwertsatz hat als Voraussetzung u.a., dass die Zufallsvariablen unabhängig und identisch verteilt sind. Diese Forderung wird streng genommen verletzt, da die transformierten Zielfunktionswerte nicht aus einer identischen Verteilung stammen. Nichtsdestotrotz wurde der T-Test angewendet, um zunächst ein Gefühl für die Daten zu bekommen. Das Vorgehen ist auch deshalb legitim, weil sich für die meisten Verfahren zeigt, dass die Ergebnisse des T-Tests robust sind und sich keine grundlegenden Widersprüche in den Rangfolgen zeigen. Dennoch ist die getrennte Betrachtung der Zielfunktionen wichtig, da sie einen Informationsmehrwert bieten. Alles in allem zeichnet sich durch die differenzierte Analyse ein harmonisches Bild auf allen betrachteten Ebenen.

Alle statistischen Tests wurden des Weiteren mit dem Wilcoxon-Rangsummentest durchgeführt, um die Forderung nach der Normalverteilung mit einem parameterfreien Test zu entschärfen. An den Rangfolgen änderte sich kaum etwas, außer dass die Trennschärfe nicht so gut wie beim T-Test war, wodurch es zu häufigeren Unentschieden zwischen den Verfahren kam.

Zusammenfassend lässt sich die *Forschnugsfrage 4*, wie folgt, beantworten: Das beste Verfahren unter den gegebenen Voraussetzungen ist über alle Zielfunktionen hinweg MMAS_ls. Wenn lediglich der deterministische Marktanteil M_{det} maximiert werden soll, ist der GAC mit neuem Clusteransatz das Verfahren der Wahl, vor allem in Anbetracht der besseren Laufzeiten ggü. MMAS_ls. Für die Zielfunktionen mit probabilistischer Entscheidungsregel sollte bei freier Wahlmöglichkeit der MMAS gewählt werden. Auch hier liegen die Laufzeiten deutlich unter denen von MMAS_ls. Für diese Art von Zielfunktionen scheint MMAS_ls ggü. MMAS im Optimierungsprozess zu frühzeitig gegen lokale Optima zu konvergieren.

6 Robustheit

Die erste Veröffentlichung, die sich mit dem Thema Robustheit in der Produktlinienoptimierung beschäftigte, ist von Camm et al. (2006). Die Autoren stellen fest, dass die Verfahren zur Optimierung von Produktlinien fehlerlose Teilnutzenwerte als statistische Schätzung verwenden. Dieses Vorgehen sei unrealistisch, da bei statistischen Schätzungen Messfehler inhärent sind und diese entsprechend berücksichtigt werden sollten (vgl. Camm et al. 2006, S. 444). Vor allem mit dem Aufkommen der hierarchischen Bayes-Schätzung (HB-Schätzung) Ende der 1990er und Anfang der 2000er Jahre, mit der individuelle Präferenzwerte geschätzt werden konnten, gab es eine Methode, die auf Basis eines zweistufigen Modells zur Modellierung der Posteriori-Verteilung der Präferenzen viele verschiedene Ziehungen von Teilnutzenwerten aus dieser Verteilung erlaubt. Die Posteriori-Verteilung setzt sich aus dem Produkt eines Logit-Modells mit einer multivariaten Normalverteilung zusammen. Das Logit-Modell berücksichtigt das individuelle Antwortverhalten eines Umfrageteilnehmers. Die multivariate Normalverteilung bezieht Informationen aus der Population der Umfrageteilnehmer. Die Idee dahinter ist, dass sich für die Schätzung der individuellen Präferenzen aufgrund der geringen Informationsdichte einer einzelnen Umfrage Informationen von den anderen Umfrageteilnehmern „geliehen“ wird. Mittels Markov-Chain-Monte-Carlo-Methoden (MCMC) werden die Parameter dieser Posteriori-Verteilung geschätzt und es wird eine empirische Verteilung erzeugt, aus der die Ziehungen stattfinden. Jede Ziehung besteht aus einer Matrix mit geschätzten Teilnutzenwerten mit den Konsumenten als Zeilen und den Eigenschaftsausprägungen als Spalten. Aus diesen Matrizen wird schlussendlich eine Matrix mit Punktschätzern durch Mittelwertbildung der Teilnutzenwerte erzeugt, so dass für jeden Umfrageteilnehmer ein Vektor aus Präferenzwerten für die Eigenschaftsausprägungen entsteht. Wenn eine HB-Schätzung durchgeführt wird, ist dies die Matrix, über die optimiert wird. Die Robustheit bezieht sich somit auf die Auswirkungen von Messfehlern auf die optimale Produktlinie. Camm et al. (2006) testeten die Robustheit ihres exakten Algorithmus an einem kleinen Datensatz mit einem einzelnen Produkt in der Produktlinie für die deterministische Marktanteilsmaximierung mit 3456 möglichen Produkten. Wang et al. (2009) untersuchen die Robustheit ihres Branch-and-Price-Ansatzes eines erweiterten Produktlinienoptimierungsmodells mittels exakten Algorithmus für eine hier nicht vorgestellte Zielfunktion. Wang/Curry (2012) entwickelten ein robustes Modell basierend auf einer HB-Schätzung für die deterministische Marktanteilsmaximierung für ein einzelnes Produkt in der Produktlinie, dass die Unsicherheit in den Teilnutzenwerte direkt miteinbezieht.

Belloni et al. (2008b) testeten die Robustheit ihres exakten Algorithmus für das deterministische Gewinnmaximierungsmodell im Einproduktfall, wobei sie keinen Zugriff auf Ziehungen einer HB-Schätzung hatten. Sie verwendeten als Basis die Teilnutzenwerte einer Regressionsanalyse, die sie mit einem normalverteilten Fehler mit Erwartungswert 0 und einer Varianz, die aus den Standardfehlern der Schätzung für die Regressionsanalyse stammt.

Dieses Kapitel soll die bisherige Forschung ergänzen, indem es für die vier Zielfunktionen in dieser Arbeit die Robustheit an einem kleineren, realen Conjointdatensatz testet. Der Unterschied liegt nicht nur in der größeren Zahl der betrachteten Zielfunktionen, sondern vor allem darin, dass die hier vorgestellten Verfahren zur Lösung von Produktlinienoptimierungsmodellen verwendet werden. Für die probabilistische Marktanteilsmaximierung kann zusätzlich der in dieser Arbeit entwickelte exakte Algorithmus verwendet werden. Zu erwarten ist eine höhere Schwierigkeit dadurch, dass nicht nur die Messfehler einen Einfluss auf die optimalen Produktlinien haben werden, sondern auch Abweichungen von diesen optimalen Produktlinien durch die approximative Natur der Machine-Learning-Verfahren.

In der folgenden Untersuchung soll für die vier Zielfunktionen die Robustheit der HB-Schätzung für einen realen CBC-Datensatz bzgl. der Produktlinienoptimierung mittels der hier vorgestellten Verfahren überprüft werden. PSO wird hierbei ausgeklammert, da die Performance in den Simulationen durchgehend so schlecht war, dass für diesen Datensatz kein besseres Abschneiden zu erwarten ist.

6.1 Beschreibung des Datensatzes

Für die Untersuchung wird ein kleinerer Datensatz herangezogen. Der Grund liegt darin, dass die Optimierung, wie in Camm et al. (2006, S. 444), 1000 Ziehungen aus der Markov-Kette beinhaltet. Jedes Verfahren (außer PSO) löst jeden dieser 1000 Datensätze einmal für jede Zielfunktion. D.h., jedes Verfahren löst in dieser Untersuchung 4000 Optimierungsprobleme. Ein weiterer Grund für die Verwendung des nachfolgenden Datensatzes ist die Tatsache, dass für diesen alle vier Zielfunktionen ausgewertet werden können, was nicht für jeden Datensatz möglich ist.

Der Datensatz stammt aus einer CBC, die im Jahr 2010 im Rahmen der Diplomarbeit des Autors dieser Arbeit entstand. Es wurde die Vorziehenswürdigkeit verschiedener Basketbälle untersucht. An der Umfrage nahmen 251 Umfrageteilnehmer teil. Die Eigenschaften und Eigenschaftsausprägungen waren:

- Marke: Marke 1, Marke 2, Marke 3.

- Preis in EUR: 15, 30, 45, 60, 75.
- Material: Gummi, Leder, mikroperforiertes Material, Verbundleder.
- Spielort: Indoor, Outdoor, Indoor/Outdoor.

Die Marken sind von drei bekannten Herstellern von Basketbällen. Zur Veröffentlichung der Untersuchung werden die Marken aus rechtlichen Gründen kodiert dargestellt. Die Umfrage umfasste für jeden Teilnehmer 20 Fragen mit jeweils drei Produktprofilen pro Frage inkl. einer Nichtwahl-Option.

Eigenschaft	**Eigenschaftsausprägung**	**aggregierter Teilnutzenwert β_{kl}**
Marke		
	Marke 1	0.3357
	Marke 2	−0.3923
	Marke 3	0.0566
Preis		
	15 EUR	1.0829
	30 EUR	0.9451
	45 EUR	0.2533
	60 EUR	−0.7063
	75 EUR	−1.5751
Material		
	Gummi	0.3484
	Leder	0.3551
	mikroperforiert	−1.9333
	Verbundleder	1.2299
Spielort		
	Indoor	−0.2728
	Outdoor	−0.1777
	Indoor/Outdoor	0.4500
Nicht-Wahl-Option		−0.4726

Tabelle 28: Aggregierte Teilnutzenwerte der hierarchischen Bayes-Schätzung der Choice-based-Conjoint-Studie. (Quelle: Eigene Darstellung)

Die 20 Fragen wurden mit der Random-Methode erzeugt, d.h., die Produktprofile wurden zufällig innerhalb ihrer Eigenschaftsausprägungen gleichverteilt erzeugt. Für die HB-Schätzung wurden insgesamt 20000 Ziehungen aus der Posteriori-Verteilungen getätigt. Nach einer Burn-in-Phase von 10000 Iterationen, um die Konvergenz der Markov-Kette gegen ihre stationäre Verteilung zu gewährleisten, wurde jede 10. Ziehung gespeichert. Nur jede 10. Ziehung deswegen, um eine Korrelation zwischen den einzelnen Ziehungen zu vermeiden, damit die Schät-

zung nicht verzerrt wird. Daraus resultieren schlussendlich 1000 Matrizen mit den Teilnutzenwerten jeder Ziehung für jeden Umfrageteilnehmer. Zusätzlich wird als Vergleich für die Robustheit die Punktschätzung der Teilnutzenwertmatrix verwendet, die im Normalfall die Basis für weitere Auswertungen ist. Die individuellen Teilnutzenwerte wurden sodann mittels einer HB-Schätzung auf Basis eines multinomialen Logit-Modells geschätzt. Der Root-Likelihood-Wert (RLH-Wert), der als Gütemaß dient und zwischen 0 und 1 liegt, ist bei der Studie bei $RLH = 0.6232$, was einem für eine CBC durchschnittlichen RLH-Wert entspricht (vgl. Selka 2013, S. 77).

Als Übersicht sind in Tabelle 28 die aggregierten Teilnutzenwerte über alle Umfrageteilnehmer hinweg gegeben.

6.2 Versuchsaufbau

Im Gegensatz zu Camm et al. (2006), die für ihre Untersuchung 100 Ziehungen einer HB-Schätzung zur Verfügung hatten, um daraus zufällig 1000 Ziehungen für die Robustheit zu erzeugen, werden in dieser Arbeit die originalen, unveränderten Ziehungen der HB-Schätzung verwendet. Das Problem an dem Vorgehen von Camm et al. (2006) ist, dass die 1000 Teilnutzenwertmatrizen nicht mehr der ursprünglichen Posteriori-Verteilung folgen, wenn aus 100 originalen Ziehungen zufällig Teilnutzenwertmatrizen aus den Umfrageteilnehmern zusammengewürfelt werden.

Die Teilnutzenwertmatrizen wurden wie in Kapitel 5 normalisiert, so dass die Teilnutzenwerte im Intervall $[0, 1]$ liegen. Da für diesen Datensatz die Teilstückdeckungsbeiträge nicht bekannt sind, wird hier der Umsatz aus den Preisen der Basketbälle verwendet. **Es handelt sich also streng genommen um eine Umsatzmaximierung**, was für die Untersuchung der Robustheit jedoch unerheblich ist. Damit eine Robustheitsstudie für den verwendeten Datensatz durchgeführt werden kann, muss ein Szenario erstellt werden. Ein realistisches Szenario könnte das folgende sein.

Szenario für die Produktlinienoptimierung:

- Der Hersteller *Marke 2* von Basketbällen möchte seine alte Produktlinie überarbeiten und eine neue Produktlinie einführen.
- Die Produktlinie soll 3 Basketbälle beinhalten.
- Der Hersteller möchte entweder seinen Umsatz oder seinen Marktanteil maximieren.

- Der Status-quo-Markt besteht aus 6 Produkten konkurrierender Hersteller.

Für den Status-quo-Markt wurden sechs Produkte zufällig erzeugt. Die entsprechenden Produktprofile können aus Tabelle 29 entnommen werden. Da es sich um einen konkreten Hersteller handelt, der eine Produktlinie einführen möchte, werden die Marken nicht für den eigentlichen Optimierungsprozess verwendet. Die Marke steht fest und wird somit lediglich für die Zielfunktionsevaluationen verwendet, aber nicht während der Optimierung variiert. Aus diesem Grund verkleinert sich der Datensatz und beinhaltet fünf Preise, vier Materialien und drei Spielorte. Daraus ergeben sich $\binom{5 \cdot 4 \cdot 3}{3} = 34220$ mögliche Produktlinien.

Produkt	Marke	Preis	Material	Spielort
1	Marke 1	75 EUR	Verbundleder	Indoor
2	Marke 1	60 EUR	Leder	Outdoor
3	Marke 1	30 EUR	mikroperforiertes Material	Indoor/Outdoor
4	Marke 3	15 EUR	Gummi	Indoor/Outdoor
5	Marke 3	30 EUR	Leder	Indoor/Outdoor
6	Marke 3	60 EUR	mikroperforiertes Material	Indoor

Tabelle 29: Zufällig erzeugter Status-quo-Markt für die Robustheitsstudie. (Quelle: Eigene Darstellung)

Basierend auf diesem Aufbau werden insgesamt 1001 Optimierungen (1000 Ziehungen der HB-Schätzung + eine Punktschätzung) durchgeführt. Die Punktschätzung der Teilnutzenwertmatrix wird als Referenz für den Vergleich mit den 1000 Ziehungen verwendet. Dieses Vorgehen entspricht Belloni et al. (2008b, S. 1550) und Camm et al. (2006, S. 444).

Die Parametereinstellungen für die Verfahren erfolgt wie in Kapitel 5 adaptiv gemäß Tabelle 24.

6.3 Test auf Robustheit

Die Ergebnisse der Robustheitsstudie der HB-Schätzung bzgl. der Produktlinienoptimierung, speziell für die hier betrachteten Zielfunktionen, sind in Tabelle 30 dargestellt. Für die Untersuchung der Robustheit definieren Belloni et al. (2008b, S. 1550) die Punktschätzung der Präferenzen (bzw. dort: Schätzwerte einer Regression) als „wahre“ Teilnutzenwerte der Konsumenten. Jede Ziehung aus der Markov-Kette der HB-Schätzung unterliegt nach dieser Interpretation Messfehlern. Damit sind für diese Untersuchung die Teilnutzenwerte der Punktschätzung die originalen und somit die Referenz-Teilnutzenwerte.

Zunächst fällt auf, dass die Zielfunktionswerte der Optimierung auf Basis der Punktschätzung sowie auf Basis der 1000 Ziehungen für jedes Verfahren und jede Zielfunktion gleich sind. Auf-

ZF	Verfahren	Optimum der Punktschätzung in EUR bzw. %	$\overline{ZF}$ in EUR bzw. % der 1000 Ziehungen	in % des Optimums der Punktschätzung	kein gemeinsames Produkt in Produktlinie in %	genau 1 gemeinsames Produkt in Produktlinie in %	genau 2 gemeinsame Produkte in Produktlinie in %	genau 3 gemeinsame Produkte in Produktlinie in %
$\mathbf{G}_{det}$	GA	**4080.00**	4347.38	106.55	25.60	42.50	25.30	6.60
	GAC	**4080.00**	4347.38	106.55	25.50	43.10	24.80	6.60
	MMAS	**4080.00**	4347.38	106.55	25.90	42.00	25.10	7.00
	MMAS_ls	**4080.00**	4347.38	106.55	25.50	42.30	25.40	6.80
	SA	**4080.00**	4347.38	106.55	26.10	42.00	24.80	7.10
$\mathbf{M}_{det}$	GA	**44.62**	44.80	100.40	5.40	58.50	32.00	4.10
	GAC	**44.62**	44.80	100.40	5.50	58.40	32.20	3.90
	MMAS	**44.62**	44.80	100.40	4.60	57.40	33.70	4.30
	MMAS_ls	**44.62**	44.80	100.40	5.20	59.10	31.60	4.10
	SA	**44.62**	44.80	100.40	5.20	59.30	32.00	3.50
$\mathbf{G}_{prob}$	GA	**6156.89**	6188.95	100.52	0.00	0.10	6.60	93.30
	GAC	**6156.89**	6188.95	100.52	0.00	0.10	6.60	93.30
	MMAS	**6156.89**	6188.95	100.52	0.00	0.10	6.60	93.40
	MMAS_ls	**6156.89**	6188.95	100.52	0.00	0.10	6.60	93.30
	SA	**6156.89**	6188.95	100.52	0.00	0.10	6.60	93.30
$\mathbf{M}_{prob}$	GA	**34.99**	34.56	98.77	0.00	0.00	24.40	75.60
	GAC	**34.99**	34.56	98.77	0.00	0.00	24.40	75.60
	MMAS	**34.99**	34.56	98.77	0.00	0.00	24.40	75.60
	MMAS_ls	**34.99**	34.56	98.77	0.00	0.00	24.40	75.60
	SA	**34.99**	34.56	98.77	0.00	0.00	24.40	75.60

Tabelle 30: Ergebnisse der Robustheitsstudie der Optimierung für vier Zielfunktionen und fünf Verfahren. Alle Zielfunktionswerte entsprechen den exakten Maxima für die 1000 Ziehungen sowie der Punktschätzung. Jedes Verfahren fand in jedem Optimierungsdurchlauf das tatsächliche Optimum (verifiziert durch komplette Enumeration). Es handelt sich um die originalen Zielfunktionswerte. Anmerkung: Die Spalte „genau 2 gemeinsame Produkte in Produktlinie“ enthält nicht die Spalte „genau 1 gemeinsames Produkte in Produktlinie“ als Teilmenge. Die drei letzten Spalten ergeben sich in Summe zu 100 %, außer wenn die optimalen Produktlinien der Punktschätzung kein gemeinsames Produkt mit der Produktlinie einer der Ziehungen hatte. Die Gewinnfunktionen repräsentieren in diesem Beispiel den Umsatz. (Quelle: Eigene Darstellung in Anlehnung an Belloni et al. (2008b))

grund der geringen Anzahl möglicher Produktlinien finden die Verfahren für jede Ziehung das exakte Optimum, was mittels kompletter Enumeration verifiziert wurde. Die Performance der

Verfahren hat in diesem kleinen Beispiel also keinen Einfluss auf die Robustheit. Diese würde weiter abnehmen, wenn nicht nur Messfehler, sondern zusätzlich Fehler durch die approximative Natur der Verfahren gemacht würden. Die Zielfunktionswerte in Tabelle 30 entsprechen den Gewinnen (hier: Umsätzen) bzw. den Marktanteilen der Basketbälle, die auf Grundlage des Status-quo-Markts für die 251 Umfrageteilnehmer erzielt wurden. Die Gewinne (hier: Umsätze) des Gewinnmaximierungsmodells mit deterministischer Entscheidungsregel G_{det} sind geringer als die Gewinne (hier: Umsätze) des Gewinnmaximierungsmodells mit probabilistischer Entscheidungsregel G_{prob}, obwohl es beim Marktanteil andersherum ist. Die Gründe hierfür sollen an dieser Stelle nicht weiter erörtert werden, da es hier nicht um einen Vergleich der Zielfunktionskonzepte untereinander geht. Die gemittelten Zielfunktionswerte der 1000 Ziehungen sind für alle Zielfunktionen höher als für die Zielfunktionswerte der Punktschätzung, außer für die probabilistische Marktanteilsmaximierung M_{prob}. Für den Einproduktfall in Belloni et al. (2008b, S. 1550) sind die gemittelten Gewinne geringer als die Gewinne mittels „wahrer“ Teilnutzenwerte (hier: Optimum der Punktschätzung) für alle betrachteten Verfahren. Die Autoren schlussfolgern, dass durch die Messfehler ein Verlust an Informationen einhergeht und der Gewinn daher geringer ausfällt. Diese Interpretation ist diskussionswürdig, da ein Informationsverlust nicht nur in die eine Richtung gehen kann, sondern auch in die andere, nämlich, dass die Zielfunktionswerte hierdurch auch steigen können, wie die Spalten „$\overline{ZF}$ in EUR bzw. % der 1000 Ziehungen“ und „in % des Optimums der Punktschätzung“ in Tabelle 30 zeigen. Hierfür wurden die einzelnen Optima der 1000 Ziehungen arithmetisch gemittelt, was teils zu höheren (siehe G_{det}, M_{det} und G_{prob}), aber auch zu niedrigeren durchschnittlichen Zielfunktionswerten (siehe M_{prob}) führen kann. Sehr stark fällt der Unterschied wiederholt zwischen den Entscheidungsregeln auf. Während für G_{prob} 93.30 % und M_{prob} immerhin noch 75.60 % der Produktlinien der 1000 Ziehungen mit der Produktlinie der Punktschätzung übereinstimmen, sind es bei G_{det} lediglich $6.60 - 7.10$ % und bei M_{det} zwischen $3.50 - 4.30$ %. Obwohl jedes Verfahren den gleichen Zielfunktionswert für G_{det} und M_{det} liefert, unterscheiden sich die Anteile für die gemeinsamen Produkte in einer Produktlinie untereinander. Der Grund hierfür liegt darin, dass in einigen der 1000 Ziehungen zwei optimale Produktlinien existieren. Findet ein Verfahren eine optimale Produktlinie in einer Ziehung, die nicht mit dem Optimum der Punktschätzung übereinstimmt, so wird diese Produktlinie nicht als gleiche gezählt. Der Effekt ist jedoch nicht besonders groß, sonst wäre die Diskrepanz in den gemeinsamen Produkten größer. Für die Punktschätzung jedoch gab es für beide Zielfunktionen ein eindeutiges Optimum. Dieser Unterschied existiert für G_{prob} und M_{prob} nicht. Das liegt daran, dass es durch die unterschiedliche Berechnung der Zielfunktionswerte deutlich unwahrscheinlicher ist, dass zwei verschiedene Produkte den gleichen Zielfunktionswert besitzen. Hierfür müssten zwei Eigenschaftsausprägungen der gleichen oder einer anderen Eigenschaft den exakt gleichen Teilnutzenwert aufweisen oder ein einzelner Teilnutzenwert die Linearkombination anderer Eigenschaftsausprägungen anderer Eigenschaften sein. Da das aber für den vorliegenden Datensatz nicht gegeben ist, existieren keine mehrfachen Optima für diese Zielfunktionen. Für G_{prob} gab es aus den 1000 Ziehungen nur eine Produktlinie (0.10 %), die ein gemeinsames Produkt mit

der Produktlinie der Punktschätzung besitzt. Für M_{prob} bestehen alle Produktlinien aus mindestens zwei gemeinsamen Produkten mit der Produktlinie der Punktschätzung. Eine hohe Anzahl von keinen gemeinsamen Produkten ($25.50 - 26.10$ %) mit der Produktlinie der Punktschätzung besitzt G_{det}. Diese Zielfunktion reagiert offenbar besonders sensitiv auf Messfehler bzw. einzelne Ziehungen der HB-Schätzung. Generell ist die Sensitivität bzgl. der optimalen Produktlinien für die Zielfunktionen mit deterministischen Entscheidungsregeln bei Änderungen in den Teilnutzenwerten größer als bei den probabilistischen Entscheidungsregeln. Ursächlich dafür ist zum einen sicherlich die IIA-Eigenschaft der probabilistischen Entscheidungsregeln (siehe Abschnitt 2.2.1.2), wodurch die Produktlinien meist aus Produkten bestehen, die sich in nur einer Eigenschaftsausprägung unterscheiden, was generell zu stabileren Optima zu führen scheint. Zum anderen ist es bei der deterministischen Entscheidungsregel so, dass durch die sich ändernden Teilnutzenwerte die Konsumenten von ihrem ursprünglichen Produkt zu einem anderen Produkt mit nun höherem Nutzenwert wechseln, wodurch sich die Zusammenstellung der Produktlinie stark ändern kann.

	$\mathbf{G}_{det}$			$\mathbf{M}_{det}$			$\mathbf{G}_{prob}$			$\mathbf{M}_{prob}$		
	$\mathbf{ZF}_{min}$	$\overline{\mathbf{ZF}}$	$\mathbf{ZF}_{max}$	$\mathbf{ZF}_{min}$	$\overline{\mathbf{ZF}}$	$\mathbf{ZF}_{max}$	$\mathbf{ZF}_{min}$	$\overline{\mathbf{ZF}}$	$\mathbf{ZF}_{max}$	$\mathbf{ZF}_{min}$	$\overline{\mathbf{ZF}}$	$\mathbf{ZF}_{max}$
GA	68.01	91.07	100	77.68	94.36	100	99.55	99.98	100	99.71	99.97	100
GAC	68.01	90.96	100	77.68	94.32	100	99.55	99.98	100	99.71	99.97	100
MMAS	68.01	91.06	100	77.68	94.53	100	99.55	99.98	100	99.71	99.97	100
MMAS_ls	68.01	91.12	100	77.68	94.39	100	99.55	99.98	100	99.71	99.97	100
SA	68.01	91.03	100	77.68	94.30	100	99.55	99.98	100	99.71	99.97	100

Tabelle 31: Optimale Produktlinien der 1000 Ziehungen der HB-Schätzung eingesetzt in die Zielfunktion mit den Teilnutzenwerten aus der Punktschätzung. $\mathbf{ZF}_{min}$ bzw. $\mathbf{ZF}_{max}$ sind der kleinste bzw. größte Zielfunktionswert, der für ein Optimum einer Ziehung für die Zielfunktion mit den Teilnutzenwerten aus der Punktschätzung erzielt wurde. $\overline{\mathbf{ZF}}$ bezeichnet das arithmetische Mittel der Zielfunktionswerte der Produktlinien der 1000 Ziehungen, eingesetzt in die Zielfunktion der Teilnutzenwerte aus der Punktschätzung. (Quelle: Eigene Darstellung in Anlehnung an Camm et al. (2006))

Wie in Camm et al. (2006) wird die Untersuchung der Robustheit in zwei Phasen aufgeteilt: in die Testphase (siehe Tabelle 30), die die Optima der 1000 Ziehungen untersucht und die Validierungsphase (siehe Tabelle 31). In der Validierungsphase werden die optimalen Produktlinien der 1000 Ziehungen in die Zielfunktionen mit den Teilnutzenwerten der Punktschätzung eingesetzt, deren kleinster (ZF_{min}) und größter (ZF_{max}) Zielfunktionswert bestimmt, sowie das arithmetische Mittel ($\overline{ZF}$) über die 1000 Zielfunktionswerte für jede Zielfunktion berechnet. Für das Maximum ZF_{max} sind die gleichen Zielfunktionswerte zu erwarten wie für die Punktschätzung, da jede Zielfunktion für die 1000 Ziehungen mindestens 35 gleiche Produktlinien besitzt (siehe Tabelle 30). Diese Erwartung ist korrekt, wie sich in Tabelle 31 für die Maximalwerte anteilig mit 100 % zeigt. Eine andere Erwartung ist, dass die mittleren Zielfunktionswerte $\overline{ZF}$ kleiner sind als die Optima, da Produktlinien existieren, die nicht der optimalen Produktlinie entsprechen und somit zu niedrigeren Zielfunktionswerten führen müssen. Für jede Zielfunktion und jedes Verfahren liegen die gemittelten Zielfunktionswerte bei mindestens 90.96 % (GAC für G_{det}). Die kleinen Abweichungen kommen, wie oben, von multiplen optimalen

Produktlinien für einige Ziehungen, sind jedoch sehr klein. Während die Schwankungsbreite für die Zielfunktionen mit probabilistischer Entscheidungsregel zwischen minimalen und maximalen Zielfunktionswerten relativ klein ist ($99.55 - 100$ % für G_{prob} und $99.71 - 100$ % für M_{prob}), ist die Schwankungsbreite für die deterministischen Entscheidungsregeln bedeutend höher. Hier erreichte die schlechteste Produktlinie lediglich Zielfunktionswerte für alle Verfahren von 68.01 % des Optimums der Punktschätzung für G_{det} und 77.68 % für M_{det}. Somit lässt sich zunächst konstatieren, dass die Test- und Validierungsphase konsistente Ergebnisse in Bezug auf die Auswirkungen von Messfehlern in den Teilnutzenwerten liefern.

Forschungsfrage 5, ob die Verfahren für alle Zielfunktionen robust auf Messfehler in den Teilnutzenwerten reagieren, muss zweierlei beantwortet werden. Zunächst lässt sich festhalten, dass die verwendeten Machine-Learning-Verfahren keinen Einfluss auf die Untersuchung hatten, da sie die kleine Probleminstanz für alle Ziehungen sowie die Punktschätzung optimal lösten. Damit bleibt die Frage der Robustheit der HB-Schätzung für das vorliegende Beispiel. Für die Zielfunktionen mit probabilistischer Entscheidungsregel ist die HB-Schätzung trotz der Existenz von Messfehlern äußerst robust. In der Test- und Validierungsphase zeigten sich Vorteile für G_{prob} gegenüber M_{prob}, für das 93.30 % der Produktlinien mit der optimalen Produktlinie der Punktschätzung übereinstimmen. Auch in der Validierungsphase besitzt G_{prob} für die gemittelten Zielfunktionswerte gegenüber M_{prob} einen leicht höheren Wert (99.98 % vs. 99.97 %). Für die Zielfunktionen mit deterministischen Entscheidungsregeln stellt sich die Lage anders dar. Sie reagieren sehr sensitiv auf das Vorhandensein von Messfehlern, was sich zum einen in der Testphase an den verhältnismäßig wenigen Produktübereinstimmungen, vor allem für G_{det} zeigt, und zum anderen an den schlechten Minimalwerten und den knapp über 5 % abnehmenden, gemittelten Zielfunktionswerten für M_{det} und ca. 9 % abnehmenden, gemittelten Zielfunktionswerten für G_{det} in der Validierungsphase liegt. G_{det} reagiert nochmal deutlich sensitiver auf die Anwesenheit von Messfehlern auf die Robustheit als M_{det}.

Abschließend lässt sich schlussfolgern, dass die Art der Entscheidungsregel einen großen Einfluss auf die Robustheit der HB-Schätzung für die Produktlinienoptimierung hat. Vor allem deterministische Entscheidungsregeln reagieren sensitiver auf sich durch Messfehler ändernde Teilnutzenwerte. Diese Untersuchung kann als Entscheidungsunterstützung für die Wahl einer Zielfunktion hilfreich sein. Als alleinige Entscheidungsgrundlage ist sie jedoch nicht geeignet, liefert allerdings wertvolle Informationen, um angemessen auf Messfehler reagieren zu können. Generell ist die Aussagekraft dieser Untersuchung eingeschränkt, da es sich lediglich um einen einzelnen untersuchten Datensatz handelt. Für allgemeingültigere Erkenntnisse wäre eine Analyse von einer großen Zahl an Datensätzen zielführend. Des Weiteren handelte es sich bei der Untersuchung um einen kleinen Datensatz, wodurch die Machine-Learning-Verfahren sehr stabil liefen. Die Analyse von größeren Datensätzen, für die lediglich approximative, lokale Optima gefunden werden können, könnte tiefere Einblicke in die Qualität der lokalen Optima gewähren.

7 Schlussbetrachtung und Ausblick

Die vorliegende Arbeit beschäftigt sich mit der Produktlinienoptimierung im Allgemeinen und fünf Machine-Learning-Verfahren zum approximativen Lösen dieser Optimierungsprobleme im Speziellen. Der Fokus liegt auf der Performance der ausgewählten Machine-Learning-Verfahren für vier verschiedene Zielfunktionen, die sich zweier Entscheidungsregeln für die Modellierung von Konsumentenpräferenzen bedienen. Weiterhin werden für jedes Machine-Learning-Verfahren adaptive Parameterkonfigurationen eingeführt und evaluiert. Schließlich wird der Einfluss von Messfehlern auf die Robustheit der HB-Schätzung für alle Machine-Learning-Verfahren untersucht. In diesem Kapitel werden die wichtigsten Ergebnisse zusammengefasst, kritisch diskutiert und es wird ein Ausblick auf die zukünftige Forschung auf diesem Gebiet gegeben.

7.1 Schlussbetrachtung und Zusammenfassung der wichtigsten Ergebnisse

Beginnend mit **Kapitel 2** wird eine kurze Einführung in die Produktliniengestaltung gegeben. Dabei werden theoretische und normative Ansätze unterschieden. Anschließend wird die Grundlage für die Produktlinienoptimierung auf Basis der Conjointanalyse gelegt. Es werden die deterministische und die probabilistische Entscheidungsregel für die Modellierung von Konsumentenpräferenzen vorgestellt. Diese Entscheidungsregeln bilden die Voraussetzungen für die Zielfunktionen. Es werden für jede Entscheidungsregel je ein Gewinnmaximierungs- und ein Marktanteilsmaximierungsmodell beschrieben.

In **Kapitel 3** wird die theoretische Grundlage für die Machine-Learning-Verfahren gelegt. Kombinatorische Optimierungsprobleme werden definiert, es wird mit einer Übersicht über verschiedene Verfahren auf die Probleme des Forschungsfelds hingewiesen und durch das „No free lunch theorem“ für Machine-Learning-Verfahren beschrieben, worin die Schwierigkeit bei der Auswahl eines geeigneten Verfahrens für ein spezielles Optimierungsproblem liegt. Folgend werden GA, PSO, ACO, MMAS und SA in ihrer allgemeinen Funktionsweise dargelegt.

Aufbauend auf den Kapiteln 2 und 3 werden in **Kapitel 4** die Machine-Learning-Verfahren für die Produktlinienoptimierung vorbereitet. In der Literaturübersicht werden alle Veröffentlichungen seit 1995 berücksichtigt, die sich mit diesem Thema beschäftigten. Des Weiteren wird eine Übersicht über exakte Methoden gegeben, die ein Optimierungsproblem optimal lösen können. Stichpunkthaft werden die wichtigsten Erkenntnisse des Kapitels zusammengefasst:

- Es wird ein exakter Lösungsalgorithmus für die probabilistische Marktanteilsmaximierung angegeben, dessen Optimalität mathematisch bewiesen wird (siehe Abschnitt 4.2.2). Somit kann eine positive Antwort auf **Forschungsfrage 1** gegeben werden, die darauf abzielte, ob das probabilistische Marktanteilsmaximierungsproblem optimal in angemessener Laufzeit gelöst werden kann. Dieser Lösungsalgorithmus nutzt eine Eigenschaft des Optimierungspackages *lpSolve* aus, wonach die Berechnung eines einzelnen optimalen Produkts für die probabilistische Marktanteilsmaximierung in Sekundenbruchteilen durchgeführt werden kann. Kann man nun zeigen, dass die Zielfunktionswerte der Produkte eine geordnete Menge bilden, lassen sich iterativ die R nutzenmaximalen Produkte erzeugen, wobei R für die Anzahl der in die Produktlinie einzuführenden Produkte darstellt. Diese R nutzenmaximalen Produkte bilden schlussendlich die optimale Produktlinie.

- Es werden für den Genetischen Algorithmus 18 verschiedene Versionen implementiert und für diese eine Monte-Carlo-Simulation durchgeführt, um aus den unterschiedlichen Operatorenkombinationen die geeignetste für die späteren Simulationsrechnungen auszuwählen. Die ausgewählte Kombination mit der besten Performance besteht aus den Operatoren Truncation, 1-Point-Crossover und Emigration, womit **Forschungsfrage 2** beantwortet ist, welche Operatorenkombination zu den besten Ergebnissen führt. Des Weiteren werden die GA-Parameter adaptiv an die zugrunde liegende Problemgröße angepasst.

- Die Performance des GA konnte durch einen neuen Clusteransatz (GAC) für die Konsumentenpräferenzen für die Startpopulation für alle Zielfunktionen entscheidend verbessert werden (siehe Abschnitt 4.3.3). Dabei werden die Konsumentenpräferenzen mittels k-Means-Algorithmus in so viele Segmente geclustert wie Produkte in der Produktlinie existieren. Aus diesen Clustern wird eine bestimmte Anzahl von nutzenmaximalen Produkten mittels des exakten Lösungsalgorithmus aus Abschnitt 4.2.2 berechnet und damit die ansonsten zufällig erzeugte Startpopulation des GAC geimpft.

- Die Particle-Swarm-Optimization ist inhärent ungeeignet, angemessene approximative Lösungen für die Produktlinienoptimierungsprobleme zu finden (siehe Abschnitt 4.4 und 5.3).

- Aufgrund des unzureichenden Modells der Ant-Colony-Optimization für die Produktli-

nienoptimierung von Albritton/McMullen (2007), das erst unnötig aufgebläht und dann wieder auf den schlecht funktionierenden Teil des globalen Updates komprimiert wurde, aber der Ansatz des Reinforcement-Learnings sehr viel versprechend ist, wurde das Min-Max Ant-System in die Produktlinienoptimierung eingeführt. Dieses Modell liefert eine überzeugende Performance und lässt sich mit Hilfe einer lokalen Suche zu MMAS_ls erweitern, was zumindest für die deterministischen Entscheidungsregeln noch besser funktioniert als MMAS ohne lokale Suche.

- Schließlich konnte für Simulated Annealing (SA) ein adaptiver Cooling-Schedule auf Basis von Überlegungen aus der Statistischen Mechanik für die Produktlinienoptimierung eingeführt werden. Die Performance gegenüber dem SA von Belloni et al. (2008b) ist deutlich besser, da die Autoren die Parametereinstellungen und den Cooling-Schedule durch probieren fest eingestellt haben.

- Für die Verfahren GA, GAC, MMAS, MMAS_ls und SA lassen sich die Parameter adaptiv nach der zugrunde liegenden Problemgröße (Anzahl der Eigenschaften und deren Ausprägungen, Anzahl der Produkte in einer Produktlinie) einstellen. Für GA und SA konnte gezeigt werden, dass dieses Vorgehen deutliche Vorteile gegenüber festen Parametereinstellungen besitzt. Für MMAS und MMAS_ls konnte dies trotz fehlender Vergleichsmöglichkeit aufgrund der eindrucksvollen Performance im Vergleich zu den anderen Verfahren bestätigt werden. Somit kann auch **Forschungsfrage 3** positiv beantwortet werden.

In **Kapitel 5** werden die ausgewählten Verfahren aus Kapitel 4 gegeneinander gebenchmarkt. Für die Simulationsstudie werden die Conjointparameter aus 2089 realen Conjointstudien mittels Wahrscheinlichkeitsverteilungen für die Eigenschaften und deren Ausprägungen sowie einer linearen Regressionen für die Anzahl der Konsumenten hergeleitet. Die Teilnutzenwerte werden mittels bewährter Verteilungen erzeugt. Für die Simulationsparameter werden drei Blöcke mit je zehn Datensätzen für drei und neun Produkte in der Produktlinie, also insgesamt 20 Datensätze pro Block, gebildet. Jeder der 20 Datensätze eines Blocks wird 10 mal von allen Verfahren gelöst, was 200 Optimierungen für jedes Verfahren pro Block, und somit insgesamt 600 Optimierungen für jede Zielfunktion, bedeutet. Insgesamt führt also jedes Verfahren 2400 Optimierungen (4 Zielfunktionen $\times$ 200 Datensätze pro Block $\times$ 3 Blöcke) durch, was bei fünf Verfahren 12000 Optimierungen entspricht. Die daraus resultierenden Zielfunktionswerte werden angemessen transformiert, um eine Vergleichbarkeit über ihre arithmetischen Mittel herzustellen. Die Entscheidung, ob ein Verfahren eine bessere Performance im Sinne der Zielfunktionswertqualität als ein anderes bietet, wird immer vor dem Hintergrund des hier verwendeten Versuchsaufbaus getätigt. Der Hintergrund der Blockbildung ist, dass eine möglichst gleiche Verteilung von Problemgrößen gegeben ist, da die Performance der Verfahren stark von der Anzahl möglicher Produktlinien abhängt. Aus dieser umfangreichen Datenbasis ergeben

sich die nachfolgenden Ergebnisse:

- Auf aggregierter Ebene über alle Zielfunktionen hinweg liefert MMAS_ls die besten durchschnittlichen Zielfunktionswerte. GAC mit neuem Clusteransatz erreichte den zweiten Rang, gefolgt von SA auf dem dritten Rang. Den vierten Rang teilen sich GA und MMAS, da es keinen statistisch signifikanten Unterschied auf dem 5 %-Signifikanzniveau. PSO erreicht für alle Simulationen mit Abstand den letzten Rang. Die Antwort auf **Forschungsfrage 4**, welches in dieser Arbeit betrachtete Machine-Learning-Verfahren die beste Performance bietet, wird im Folgenden detailliert beantwortet. Die Performance kommt sowohl auf die Entscheidungsregeln als auch auf die Zielfunktion an. Damit wird auch das „No free lunch theorem“ für Machine-Learning-Verfahren empirisch bestätigt, wonach es unzulässig ist, durch die Performance eines Verfahrens für eine spezielle Zielfunktion automatisch auf die Performance des gleichen Verfahrens auf eine andere Zielfunktion zu schließen.

- Auf Entscheidungsregelebene differenziert sich das Bild etwas aus.

  - Für die deterministische Entscheidungsregeln ist die Reihenfolge: 1. Rang: MMAS_ls, 2. Rang: GAC, 3. Rang: SA, 4. Rang: GA, 5. Rang: MMAS, 6. Rang: PSO.

  - Für die probabilistischen Entscheidungsregeln ergibt sich anderes Bild: 1. Rang: MMAS, 2. Rang: MMAS_ls, 3. Rang: SA, 4. Rang: GA, 5. Rang: GAC, 6. Rang: PSO.

- Auf Zielfunktionsebene ergibt sich ein noch genaueres Bild:

  - Deterministische Gewinnmaximierung: 1. Rang: MMAS_ls, 2. Rang: GAC, 3. Rang: GA, SA, 5. Rang: MMAS, 6. Rang: PSO.

  - Deterministische Marktanteilsmaximierung: 1. Rang: GAC, 2. Rang: MMAS_ls, 3. Rang: MMAS, 4. Rang: SA, GA, 6. Rang: PSO.

  - Probabilistische Gewinnmaximierung: 1. Rang: MMAS, 2. Rang: MMAS_ls, 3. Rang: SA, 4. Rang: GA, 5. Rang: GAC, 6. Rang: PSO.

  - Probabilistische Marktanteilsmaximierung: 1. Rang: MMAS, 2. Rang: MMAS_ls, GAC, 4. Rang: SA, 5. Rang: GA, 6. Rang: PSO.

In **Kapitel 6** wird die Robustheit der HB-Schätzung bzgl. Messfehlern untersucht und ob die Verfahren durch ihre approximative Natur einen Einfluss auf die Robustheit haben. Es wurde ei-

ne reale Conjointstudie verwendet, für die die Teilnutzenwerte mittels HB-Schätzung ermittelt wurde. Die Punktschätzung, die aus 1000 HB-Ziehungen geschätzt wurde, wird mit den Optima für jede Zielfunktion mit den 1000 Ziehungen in der Testphase verglichen. In der Validierungsphase werden die 1000 Optima jeder Zielfunktion in das Modell für die Punktschätzung eingesetzt und die Veränderung auf die arithmetischen Mittel der Zielfunktionswerte untersucht. Ob die Verfahren an sich einen Einfluss auf die Robustheit haben, kann nicht beantwortet werden, da der vorliegende reale Conjointdatensatz zu klein ist und jedes Verfahren für jede der Ziehungen das Optimum fand. In der Test- und Validierungphase ergab sich ein konsistentes Bild, wonach die Zielfunktionen mit probabilistischer Entscheidungsregel weniger sensitiv auf Messfehler in den Daten reagierten als für die Zielfunktionen mit deterministischer Entscheidungsregel. Die **5. Forschungsfrage** wurde somit differenziert beantwortet.

7.2 Kritische Diskussion und Ausblick auf zukünftige Forschung

Das größte Problem, das sich bei einem solchen Verfahrensvergleich stellt, ist die tatsächliche Vergleichbarkeit der Verfahren. Durch die unterschiedliche Funktionsweise der Verfahren, können keine objektiven Kriterien definiert werden, die es möglich machen, eine uneingeschränkte Vergleichbarkeit herzustellen. Definiert man bspw. eine fest vorgegebene Laufzeit für alle Verfahren, stellt sich das Problem, dass einige Verfahren bis zu dieser Laufzeit längst konvergiert haben, andere jedoch nicht. Definiert man eine fest vorgebene Anzahl an Zielfunktionswertevaluationen steht man vor dem gleichen Problem. Des Weiteren ist es zum Teil von der Programmiersprache und von der Art der Implementierung abhängig, wie schnell ein Verfahren läuft. Aus diesen Gründen wurde für diese Arbeit empirisch versucht, dass den Verfahren eine ähnliche Laufzeit zugestanden wird, so dass keine zu große Diskrepanz zwischen den Verfahren entsteht. Das ist prinzipiell gelungen, jedoch ist kritisch anzumerken, dass sich dadurch die Anzahl von Zielfunktionswert- evaluationen unterscheidet. In der Literatur wird dieses Vorgehen zumeist gewählt bzw. sogar noch abgeschwächt und die Verfahren werden bis die Konvergenzkriterien erfüllt sind, laufen gelassen. Damit die Simulationsrechnungen zeitlich kontrollierbar bleiben, wurden in dieser Arbeit problemabhängige, maximale Laufzeiten definiert. Deshalb wird in dieser Arbeit stets von „Performance unter den gegebenen Voraussetzungen“ gesprochen.

Ein weiterer wichtiger Kritikpunkt ist die Verteilung der Teilnutzenwerte der Konsumenten für die Simulationsrechnung. Da es schwierig ist, beliebige Teilnutzenwerte mittels einer Posteriori-Verteilung wie sie die HB-Schätzung besitzt, zu erzeugen, muss für die zukünftige Forschung darauf bestanden werden, eine große Anzahl von realen Conjointdatensätzen mit den Machine-

Learning-Verfahren zu untersuchen, um eventuelle, unerwünschte Effekte auf die Performance der Verfahren durch eine unzureichende Verteilungsannahme entgegenzuwirken. Es könnte sein, dass sich hierdurch die Struktur des Lösungsraums verändert, was zu Performanceverlusten oder -gewinnen führen kann, wie das unterschiedliche Abschneiden mancher Verfahren für die verschiedenen Zielfunktionen zeigte.

Für die Produktlinienoptimierung ist die vorliegende Arbeit die erste, die eine statistische Stringenz für eine Vergleichbarkeit der Verfahren eingeführt hat. Durch diese Stringenz ist es erst möglich eine Rangfolge zwischen den Verfahren festzustellen, auch wenn diese durch den T-Test nicht notwendigerweise transitiv sein muss, was sie hier allerdings ist.

Für die zukünftige Forschung bieten sich spannende Perspektiven. Im Grunde gibt es momentan zwei Alternativen, die die Machine-Learning-Verfahren in der Produktlinienoptimierung obsolet machen könnten. Zum einen ist hier eines der berühmten Millennium-Probleme des Clay Mathematics Institute zu nennen. Wenn $P = NP$ sein sollte, was bisher nicht bewiesen werden konnte, bedeutet dies, dass eine Lösungsstrategie gefunden werden kann, die das Produktlinienoptimierung in polynomieller Zeit lösen kann. Denn dann können die NP-harten Produktlinienoptimierungsprobleme theoretisch in Probleme der Klasse P umgewandelt werden. Da genau dies allerdings noch für kein NP-hartes Problem aus irgendeiner Fachrichtung gelungen ist, ist es aber wohl eher unwahrscheinlich. Eine andere, sich andeutende Revolution ist deutlich wahrscheinlicher. Sollte sich in den nächsten Jahren die Überlegenheit von einsatzfähigen Quantencomputern zeigen, besteht eine große Chance, dass die Produktlinienoptimierungsprobleme als gelöst angesehen werden können, da zu erwarten ist, dass aufgrund der deutlich größeren Rechenpower im Potenzbereich, die exakten Lösungen entweder mit speziellen Algorithmen oder gar mit kompletter Enumeration gelöst werden können.

Ein anderer interessanter Ansatz könnten Deep-Learning-Modelle wie von Bello et al. (2016) sein. Die Autoren lösten verschiedene Probleminstanzen des Problem des Handlungsreisenden (ein anderes NP-hartes Optimierungsmodell) mittels einer Kombination aus Deep-Learning und Reinforcement-Learning. Für das Problem des Handlungsreisenden konnten allerdings keine zufriedenstellenden Ergebnisse im Vergleich zu bestehenden Verfahren erzielt werden. Da sich dieses Forschungsfeld jedoch sehr rasant entwickelt, ist nicht auszuschließen, dass sich daraus interessante Möglichkeiten für die Produktlinienoptimierung ergeben.

Anhang

Siehe nächste Seite.

Nummer	Konsumenten	Eigenschaften	Produkte in Produktlinie	Status-quo-Produkte	Anzahl möglicher Produktlinien	Eigenschaftsausprägungen
1	536	4	3	15	695520	3,3,6,3
2	550	4	3	21	15086400	9,5,2,5
3	550	6	3	24	602800640	2,3,4,4,4,4
4	554	6	3	27	3276855900	3,3,4,5,3,5
5	570	6	3	33	223288501800	5,5,3,7,3,7
6	570	7	3	33	575993985840	2,3,7,4,3,5,6
7	573	8	3	39	13435994894400	4,4,5,5,3,3,3,4
8	580	7	3	39	14293381109700	4,3,5,3,7,7,5
9	536	4	9	126	168899639028120	3,3,6,3
10	541	4	9	135	9774402909134502	7,3,3,4
11	607	8	3	51	160659811878780480	7,6,4,6,7,5,7,4
12	559	3	9	153	366866022765580288	15,5,5
13	550	4	9	162	1923911959887367680	9,5,2,5
14	632	12	3	69	$1.14354824110380 \times 10^{23}$	5,5,7,4,5,5,3,5,3,4,7,4
15	550	6	9	207	$1.28103295884904 \times 10^{23}$	2,3,4,4,4,4
16	554	6	9	225	$2.07354834463652 \times 10^{25}$	3,3,4,5,3,5
17	570	6	9	270	$6.61037171367144 \times 10^{30}$	5,5,3,7,3,7
18	570	7	9	288	$1.13545220835038 \times 10^{32}$	2,3,7,4,3,5,6
19	573	8	9	324	$1.44287514612821 \times 10^{36}$	4,4,5,5,3,3,3,4
20	580	7	9	324	$1.73711961186313 \times 10^{36}$	4,3,5,3,7,7,5
21	587	9	9	378	$1.69484664443314 \times 10^{42}$	3,3,3,3,6,7,3,5,4
22	736	24	3	129	$1.63166530852491 \times 10^{43}$	3,3,4,4,7,4,7,4,5,5,3,4,3,5,5,4,3,3,7,4,3,5,4,3
23	740	24	3	129	$1.67113939108139 \times 10^{43}$	3,4,4,7,7,3,3,4,3,4,3,4,7,6,5,3,7,4,2,4,3,3,6,5
24	593	9	9	396	$4.66408999018536 \times 10^{44}$	3,3,3,7,7,6,4,4,3
25	770	22	3	132	$6.73984823390349 \times 10^{44}$	4,5,15,4,7,7,5,6,6,7,3,5,4,4,4,4,4,5,7,5,2,4
26	779	25	3	144	$1.39941604282451 \times 10^{48}$	5,4,7,4,4,3,3,4,2,3,7,6,7,5,5,10,6,3,7,7,3,4,3,6,3
27	607	8	9	432	$2.46831515798289 \times 10^{48}$	7,6,4,6,7,5,7,4
28	605	9	9	441	$3.94292714637912 \times 10^{49}$	4,5,5,5,8,4,7,4,3
29	642	8	9	450	$2.44932089604688 \times 10^{50}$	25,3,7,4,7,4,7,4
30	603	10	9	450	$3.15613854497493 \times 10^{50}$	3,4,4,3,5,7,4,3,4,7
31	786	27	3	150	$5.74809106434901 \times 10^{50}$	7,3,4,4,7,7,4,3,4,4,5,3,4,3,4,4,4,3,7,6,3,4,7,7,2,4,7
32	823	27	3	153	$1.45354934568203 \times 10^{51}$	5,3,3,2,5,5,4,3,3,5,2,7,4,25,3,7,4,7,7,5,7,3,2,5,7,3,4
33	825	27	3	153	$4.83456387711089 \times 10^{51}$	5,3,4,4,5,2,3,2,4,3,4,6,5,25,5,5,7,7,4,7,3,3,4,6,5,8,2
34	816	27	3	162	$2.57249840659782 \times 10^{54}$	3,5,3,5,5,7,9,7,3,5,4,6,7,4,3,7,3,6,3,4,5,7,7,7,4,4,4
35	614	11	9	486	$4.82106859730598 \times 10^{54}$	7,2,3,5,4,7,2,4,3,5,7
36	623	11	9	540	$1.26382111500137 \times 10^{60}$	7,4,4,3,5,5,7,4,7,3,4
37	632	12	9	585	$8.90131266164932 \times 10^{65}$	5,5,7,4,5,5,3,5,3,4,7,4
38	899	36	3	198	$7.67184527921986 \times 10^{66}$	5,5,4,4,4,6,5,3,2,3,5,6,5,3,3,4,25,3,4,5,4,4,3,6,7,3,5,7,7,4,3,3,2,3,4,4
39	880	36	3	204	$1.07179483000299 \times 10^{68}$	4,4,3,4,3,4,9,4,3,4,5,4,3,2,7,3,4,6,3,3,6,7,7,4,7,5,4,5,6,3,4,4,4,6,4,7
40	651	13	9	630	$9.51586480851509 \times 10^{70}$	4,2,3,11,5,7,5,4,4,4,4,3,9
41	658	15	9	729	$1.20399689244990 \times 10^{81}$	5,5,4,4,5,4,5,5,4,7,4,3,7,3,3
42	676	17	9	765	$4.73729533302235 \times 10^{85}$	3,5,2,7,7,3,3,4,4,3,3,3,2,4,10,10,3
43	692	16	9	792	$5.43329163019961 \times 10^{88}$	5,3,7,15,4,7,2,3,5,5,2,6,3,7,3,6
44	1030	51	3	282	$2.38965673511264 \times 10^{94}$	3,3,4,3,3,2,4,4,4,7,4,3,7,3,5,5,7,5,7,4,5,6,7,4,11,4,3,3,4,4,4,4,4,5,2,6,10,5,7,2,5,3,7,3,5,3,4,3,3,4,3
45	1076	51	3	288	$1.14969424728821 \times 10^{96}$	6,3,2,7,3,25,2,3,7,3,4,3,4,7,3,4,4,7,12,2,3,7,3,7,3,6,3,4,4,6,4,5,3,3,4,4,5,4,3,3,4,3,4,7,3,7,4,7,7,3,6
46	1048	51	3	288	$1.23881217950845 \times 10^{96}$	4,3,2,5,6,4,7,4,7,3,5,4,4,3,3,3,2,5,4,5,5,3,7,4,5,3,3,3,4,4,4,5,7,7,3,3,5,15,7,4,4,10,3,5,4,3,5,3,3,7,7
47	1085	51	3	291	$1.21399519071490 \times 10^{97}$	6,3,4,4,3,5,4,2,3,3,7,5,5,4,5,25,6,3,4,7,4,7,4,4,3,15,4,7,3,7,4,4,7,8,4,3,2,4,4,3,4,3,4,4,3,3,4,3,4,3,10
48	1048	51	3	291	$1.46153947618168 \times 10^{97}$	7,3,4,3,4,7,8,4,4,3,3,3,4,5,7,7,7,7,7,5,3,3,4,6,7,4,3,3,4,7,5,3,4,7,3,4,2,4,2,5,4,4,5,5,3,2,5,7,7,5,6
49	1083	51	3	297	$2.72366474302846 \times 10^{99}$	3,4,5,4,7,6,4,15,4,3,5,3,5,3,5,4,4,4,4,4,7,2,5,5,2,2,3,7,4,6,2,7,5,7,4,6,7,6,5,4,6,6,4,5,4,7,3,4,4,15,3
50	1110	51	3	297	$5.41814097139786 \times 10^{99}$	4,25,3,3,6,3,4,7,4,7,3,7,7,2,4,3,4,4,15,3,2,7,5,7,4,4,4,4,7,3,5,7,3,6,5,3,5,4,4,4,3,3,7,5,11,4,4,7,2,4,4
51	731	17	9	900	$1.79320229540845 \times 10^{100}$	3,3,3,4,4,7,9,4,5,4,4,25,5,5,3,7,5
52	1076	51	3	300	$1.91233841068388 \times 10^{100}$	3,4,3,7,5,5,4,3,3,7,4,4,5,7,3,7,5,5,5,3,4,7,7,4,3,7,5,7,3,4,7,5,7,4,4,3,7,7,4,3,3,4,4,4,4,3,3,5,4,15,7
53	1092	51	3	300	$2.48890335954881 \times 10^{100}$	3,3,4,3,3,7,4,5,7,5,4,12,2,4,2,5,7,3,7,4,7,4,5,14,4,7,4,5,2,8,5,4,5,6,4,3,3,4,6,4,4,4,9,10,7,3,4,4,3,7,4
54	1073	51	3	300	$9.96618949730298 \times 10^{100}$	7,4,4,4,5,7,7,6,4,5,7,4,7,7,10,5,7,6,4,4,2,4,7,4,7,7,4,3,5,3,5,4,4,6,5,7,4,4,3,4,4,3,4,3,3,5,7,4,3,4,3
55	715	20	9	990	$3.98039705490374 \times 10^{110}$	7,5,6,7,6,4,4,3,3,4,7,7,3,3,4,5,4,5,3,3
56	727	20	9	1008	$1.27579944523437 \times 10^{112}$	3,3,3,7,2,9,3,6,3,7,4,4,7,3,4,4,7,7,5,7
57	770	22	9	1179	$1.82238893664965 \times 10^{131}$	4,5,15,4,7,7,5,6,6,7,3,5,4,4,4,4,4,5,7,5,2,4
58	825	27	9	1359	$6.72608926220802 \times 10^{151}$	5,3,4,4,5,2,3,2,4,3,4,6,5,25,5,5,7,7,4,7,3,3,4,6,5,8,2
59	816	27	9	1440	$1.01334204279184 \times 10^{160}$	3,5,3,5,5,7,9,7,3,5,4,6,7,4,3,7,3,6,3,4,5,7,7,7,4,4,4
60	880	36	9	1800	$7.32867886389561 \times 10^{200}$	4,4,3,4,3,4,9,4,3,4,5,4,3,2,7,3,4,6,3,3,6,7,7,4,7,5,4,5,6,3,4,4,4,6,4,7

Tabelle 32: Simulierte Conjointdatensätze der Simulationsstudie aus Kapitel 5 aufsteigend sortiert nach der Anzahl der möglichen Produktlinien. (Quelle: Eigene Darstellung)

Literaturverzeichnis

Aarts, E.; Korst, J. (1989): Simulated Annealing and Boltzmann Machines, Wiley.

Aarts, E.; Korst, J.; van Laarhoven, J. (1988): A Quantitative Analysis of the Simulated Annealing Algorithm: A Case Study for the Traveling Salesman Problem, in: Journal of Statistical Physics, 50 (1/2), 187–206.

Aarts, E.; Lenstra, J. K. (1997): Introduction to Local Search in Combinatorial Optimization, in: Aarts, E. H.; Lenstra, J. K. (Hrsg.): Local Search in Combinatorial Optimization, John Wiley & Sons, 3–22.

Aarts, E. H.; Van Laarhoven, P. J. (1985): Statistical cooling: A general approach to combinatorial optimization problems, in: Philips Journal of Research, 40, 193–226.

Abbass, H. (2001): MBO: Marriage in Honey Bees Optimization – A Haplometrosis Polygynous Swarming Approach, in: Evolutionary Computation, IEEE, 207–214.

Abedinia, O.; Amjady, N.; Ghasemi, A. (2016): A new metaheuristic algorithm based on shark smell optimization, in: Complexity, 21 (5), 97–116.

Albers, S. (1979): An Extended Algorithm for Optimal Product Positioning, in: European Journal of Operational Research, 3, 222–231.

Albers, S.; Brockhoff, K. (1977): A Procedure for New Positioning in an Attribute Space, in: European Journal of Operational Research, 1, 230–238.

Albritton, M. D.; McMullen, P. R. (2007): Optimal product design using a colony of virtual ants, in: European Journal of Operational Research, 176 (1), 498–520.

Alexouda, G.; Paparizzos, K. (2001): A genetic algorithm approach to the product line design problem using the seller's return criterion: An extensive comparative computional study, in: European Journal of Operational Research, 134, 165–178.

Allenby, . R. P. E., G. M. (1998): Marketing models of consumer heterogeneity, in: Journal of Econometrics, 89 (1-2), 57–78.

Allenby, G.; Arora, N.; Ginter, J. (1995): Incorporating Prior Knowledge into the Analysis of Conjoint Studies, in: Journal of Marketing Research, 23, 152–162.

Allenby, G.; Ginter, J. (1995): Using Extremes to Design Products and Segment Markets, in: Journal of Marketing Research, 32, 392–403.

Atashpaz-Gargari, E.; Lucas, C. (2007): Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, in: Evolutionary computation CEC, IEEE, 4661–4667.

Aust, E. (1995): Simultane Conjointanalyse, Benefitsegmentierung, Produktlinien- und Preisgestaltung, Dissertation, Universität Karlsruhe.

Aydin, G.; Ryan, J. K. (2000): Product Line Selection and Pricing Under the Multinominal Logit Choice Model, in: Proceedings of the 2000 MSOM Conference, 1–49.

Aydin, R.; Kwong, C.; Ji, P. (2015): Coordination of the closed-loop supply chain for product line design with consideration of remanufactured products, in: Journal of Cleaner Production, 114, 286–298.

Baier, D. (2014): Bayesian Methods for Conjoint Analysis-Based Predictions: Do We Still Need Latent Classes?, in: German-Japanese Interchange of Data Analysis Results. Springer, Cham, 103–113.

Baier, D.; Gaul, W. (1999): Optimal Product Positioning Based on Paired Comparison Data, in: Journal of Econometrics, 89 (1-2), 365–392.

Baker, J. E. (1985): Adaptive selection methods for genetic algorithms, in: Proceedings of the 1st International Conference on Genetic Algorithms, 101–111.

Baker, J. E. (1987): Reducing Bias and Inefficiency in the Selection Algorithm, in: Proceedings of the 2nd International Conference on Genetic Algorithms, 206, 14–21.

Balakrishnan, P. V.; Gupta, R.; Jacob, V. S. (2004): Development of Hybrid Genetic Algorithms for Product Line Designs, in: IEEE Trans. Systems, Man, Cybernetics, 34 (1), 468–483.

Balakrishnan, P. V.; Jacob, V. S. (1995): Triangulation in decision support systems: algorithms for product design, in: Decision Support Systems, 14, 313–327.

Balakrishnan, P. V.; Jacob, V. S. (1996): Genetic Algorithms for Product Design, in: Management Science, 42 (8), 1105–1117.

Bastos Filho, C.; de Lima Neto, F.; Lins, A.; Nascimento, A.; Lima, M. (2008): A novel search algorithm based on fish school behavior, in: Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on, IEEE, 2646–2651.

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; Bengio, S. (2016): Neural Combinatorial Optimization with Reinforcement Learning, in: CoRR, abs/1611.09940.

Belloni, A.; Freund, R.; Selove, M.; Simester, D. (2008a**):** Appendix: Electronic Companion – "Optimizing Product Line Designs: Efficient Methods and Comparisons", in: Management Science, 1–10.

Belloni, A.; Freund, R.; Selove, M.; Simester, D. (2008b**):** Optimizing Product Line Designs: Efficient Methods and Comparisons, in: Management Science, 54 (9), 1544–1552.

Bertsimas, D.; Misic, V. (2019): Exact First-Choice Product Line Optimization, in: Operations Research, 67, 651–670.

Brander, J.; Eaton, J. (1984): Product Line Rivalry, in: The American Economic Review, 74, 323–335.

Böckenholt, I. (1989): Mehrdimensionale Skalierung qualitativer Daten - Ein Instrument zur Unterstützung von Marketingentscheidungen, Peter Lang, Frankfurt am Main.

Camm, J. D.; Cochran, J. J.; Curry, D. J.; Kannan, S. (2006): Conjoint Optimization: An Exact Branch-and-Bound Algorithm for the Share-of-Choice Problem, in: Management Science, 13, 435–447.

Cao, J.; Luo, X. G.; Kwong, C. K.; Tang, J. F.; Zhou, W. (2012): Joint optimization of product family design and supplier selection under multinomial logit consumer choice rule, in: Concurrent Engineering: Research and Applications, (20), 335 – 347.

Carpenter, G. (1989): Perceptional Position and Competitive Brand Strategy in an Two-Dimensional, Two-Brand Market, in: Management Science, 35, 1029–1044.

Cattin, P.; Wittink, D. (1982): Commercial Use of Conjoint Analysis: A Survey, in: Journal of Marketing, 46, 44–53.

Chen, H.; Zhu, Y.; Hu, K.; He, X. (2010): Hierarchical swarm model: a new approach to optimization, in: Discrete Dynamics in Nature and Society, 1, 1–31.

Chen, J.; Cai, H.; Wang, W. (2018): A new metaheuristic algorithm: car tracking optimization algorithm, in: Soft Computing, 22 (12), 3857–3878.

Chen, K. D.; Hausman, W. H. (2000): Technical Note: Mathematical Properties of the Optimal Product Line Selection Problem Using Choice-Based Conjoint Analysis, in: Management Science, 46 (2), 327–332.

Chen, T.-C.; Tsai, P.-W.; Chu, S.-C.; Pan, J.-S. (2007): A novel optimization approach: bacterial-ga foraging, in: Innovative Computing, Information and Control, IEEE, 391–391.

Cheng, M.-Y.; Prayogo, D. (2014): Symbiotic organisms search: a new metaheuristic optimization algorithm, in: Computers & Structures, 139, 98–112.

Choi, C. S.; DeSarbo, W. S. (1993): Game Theoretic Derivations of Competitive Strategies in Conjoint Analysis, in: Marketing Letters, 4 (4), 337–348.

Choi, C. S.; DeSarbo, W. S. (1994): A Conjoint-based Product Designing Procedure Incorporating Price Competition, in: Journal of Product Innovation Management, 11 (5), 451–459.

Chu, S.-C.; Tsai, P.-W.; Pan, J.-S. (2006): Cat swarm optimization, in: Pacific Rim International Conference on Artificial Intelligence, Springer, 854–858.

Chu, Y.; Mi, H.; Liao, H.; Ji, Z.; Wu, Q. H. (2008): A fast bacterial swarming algorithm for high-dimensional function optimization, in: Evolutionary Computation, IEEE World Congress on Computational Intelligence, IEEE, 3135–3140.

Church, R.; ReVelle, C. (1974): The maximal covering location problem, in: Papers Regional Sci. Assoc., (23), 101–118.

Civicioglu, P. (2012): Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm, in: Computers & Geosciences, 46, 229–247.

Civicioglu, P. (2013a**):** Artificial cooperative search algorithm for numerical optimization problems, in: Information Sciences, 229, 58–76.

Civicioglu, P. (2013b**):** Backtracking search optimization algorithm for numerical optimization problems, in: Applied Mathematics and Computation, 219 (15), 8121–8144.

Clerc, M. (1999): The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, in: IEEE Congress on Evolutionary Computation, 1951–1957.

Clerc, M.; Kennedy, J. (2002): The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space, in: IEEE Transactions on Evolutionary Computation, 6 (1), 58–73.

Colorni, A.; Dorigo, M.; Maniezzo, V. (1991): Distributed Optimization by Ant Colonies, in: actes de la première conférence européenne sur la vie artificielle, 134–142.

Comellas, F.; Martinez-Navarro, J. (2009): Bumblebees: a multiagent combinatorial optimization algorithm inspired by social insect behaviour, in: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, ACM, 811–814.

Cuevas, E.; Oliva, D.; Zaldivar, D.; Pérez-Cisneros, M.; Sossa, H. (2012): Circle detection using electro-magnetism optimization, in: Information Sciences, 182 (1), 40–55.

Day, R. (1968): Preference Tests and the Management of Product Features, in: Journal of Marketing, 32 (3), 24–29.

De Jong, K. (1975): Analysis of the behavior of a class of genetic adaptive systems, Dissertation.

Dean, J. (1950): Problems of Product-Line Pricing, in: Journal of Marketing, 28 (6), 518–528.

Debreu, G. (1960): Review of RD Luce, Individual choice behavior: A theoretical analysis, in: American Economic Review, 50 (1), 186–188.

Dobsen, G.; Kalish, S. (1988): Positioning and Pricing a Product Line, in: Marketing Science, 7 (2), 107–125.

Dobsen, G.; Kalish, S. (1993): Heuristics for Pricing and Positioning a Product-line using Conjoint and Cost Data, in: Management Science, 39 (2), 160–175.

Dorigo, M. (1992): Optimization, Learning and Natural Algorithms, Politecnico di Milano.

Dorigo, M.; Di Caro, G.; Gambardella, L. (1999): Ant algorithms for discrete optimization, in: Artificial life, 5 (2), 137–172.

Dorigo, M.; Stützle, T. (2003): The ant colony optimization metaheuristic: Algorithms, applications, and advances, in: Handbook of metaheuristics, Springer, Boston, MA, 250–285.

Dorigo, M.; Stützle, T. (2010): Ant Colony Optimization: Overview and Recent Advances, Handbook of metaheuristics, Springer, 227–263.

Drias, H.; Sadeg, S.; Yahi, S. (2005): Cooperative bees swarm for solving the maximum weighted satisfiability problem, in: Computational intelligence and bioinspired systems, 417–448.

Duman, E.; Uysal, M.; Alkaya, A. F. (2012): Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem, in: Information Sciences, 217, 65–77.

Dutta, S.; Biswal, M.; Mishra, R. (2018): Fuzzy stochastic price scenario based portfolio selection and its application to BSE using genetic algorithm, in: Applied Soft Computing, 62, 867–891.

Eberhart, R.; Shi, Y. (2000): Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization, in: Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, 1, 84–88.

Eberhart, R.; Shi, Y. (2001): Particle Swarm Optimization: Developments, Applications ans Resources, in: Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, 2, 81–86.

Elrod, T.; Kumar, K. (1989): Bias in the first choice rule for predicting share, in: Sawtooth Software Conference Proceedings.

Engelbrecht, A. (2007): Computational Intelligence: An Introduction, John Wiley & Sons.

Eskandar, H.; Sadollah, A.; Bahreininejad, A.; Hamdi, M. (2012): Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems, in: Computers & Structures, 110, 151–166.

Fadakar, E.; Ebrahimi, M. (2016): A new metaheuristic football game inspired algorithm, in: 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC), IEEE, 6–11.

Fausto, F.; Reyna-Orta, A.; Cuevas, E.; Andrade, Á. G.; Perez-Cisneros, M. (2020): From ants to whales: metaheuristics for all tastes, in: Artificial Intelligence Review, 53 (1), 753–810.

Ferreira, C. (2001): Gene expression programming: A new adaptive algorithm for solving problems, in: Complex Systems, 13 (2), 87–129.

Formato, R. A. (2007): Central force optimization: a new metaheuristic with applications in applied electromagnetics, in: Progress In Electromagnetics Research, 77, 425–491.

Fruchter, G.; Fligler, A.; Winer, R. (2006): Optimal Product Line Design: Genetic Algorithm Approach to Mitigate Cannibalization, in: Journal of Optimization Theory and Applications, 131(2), 227–244.

Gambardella, L.; Dorigo, M. (1995): Ant-Q: A reinforcement learning approach to the traveling salesman problem, in: , 252–260.

Gambardella, L.; Dorigo, M. (1996): Solving symmetric and asymmetric TSPs by ant colonies, in: Proceedings of the IEEE International Conference on Evolutionary Computation, 622–627.

Gandomi, A. H.; Alavi, A. H. (2012): Krill herd: a new bio-inspired optimization algorithm, in: Communications in Nonlinear Science and Numerical Simulation, 17 (12), 4831–4845.

Gaul, W.; Aust, E.; Baier, D. (1995): Gewinnorientierte Produktliniengestaltung unter Berücksichtigung des Kundennutzens, in: Zeitschrift für Betriebswirtschaftslehre, 65, 835–855.

Gaul, W.; Baier, D. (2009): Simulations- und Optimierungsrechnungen auf Basis der Conjointanalyse, in: Conjointanalyse: Methoden – Anwendungen – Praxisbeispiele, Springer-Verlag Berlin Heidelberg, 163–182.

Geem, Z. W.; Kim, J. H.; Loganathan, G. V. (2001): A new heuristic optimization algorithm: harmony search, in: Simulation, 76 (2), 60–68.

Ghasemalizadeh, O.; Khaleghian, M.; Taheri, S. (2016): A Review of Optimization Techniques in Artificial Networks, in: International Journal of Advanced Research, 4, 1668–1686.

Goldberg, D. E. (1989): Genetic algorithms in search, optimization, and machine learning, Addison-Wesley Publishing Company, Inc.

Goldberg, D. E.; Deb, K. (1991): A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, in: Foundations of Genetic Algorithms, 1, 69–93.

Green, P. E.; Carroll, J. D.; Goldberg, S. M. (1981): A General Approach to Product Design Optimization via Conjoint Analysis, in: Journal of Marketing Research, 45 (3), 17–37.

Green, P. E.; Krieger, A. M. (1985): Models and Heuristics for Product Line Selection, in: Marketing Science, 4 (1), 1–19.

Green, P. E.; Krieger, A. M. (1987): A Consumer-Based Approach to Designing Product Line Extensions, in: Journal of Product Innovation Management, 4 (1), 21–32.

Green, P. E.; Krieger, A. M. (1988): Choice Rules and Sensitivity Analysis in Conjoint Simulators, in: Journal of the Academy of Marketing Science, 16 (1), 114–127.

Green, P. E.; Krieger, A. M. (1989): Recent Contributions to Optimal Product Positionig and Buyer Segmentation, in: European Journal of Operational Research, 4 (1), 127–141.

Green, P. E.; Krieger, A. M. (1992): An Application of a Product Positioning Model to Pharmaceutical Products, in: Marketing Science, 2 (1), 117–132.

Green, P. E.; Krieger, A. M. (1993): Conjoint analysis with product-positioning applications, in: Handbooks in Operations Research & Management Science, 5 (1), 467–515.

Green, P. E.; Krieger, A. M.; Zelnio, R. N. (1989): A Componential Segmentation Model with Optimal Product Design Features, in: Decision Sciences, 20 (2), 221–238.

Green, P. E.; Rao, V. R. (1969): Nonmetric Approaches to Multivariate Analysis in Marketing, in: Wharton School, University of Pensnsylvania, Working Paper.

Green, P. E.; Rao, V. R. (1972): Applied Multidimensional Scaling: A Comparison of Approaches and Algorithms, Holt, Rinehart and Winston.

Gutsche, J. (1995): Produktpräferenzanalyse – Ein modelltheoretisches und methodisches Konzept zur Marktsimulation mittels Präferenzerfassungsmodellen, Dissertation, Universität Mannheim.

Harifi, S.; Khalilian, M.; Mohammadzadeh, J.; Ebrahimnejad, S. (2019): Emperor Penguins Colony: a new metaheuristic algorithm for optimization, in: Evolutionary Intelligence, 12 (2), 211–226.

Hatamlou, A. (2013): Black hole: A new heuristic optimization approach for data clustering, in: Information sciences, 222, 175–184.

Hauser, J. R. (2011): Comment: New developments in product-line optimization, in: International Journal of Research in Marketing, 28, 26–27.

Havens, T.; Spain, C.; Salmon, N.; Keller, J. (2008): Roach infestation optimization, in: Swarm Intelligence Symposium, IEEE, 1–7.

He, S.; Wu, Q. H.; Saunders, J. (2009): Group search optimizer: an optimization algorithm inspired by animal searching behavior, in: IEEE transactions on evolutionary computation, 13 (5), 973–990.

Hedayatzadeh, R.; Salmassi, F.; Keshtgari, M.; Akbari, R.; Ziarati, K. (2010): Termite colony optimization: A novel approach for optimizing continuous problems, in: Electrical Engineering (ICEE), IEEE, 553–558.

Hermann, S. (2006): Zum Messen, Managen und Monitoren der Consumer Experience, in: marketingjournal, 11, 8–13.

Holland, J. H. (1975): Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence, Oxford: U Michigan Press.

Holland, J. H. (1992): Genetic Algorithms, in: Scientific american, 267 (1), 66–72.

Hosseini, H. (2007): Problem solving by intelligent water drops, in: Evolutionary Computation, IEEE, 3226–3231.

Hotelling, H. (1929): Stability in Competition, in: The Economic Journal, 39 (153), 41–57.

Iordache, S. (2010): Consultant-guided search: a new metaheuristic for combinatorial optimization problems, in: Proceedings of the 12th annual conference on Genetic and evolutionary computation, ACM, 225–232.

Jung, S. (2003): Queen-bee evolution for genetic algorithms, in: Electronics letters, 39 (6), 575–576.

Kallioras, N. A.; Lagaros, N. D.; Avtzis, D. N. (2018): Pity beetle algorithm – a new metaheuristic inspired by the behavior of bark beetles, in: Advances in Engineering Software, 121, 147–166.

Karaboga, D.; Basturk, B. (2007): A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm, in: Journal of global optimization, 39 (3), 459–471.

Karegowda, A. G.; Manjunath, A. S.; Jayaram, M. A. (2011): Application of genetic algorithm optimized neural network connection weights for medical diagnosis of pima Indians diabetes, in: International Journal on Soft Computing, 2 (2), 15–23.

Kashan, A. H. (2009): League championship algorithm: a new algorithm for numerical function optimization, in: Soft Computing and Pattern Recognition, IEEE, 43–48.

Kashan, A. H. (2015): A new metaheuristic for optimization: optics inspired optimization (OIO), in: Computers & Operations Research, 55, 99–125.

Kaul, A.; Rao, V. R. (1995): Research for Product Positioning and Design Decisions: An integrative Review, in: International Journal of Research in Marketing, 12 (4), 293–320.

Kaveh, A.; Bakhshpoori, T. (2016): A new metaheuristic for continuous structural optimization: Water evaporation optimization, in: Structural and Multidisciplinary Optimization, 54 (1), 23–43.

Kaveh, A.; Farhoudi, N. (2013): A new optimization method: dolphin echolocation, in: Advances in Engineering Software, 59, 53–70.

Kaveh, A.; Talatahari, S. (2010): A novel heuristic optimization method: charged system search, in: Acta Mechanica, 213 (3), 267–289.

Kekre, S.; Srinivasan, K. (1990): Broader Product Line: A Necessity to Achieve Success?, in: Management Science, 36 (10), 1216–1231.

Kennedy, J.; Eberhart, R. (1995a**):** Particle swarm optimization., in: Neural Networks, IEEE, 1942–1948.

Kennedy, J.; Eberhart, R. (1995b**):** Particle Swarm Optimization, in: Proceedings of IEEE International Conference on Neural Networks IV, 1942–1948.

Kennedy, J.; Eberhart, R. C. (1997): A discrete binary version of the particle swarm algorithm, in: IEEE International Conference on Systems, Man, and Cybernetics, 4104–4108.

Kirkpatrick, S.; Gelatt Jr, C. D.; Vecchi, M. P. (1983): Optimization by Simulated Annealing, in: Science, 220 (4598), 671–680.

Kohli, R.; Krishnamurti, R. (1987): A Heuristic Approach to Product Design, in: Management Science, 33, 1523–1533.

Kohli, R.; Krishnamurti, R. (1989): Optimal Product Design Using Conjoint Analysis: Computational Complexity and Algorithms, in: Journal of Operational Research, 40, 186–195.

Kohli, R.; Sukumar, R. (1990): Heuristics for Product-line Design Using Conjoint Analysis, in: Management Science, 36, 1464–1478.

Kotler, P.; Armstrong, G. (2010): Principles of Marketing, Pearson Education, Upper Saddle River, New Jersey.

Kotler, P.; Bliemel, F. (1999): Marketing-Management, Schäffer-Poeschel Verlag, Stuttgart.

Lamy, J.-B. (2019): Artificial Feeding Birds (AFB): a new metaheuristic inspired by the behavior of pigeons, in: Advances in nature-inspired computing and applications, Springer, 43–60.

Lehmann, D. (1971): Television Show Preference: Application of a Choice Model, in: Journal of Marketing Research, 8 (1), 47–55.

Lenk, P. J.; DeSarbo, W. S.; Green, P. E.; Young, M. R. (1996): Hierarchical Bayes Conjoint Analysis: Recovery of Part-Worth Heterogeniety from Reduced Experimental Design, in: Marketing Science, 15 (2), 173–191.

Li, S.; Chen, H.; Wang, M.; Heidari, A. A.; Mirjalili, S. (2020): Slime mould algorithm: A new method for stochastic optimization, in: Future Generation Computer Systems, 111, 300–323.

Liberali, G. (2011): Comment: Product line design optimization, in: International Journal of Research in Marketing, 28 (1), 28–29.

Louviere, J. (1988): Conjoint analysis modelling of stated preferences, in: Journal of Transport Economics and Policy, 22 (1), 93–119.

Louviere, J.; Woodworth, G. (1983): Design and Analysis of Simulated Consumer Choice or

Allocation Experiments: An Approach based on Aggregate Data, in: Journal of Marketing Research, 20 (4), 350–367.

Lowerre, B. (1976): The Harpy Speech Recognition System, Dissertation, Carnegie Mellon University, Pittsburgh.

Luce, R. (1959): Individual Choice Behavior, New York: John Wiley and Sons.

Luce, R.; Tuckey, J. (1964): Simultaneous Conjoint Measurement. A New Type of Fundamental Measuring, in: Journal of Mathematical Psychology, 1, 1–27.

Luo, L. (2011): Product Line Design for Consumer Durables: An Integrated Marketing and Engineering Approach, in: Journal of Marketing Research, 48 (1), 128–139.

Maia, R.; de Castro, L.; Caminhas, W. (2012): Bee colonies as model for multimodal continuous optimization: The optbees algorithm, in: Evolutionary Computation (CEC), IEEE, 1–8.

Mehrabian, A.; Lucas, C. (2006): A novel numerical optimization algorithm inspired from weed colonization, in: Ecological informatics, 1 (4), 355–366.

Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; Teller, E. (1953): Equation of state calculations by fast computing machines, in: The Journal of Chemical Physics, 21 (6), 1087–1092.

Michalek, J. J.; Ebbes, A. F., Peter; Feinberg, F. M.; Papalambros, P. Y. (2011): Enhancing marketing with engineering: Optimal product line design for heterogeneous markets, in: International Journal of Research in Marketing, 28 (1), 1–12.

Michalek, J. J.; Feinberg, F. M.; Papalambros, P. Y. (2005): Linking Marketing and Engineering Product Design Decisions via Analytical Target Cascading, in: Journal of Product Innovation Management, 22 (1), 42–62.

Mirjalili, S.; Mirjalili, S.; Hatamlou, A. (2015): Multi-verse optimizer: a nature-inspired algorithm for global optimization, in: Neural Computing and Applications, 1–19.

Mirjalili, S.; Mirjalili, S. M.; Lewis, A. (2014): Grey wolf optimizer, in: Advances in engineering software, 69, 46–61.

Moorthy, K. S. (1984): Market Segmentation, Self-Selection, and Product Line Design, in: Marketing Science, 3 (4), 288–307.

Moorthy, K. S. (1987): Product Line Competition, in: Annales des Telecommunication, 42, 655–663.

Moorthy, K. S. (1988): Product and Price Competition in a Duopoly, in: Marketing Science, 7 (2), 141–169.

Moorthy, K. S. (1993): Competitive Market Strategies: Game-Theoretic Models, in: Eliashberg, J., Lilien, G.L. (Eds.): Marketing, Handbooks in Operations Research.

Moradi, M. H.; Abedini, M. (2012): A combination of genetic algorithm and particle swarm optimization for optimal DG location and sizing in distribution systems., in: International Journal of Electrical Power & Energy Systems, 34 (1), 66–74.

Murray, G., JR.; Wolfe, H. (1970): Length of Product Line, in: California Management Review, 12, 79–85.

Mussa, E.; Rosen, S. (1978): Monopoly and Product Quality, in: Journal of Economic Theory, 18, 301–317.

Mühlenbein, H. (1997): Genetic algorithms, in: Aarts, E. H.; Lenstra, J. K. (Hrsg.): Local Search in Combinatorial Optimization, John Wiley & Sons, 137–172.

Nair, S. K.; Thakur, L. S.; Wen, K.-W. (1995): Near Optimal Solutions for Product Line Design and Selection: Beam Search Heuristics, in: Management Science, 41 (5), 767–785.

Parpinelli, R.; Lopes, H. (2011): An eco-inspired evolutionary algorithm applied to numerical optimization, in: Nature and Biologically Inspired Computing (NaBIC), IEEE, 466–471.

Passino, K. (2002): Biomimicry of bacterial foraging for distributed optimization and control, in: IEEE control systems, 22 (3), 52–67.

Pham, D.; Ghanbarzadeh, A.; Koc, E.; Otri, S.; Rahim, S.; Zaidi, M. (2006): The bees algorithm – A novel tool for complex optimisation, in: 2nd Virtual International Conference on Intelligent Production Machines and Systems.

Pierezan, J.; Coelho, L. D. S. (2018): Coyote optimization algorithm: a new metaheuristic for

global optimization problems, in: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE, 1–8.

Premaratne, U.; Samarabandu, J.; Sidhu, T. (2009): A new biologically inspired optimization algorithm, in: Industrial and Information Systems (ICIIS), IEEE, 279–284.

Rabanal, P.; Rodríguez, I.; Rubio, F. (2007): Using river formation dynamics to design heuristic algorithms, in: International Conference on Unconventional Computation, Springer, 163–177.

Raman, N.; Chhajed, D. (1995): Simultaneous determination of product aattribute and prices, and Pproducti prozess in product-line design, in: Journal of Operations Management, 12, 187–204.

Raouf, O. A.; Hezam, I. M. (2017): Sperm motility algorithm: a novel metaheuristic approach for global optimisation, in: International Journal of Operational Research, 28 (2), 143–163.

Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. (2009): Gsa: a gravitational search algorithm, in: Information sciences, 179 (13), 2232–2248.

Reddy K, S.; Panwar, L.; Panigrahi, B.; Kumar, R. (2019): Binary whale optimization algorithm: a new metaheuristic approach for profit-based unit commitment problems in competitive electricity markets, in: Engineering Optimization, 51 (3), 369–389.

Reeves, C. (2003): Genetic Algorithms, in: Glover, F.; Kochenberger, G. (Hrsg.): Handbook of Metaheuristics, Kluwer Academic Publishers, International Series in Operations Research & Management Science.

Roberge, V.; Tarbouchi, M.; Labonte, G. (2013): Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning, in: IEEE Transactions on Industrial Informatics, 9 (1), 132–141.

Rubin, S. (1978): The Argos Image Understanding System, Dissertation, Carnegie Mellon University, Pittsburgh.

Ryan, C.; Collins, J.; Neill, M. O. (1998): Grammatical evolution: Evolving programs for an arbitrary language, in: European Conference on Genetic Programming, Springer, 83–96.

Sadeghi, A.; Alem-Tabriz, A.; Zandieh, M. (2011): Product portfolio planning: a

metaheuristic-based simulated annealing algorithm, in: International Journal of Production Research, 49, 2327–2350.

Salimi, H. (2015): Stochastic fractal search: a powerful metaheuristic algorithm, in: Knowledge-Based Systems, 75, 1–18.

Schön, C. (2010): On the Optimal Product Selection Problem with Price Discrimination, in: Management Science, 56, 896–902.

Selka, S. (2013): Validität computergestützer Verfahren der Präferenzmessung, Dissertation.

Selka, S.; Baier, D. (2014): Kommerzielle Anwendung auswahlbasierter Verfahren der Conjointanalyse: Eine empirische Untersuchung zur Validitätsentwicklung, in: Marketing ZFP, 36 (1), 54–64.

Shah-Hosseini, H. (2011): Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation, in: International Journal of Computational Science and Engineering, 6 (1-2), 132–140.

Shapiro, J.; Prügel-Bennett, A.; Rattray, M. (1994): A statistical mechanics formulation of the dynamics of genetic algorithms, in: Lecture Notes in Computer Science, 17–27.

Shayanfar, H.; Gharehchopogh, F. S. (2018): Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems, in: Applied Soft Computing, 71, 728–746.

Shen, B.; Cao, Y.; Xu, X. (2020): Product line design and quality differentiation for green and non-green products in a supply chain, in: International Journal of Production Research, 58, 148–164.

Shi, L.; Ólafsson, S. (2000): Nested partitions method for global optimization, in: Operations Research, (48), 390–407.

Shi, L.; Ólafsson, S.; Chen, Q. (2001): An Optimization Framework for Product Design, in: MANAGEMENT SCIENCE, 47, 1681–1692.

Shi, Y.; Eberhart, R. (1998): A modified Particle Swarm Optimizer, in: IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence, 69–73.

Shocker, A. D.; Srinivasan, V. (1974): A Consumer-based Methodology for the Identification of New Product Ideas, in: Management Science, 20, 921–937.

Shocker, A. D.; Srinivasan, V. (1979): Multiattribute Approaches for Product Concept Evaluation and Generation: A Critical Review, in: Journal of Marketing Research, 16 (2), 159–180.

Shugan, S.; Balachandran, V. (1977): A Mathematical Programming Model for Optimal Product Line Structuring, in: Working Paper Series 7734 (October), University of Rochester Graduate School of Business, NY.

Simon, D. (2008): Biogeography-based optimization, in: IEEE transactions on evolutionary computation, 12 (6), 702–713.

Soerensen, K. (2013): Metaheuristics – the metaphor exposed, in: International Transactions in Operational Research, 22 (1), 3–18.

Srinivas, M.; Patnaik, L. (1994): Genetic algorithms: a survey, in: Computer, 27, 17–26.

Steiner, W.; Hruschka, H. (2002): Produktliniengestaltung mit genetischen Algorithmen, in: Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung, 54 (7), 575–601.

Steiner, W.; Hruschka, H. (2003): Genetic Algorithms for product design: how well do they really work?, in: International Journal of Market Research, 45 (2), 229–240.

Steiner, W. J.; Hruschka, H. (2000): Conjoint-basierte Produkt(linien)gestaltung unter Berücksichtigung von Konkurrenzreaktionen, in: OR Spektrum, 22, 71–95.

Storn, R.; Price, K. (1997): Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, in: Journal of global optimization, 11 (4), 341–359.

Stützle, T.; Hoos, H. (1997): MAX-MIN Ant System and Local Search for the Traveling Salesman Problem, in: Proceedings of 1997 IEEE International Conference on Evolutionary Computation, 309–314.

Stützle, T.; Hoos, H. (1998): Improvements on the Ant-System: Introducing the MAX-MIN Ant System, in: Artificial Neural Nets and Genetic Algorithms, 245–249.

Stützle, T.; Hoos, H. (2000): MIN-MAX Ant-System, in: Future Generation Computer Systems, 16, 889–914.

Su, S.; Wang, J.; Fan, W.; Yin, X. (2007): Good lattice swarm algorithm for constrained engineering design optimization, in: Wireless Communications, Networking and Mobile Computing, IEEE, 6421–6424.

Tacke, G.; Vidal, D.; Haemer, J. (2014): Profitable Innovation – Why it's urgent and how the best companies combine innovation and marketing to rebuild their pricing power and their profits, Simon-Kucher & Partners Strategy & Marketing Consultants GmbH.

Tang, R.; Fong, S.; Yang, X.-S.; Deb, S. (2012): Wolf search algorithm with ephemeral memory, in: Digital Information Management (ICDIM), IEEE, 165–172.

Tarasewich, P.; McMullen, P. R. (2001): A pruning heuristic for use with multisource product design, in: European Journal of Operational Research, 128, 58–73.

Teodorović, D.; Dell'Orco, M. (2005): Bee colony optimization a cooperative learning approach to complex transportation problems, in: Advanced OR and AI Methods in Transportation, 51–60.

Thakur, L.; Nair, S.; Wen, K.-W.; Tarasewich, P. (2000): A new model and solution method for product line design with pricing, in: Journal of the Operational Research Society, 51(1), 90–101.

Tilahun, S. L.; Ong, H. C. (2015): Prey-predator algorithm: a new metaheuristic algorithm for optimization problems, in: International Journal of Information Technology & Decision Making, 14 (6), 1331–1352.

Ting, T.; Man, K. L.; Guan, S.-U.; Nayel, M.; Wan, K. (2012): Weightless swarm algorithm (wsa) for dynamic optimization problems, in: IFIP International Conference on Network and Parallel Computing, Springer, 508–515.

Tsafarakis, S.; Marinakis, Y.; Matsatsinis, N. (2011): Particle swarm optimization for optimal product line design, in: International Journal of Research in Marketing, 28 (1), 13–22.

Tsafarakis, S.; Saridakis, C.; Baltas, G.; Matsatsinis, N. (2013): Hybrid particle swarm optimization with mutation for optimizing industrial product lines: An application to a mixed solution space considering both discrete and continuous design variables, in: Industrial Marketing Management, 42, 496–506.

Tsafarakis, S.; Zervoudakis, K.; Andronikidis, A.; Altsitsiadis, E. (2020): Fuzzy self-tuning

differential evolution for optimal product line design, in: European Journal of Operational Research, in press, Online: https://doi.org/10.1016/j.ejor.2020.05.018.

Tzanetos, A.; Dounias, G. (2017): A new metaheuristic method for optimization: sonar inspired optimization, in: International Conference on Engineering Applications of Neural Networks, Springer, 417–428.

Urban, G. L. (1969): A Mathematical Model Approach to Product Line Decisions, in: Journal of Marketing Research, 6, 40–47.

Voekler, S.; Baier, D. (2020): Investigating Machine Learning Techniques for Solving Product-line Optimization Problems, in: Archives of Data Science, Series A, 6 (1).

Voekler, S.; Krausche, D.; Baier, D. (2013): Product Design Optimization Using Ant Colony And Bee Algorithms: A Comparison, in: Studies in Classification, Data Analysis, and Knowledge Organization, 46, 491–498.

Vose, M. (1993): Modeling simple genetic algorithms, in: Whitley, L. (Hrsg.): Foundations of Genetic Algorithms 2, Morgan Kaufmann, San Mateo, CA, 63–73.

Vose, M. (1994): A closer look at mutation in genetic algorithms, in: Annals of Mathematics and Artificial Intelligence, 10, 423–434.

Vose, M.; Wright, A. (1995): Stability of vertex fixed points and applications, in: Whitley, D.; Vose, M. (Hrsg.): Foundations of Genetic Algorithms 3, Morgan Kaufmann, San Mateo, CA, 103–113.

Wang, G.-G.; Deb, S.; Gao, X.-Z.; Coelho, L. D. S. (2016): A new metaheuristic optimisation algorithm motivated by elephant herding behaviour, in: International Journal of Bio-Inspired Computation, 8 (6), 394–409.

Wang, H. F.; Hsu, H. W. (2010): A closed-loop logistic model with a spanning-tree based genetic algorithm, in: Computers & operations research, 37 (2), 376–389.

Wang, X.; Camm, J. D.; Curry, D. J. (2009): A Branch-and-Price Approach to the Share-of-Choice Product Line Design Problem, in: Management Science, 55, 1718–1728.

Wang, X. J.; Curry, D. J. (2012): A robust approach to the share-of-choice product design problem, in: Omega, 40, 818–826.

Wedde, H. F.; Farooq, M.; Zhang, Y. (2004): Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior, in: International Workshop on Ant Colony Optimization and Swarm Intelligence, Springer, 83–94.

Whitley, D. (1993): An executable model of a simple genetic algorithm, in: Whitley, L. (Hrsg.): Foundations of Genetic Algorithms 2, Morgan Kaufmann, San Mateo, CA, 45–62.

Wind, Y.; Claycamp, H. J. (1976): Planning Product Line Strategy: A Matrix Approach, in: Journal of Marketing, 40, 2–9.

Wittink, D.; Cattin, P. (1989): Commercial Use of Conjoint Analysis: An Update, in: Journal of Marketing, 53, 91–96.

Wittink, D.; Vriens, M.; Burhenne, W. (1994): Commercia Use of Conjoint Analysis in Europe: Results and Critical Reflections, in: International Journal of Research in Marketing, 11, 41–52.

Wolpert, H.; Macready, W. (1997): No free lunch theorems for optimization, in: IEEE Transactions on Evolutionary Computation, 1, 67–82.

Yang, X. (2012): Flower pollination algorithm for global optimization, in: International Conference on Unconventional Computing and Natural Computation, Springer, 240–249.

Yang, X.-S. (2005): Engineering optimizations via nature-inspired virtual bee algorithms, in: Artificial intelligence and knowledge engineering applications: a bioinspired approach, 317–323.

Yang, X.-S. (2010a**):** Firefly algorithm, stochastic test functions and design optimisation, in: International Journal of Bio-Inspired Computation, 2 (2), 78–84.

Yang, X.-S. (2010b**):** A new metaheuristic bat-inspired algorithm, in: Nature inspired cooperative strategies for optimization (NICSO), 65–74.

Yang, X. S.; Deb, S. (2009): Cuckoo search via lévy flights, in: Nature & Biologically Inspired Computing, IEEE, 210–214.

Zandi, Z.; Afjei, E.; Sedighizadeh, M. (2012): Reactive power dispatch using big bang-big crunch optimization algorithm for voltage stability enhancement, in: Power and Energy (PECon), IEEE, 239–244.

Zhang, L.; Dahlmann, C.; Zhang, Y. (2009): Human-inspired algorithms for continuous function optimization, in: Intelligent Computing and Intelligent Systems, IEEE, Band 1, 318–321.

Zhang, Z.; Dong, X.; Mantrala, M.; Zhang, Y. (2018): Optimal depth and timing of price promotions in a vertically differentiated product line, in: Journal of Business Research, 83, 215–228.

Zou, T.; Zhou, B.; Jiang, B. (2020): Product-Line Design in the Presence of Consumers' Anticipated Regret, in: Management Science, in press, Online: https://doi.org/10.1287/mnsc.2019.3506.

Zufryden, F. S. (1977): A Conjoint Measurement-Based Approach for Optimal New Product Design and Market Segmentation, in: A.D. Shocker (Ed.), Analytic Approaches to Product and Market Planning, Cambridge, MA: Marketing Science Institute, 100–114.