OPERATIONS RESEARCH
PROF. DR. BRIGITTE WERNERS

RUB

Niels Becker
Alexander Kornrumpf
Brigitte Werners

Hybrider evolutionärer Algorithmus zur optimalen Gestaltung von Produktprogrammen unter Berücksichtigung von Preisbündelung

Beitrag Nr. 2010/04 aus der Reihe
Beiträge zur Unternehmensforschung
Herausgegeben von Prof. Dr. Brigitte Werners
Fakultät für Wirtschaftswissenschaft
Ruhr-Universität Bochum

OPERATIONS RESEARCH
PROF. DR. BRIGITTE WERNERS

RUB

Niels Becker
Alexander Kornrumpf
Brigitte Werners

Hybrider evolutionärer Algorithmus zur optimalen Gestaltung von Produktprogrammen unter Berücksichtigung von Preisbündelung

Beitrag Nr. 2010/04 aus der Reihe
Beiträge zur Unternehmensforschung
Herausgegeben von Prof. Dr. Brigitte Werners

Universitätsstraße 150
44801 Bochum

Fon +49 (0)234 32-28311
Fax +49 (0)234 32-14267
Mail or@ruhr-uni-bochum.de

ISBN 978-3-939281-00-9 (PDF)

Inhaltsverzeichnis

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungs- und Symbolverzeichnis

Γ	Fitnessfunktion
$\Gamma(c)$	Menge der kundenindividuellen Bündel k, die in Restriktion c enthalten sind
δ	Design, Entscheidungsvariable
δ_{blvma}	Binärvariable, die angibt, ob Bündel b aus Linie l Variante v Merkmal m in Ausprägung a enthält
δ_{lvma}	Binärvariable, die angibt, ob aus Linie l Variante v Merkmal m in Ausprägung a enthält
δ_{vma}	Binärvariable, die angibt, ob Variante v Merkmal m in Ausprägung a enthält
$\Pi(\Psi)$	Preissystem zu Produktdesign Ψ
$\Pi^*(\Psi)$	Optimales Preissystem zu Produktdesign Ψ
$\Phi(q)$	Menge der kundenindividuellen Bündel k, die in Leistungsverbund q enthalten sind
θ	Allgemein Auswahlwahrscheinlichkeit, in deterministischen Modellen Binärvariable zur Erfassung der Kundenwahl
θ_{kb}	Binärvariable, die angibt, ob Kunde k Bündel b erwirbt
Ω	Synergieeffekte (=Kosteneinsparungen)
Ω_b	Synergieeffekte bei Bündel b
Ψ	Produktdesign
a	Ausprägung, a = 1, …, A_m
a_{bi}	Minimale Surplusdifferenz zwischen Bündel i und b
B´	Menge der gekauften Bündel. Es gilt: $B' = \{b : G_b \neq \varnothing\}$
b	Bündel, B = 0, …, B
b(k)	Bündel b, dem Kunde k zugeordnet ist
c	Restriktionen zur Einhaltung der Surplussuperadditivitätsbedingungen c = 1, …, C
DK	Dobson Kalish

DB	Deckungsbeitrag
E	Anzahl an Elitisten
EG	Erreichungsgrad
f_{bi}	Dualvariable
G_b	Menge aller Bündel b zugeordneten Kunden. Es gilt: $G_b = \{b : \theta_{kb} = 1\}$
GDB	Gesamtdeckungsbeitrag
GDB^{EA}	Gesamtdeckungsbeitrag der Lösung, die mit Hilfe des evolutionären Algorithmus ermittelt wurde
GDB^{opt}	Gesamtdeckungsbeitrag der optimalen Lösung
GDB_i	Gesamtdeckungsbeitrag von Instanz i
HR	Hit Rate
h_i	Variable, die angibt, ob der in einer Instanz i erzielte Erreichungsgrad ein vorgegebenes Anspruchsniveau übersteigt
i	Instanz, i = 1, …, I
k	Kunde, Kundensegment, k = 1, ..., K
K	Anzahl an Nachkommen bzw. Anzahl Kunden
k(c)	Kunde k für die Restriktion c gilt
l	Produktlinie, Produkttyp, l = 1, ..., L
m	Merkmal, m = 1, …, M_l
MaxGen	Maximale Anzahl an Generationen
MaxR	Maximaler Reservationspreis-Preisheuristik
MaxW	Maximale Wohlfahrt-Preisheuristik
MinGen	Minimale Anzahl an Generationen
MR	Mixing Rate, Wahrscheinlichkeit, dass an einem möglichen Crossover-Punkt die Ausgangsquelle wechselt
MR_{bun}	Mutationsrate Replace-Product-In-Bundle
MR_{line}	Mutationsrate Replace-Product-In-Line
MR_{spec}	Mutationsrate Change-Product-Specification
N	Anzahl gerechneter Instanzen

N_b	Anzahl der Kunden, die Bündel b erwerben. Es gilt: $N_b = \left\|G_b\right\|$
N_{ind}	Populationsgöße
n	Iteration
p	Preis, Entscheidungsvariable
p_k	Preis, den Kunde k für die von ihm gekauften Leistungen zahlt
p_b	Preis von Bündel b
PLD	Produktliniendesign
Pos	Position eines Individuums in einer nach aufsteigender Fitness sortierten Liste
q	Leistungsverbund, q = 1, …, Q
r_{cl}	Maximale Zahlungsbereitschaft von Kunde k(c) für eine Variante aus Linie l bei Kauf aller der in Restriktion c vorgesehenen Bündel, Variable
R	Reservationspreis (= Zahlungsbereitschaft)
R_{kb}	Reservationspreis von Kunde k für Bündel b
R_{klma}	Reservationspreis von Kunde k für Produkttyp l mit Merkmal m in Ausprägung a
$\text{runmean}_{Gen}^{best}$	laufender Mittelwert des besten Zielfunktionswertes der letzten Gen-Generationen
$\text{runmean}_{Gen}^{\varnothing}$	laufender Mittelwert des durchschnittlichen Zielfunktionswertes der letzten Gen-Generationen
$s_{kk'}$	Surplus von Kundensegment k für Leistungen, die Kunde k‘ erwirbt, Variable
SG_k	Segmentgröße, Anzahl von Kunden in Segment k
SimBun	Optimierungsmodell zur simultanen Bündelung
SP	Selection Pressure, Selektionsdruck
TH	Abbruchschwelle
v	Variante, v = 1 … V_l
VK	Variable Herstellkosten
VK_b	Variable Kosten von Bündel b

VK_{lma}	Variable Herstellkosten von Produkttyp l mit Merkmal m in Ausprägung a
x	Entscheidungsvariable
x_{klvma}	Binärvariable, die angibt, ob Kunde k aus Linie l Variante v mit Merkmal m in Ausprägung a erwirbt
ZB	Zahlungsbereitschaft

1 Einleitung

Die Erstellung und Verwertung von Leistungen steht im Mittelpunkt der unternehmerischen Tätigkeiten in Industrieunternehmen. Vor der Leistungserstellung ist jedoch von jedem Unternehmen festzulegen, welche Leistungen überhaupt angeboten werden sollen. Die Gesamtheit der angebotenen Leistungen, das Produktprogramm, besteht häufig aus mehreren Produktlinien, welche die Varianten eines Produkttyps umfassen. Nur Unternehmen, die in der Lage sind, ihr Produktprogramm immer wieder an sich verändernde Kundenbedürfnisse und Marktbedingungen anzupassen, können langfristig erfolgreich sein.

Für den ökonomischen Erfolg eines Unternehmens hingegen ist die Fähigkeit, Produkte entsprechend der Kundenbedürfnisse zu gestalten, nicht ausreichend. Es gilt, die angebotenen Leistungen auch bestmöglich zu verwerten. Dazu sind die Zahlungsbereitschaften der Kunden für die angebotenen Produkte zu nutzen und durch eine geeignete Preissetzung in Deckungsbeiträge für das Unternehmen umzuwandeln. Immer häufiger wird in der Praxis hierzu das Instrument der Preisbündelung eingesetzt. Eigenständige Produkte werden zu Bündel kombiniert und zu einem einheitlichen Paketpreis angeboten. Dadurch können von unterschiedlichen Kunden unterschiedliche Preise verlangt und so der Gewinn deutlich gegenüber einem ungebündelten Angebot gesteigert werden.

Preisbündelung sollte bereits bei der Gestaltung von Produkten und Produktprogrammen berücksichtigt werden, um bestmögliche Ergebnisse zu erzielen, wozu leistungsfähige Algorithmen bisher jedoch fehlen. Vor diesem Hintergrund liegt der Schwerpunkt dieses Arbeitsberichts in der Entwicklung eines leistungsfähigen Algorithmus zur Unterstützung von Entscheidungsträgern im Produkt- und Preismanagement bei der gemeinsamen Gestaltung von Produktdesign, Bündelkonfiguration und Preisen. Hierzu wird im Weiteren ein hybrider evolutionärer Algorithmus entwickelt und umfassend evaluiert.

Der Arbeitsbericht gliedert sich entsprechend Abbildung 1.1 in fünf Kapitel, die im Folgenden erläutert werden. Zunächst werden in Kapitel 2 kurz die Grundlagen einer quantitativen Produktprogrammgestaltung mit Berücksichtigung von Preisbündelung thematisiert. Zunächst wird die Problemstellung dargelegt, bevor verschiedene Formen der Integration der Preisbündelung diskutiert werden. Anschließend werden benötigte Modellannahmen erläutert, bevor ein quantitatives Grundmodell zur Produktprogrammoptimierung mit simultaner Berücksichtigung der Preisbündelung vorgestellt wird.

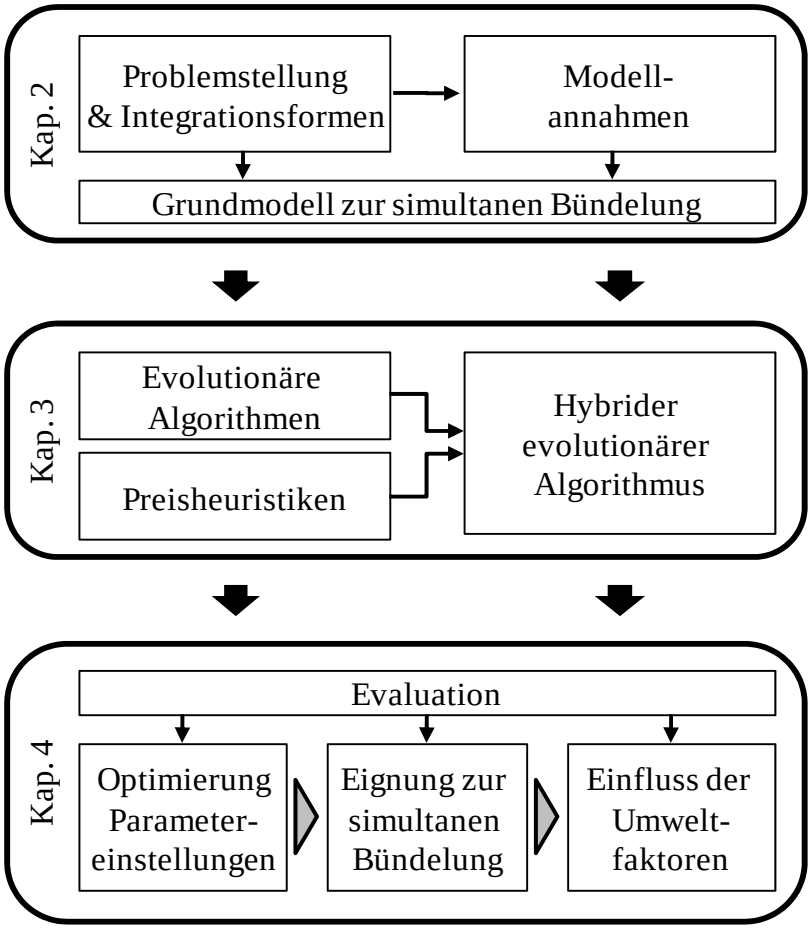

Abbildung 1.1: Aufbau des Arbeitsberichts

Kapitel 3 beschäftigt sich mit der Entwicklung eines hybriden evolutionären Algorithmus zur Lösung der Produktprogrammoptimierung. Nach einer kurzen Vorstellung evolutionärer Algorithmen, ihrer Einordnung in die Meta-Heuristiken und die Vorstellung der Hybridisierung von Heuristiken wird der hybride evolutionäre Algorithmus im Detail erläutert. Dazu wird zuerst die Dekomposition des Gesamtproblems in ein Design-Master- und ein Pricing-Subproblem dargelegt, bevor sowohl der evolutionäre Teil als auch die Preisheuristiken und die Integration beider Komponenten ausführlich vorgestellt werden.

Die Evaluation des entwickelten, hybriden Algorithmus ist Gegenstand von Kapitel 4. Vor der eigentlichen Evaluation werden mit Hilfe von Berechnungsexperimenten gute Einstellungen für die verschiedenen Parameter bestimmt. In einer umfangreichen experimentellen Studie werden anschließend die mit Hilfe des Algorithmus erzielten Lösungen der optimalen Lösung gegenübergestellt. Im dritten Teil des Kapitels wird zusätzlich der Einfluss der Umweltfaktoren auf die Performance des Algorithmus näher untersucht. Die Arbeit schließt im fünften Kapitel mit einer Zusammenfassung der Ergebnisse und einem Ausblick.

2 Produktprogrammoptimierung mit Berücksichtigung von Preisbündelung

2.1 Problemstellung und Integrationsformen

Die optimale Gestaltung von Produktprogrammen unter Berücksichtigung von Preisbündelung vereint die beiden Teilprobleme der Produktliniengestaltung und der optimalen Preisbündelung. Der Betrachtungsgegenstand der Produktliniengestaltung ist in Abbildung 2.1 dargestellt.

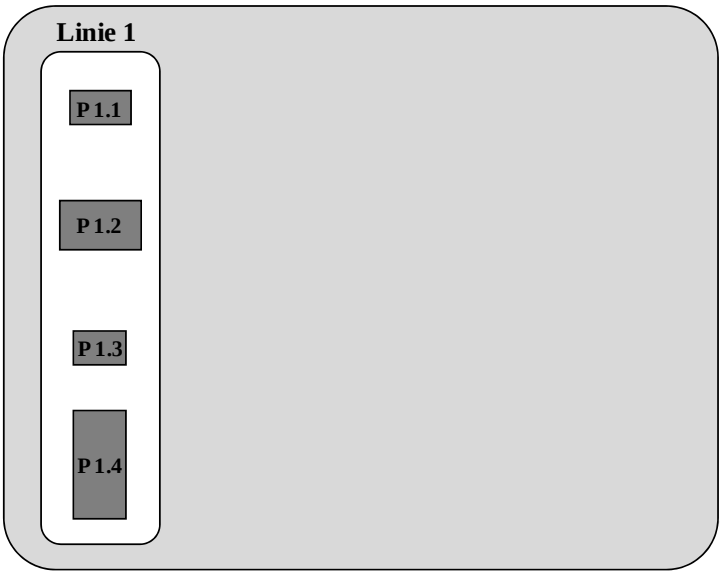

Abbildung 2.1: Betrachtungsgegenstand Produktliniengestaltung

Eine Produktlinie besteht aus mehreren Varianten eines Produkttyps. Aufgrund dessen werden sämtliche Varianten einer Linie durch die gleichen Merkmale beschrieben. Sie können jedoch unterschiedliche Ausprägungen aufweisen und unterscheiden sich paarweise in mindestens einer Merkmalsausprägung. Aufgrund ihrer Ähnlichkeit stellen sie aus Sicht der Kunden Substitute dar, sodass Kunden zur Bedürfnisbefriedigung maximal eine Variante erwerben. Zur optimalen Gestaltung einer Produktlinie ist die Anzahl der in ihr enthaltenen Varianten zu bestimmen und das Produktdesign sowie der Preis jeder Variante festzulegen. Das Design einer Variante ist vollständig spezifiziert, wenn in jedem beschreibenden Merkmal genau eine Ausprägung festgelegt ist.

Im Gegensatz zur Produktliniengestaltung wird im Rahmen der Preisbündelung üblicherweise davon ausgegangen, dass das Design der Produkte gegeben ist. Der Betrachtungsgegenstand der Preisbündelung ist in Abbildung 2.2 dargestellt.

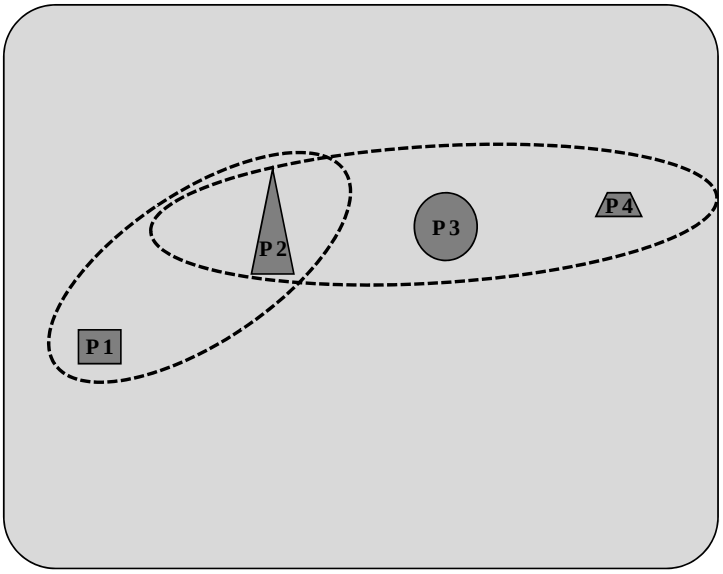

Abbildung 2.2: Betrachtungsgegenstand Preisbündelung

Im Rahmen der Preisbündelung wird angenommen, dass mehrere Einzelprodukte existieren, die beliebig zu Bündeln kombiniert werden können. Dies impliziert, dass die vorhandenen Einzelprodukte verschiedene Produkttypen darstellen und somit aus verschiedenen Produktlinien entstammen. Die optimale Preisbündelung besteht aus der Bestimmung der optimalen Anzahl an Bündel, deren Zusammensetzung und die Festlegung der Bündelpreise.

Die optimale Produktprogrammgestaltung unter Berücksichtigung von Preisbündelung vereint die dargestellten Problemstellungen (vgl. Abbildung 2.3). So ist zum einen für jede betrachtete Linie das Design sämtlicher Varianten sowie alle Preise festzulegen, zum anderen sind zusätzlich Varianten zu Bündel zu kombinieren und zu bepreisen.

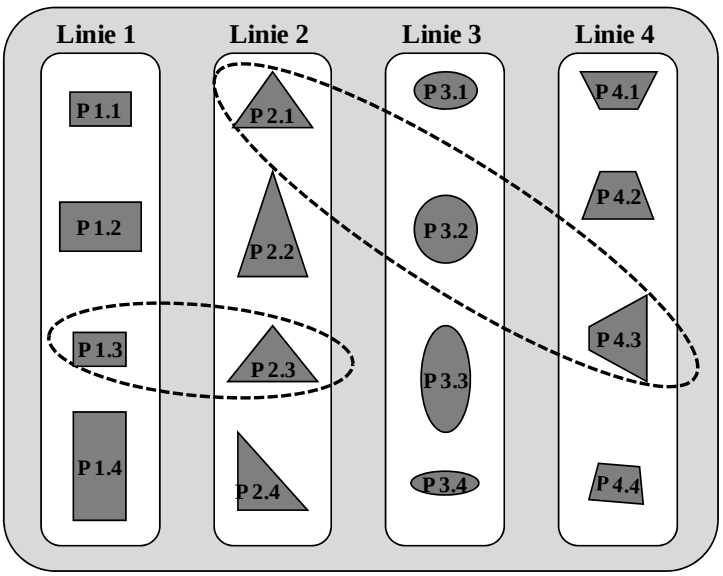

Abbildung 2.3: Betrachtungsgegenstand Produktprogrammgestaltung mit Preisbündelung

Grundsätzlich existieren zwei in Abbildung 2.4 dargestellte Möglichkeiten, Preisbündelung in das Produktliniendesign zu integrieren.

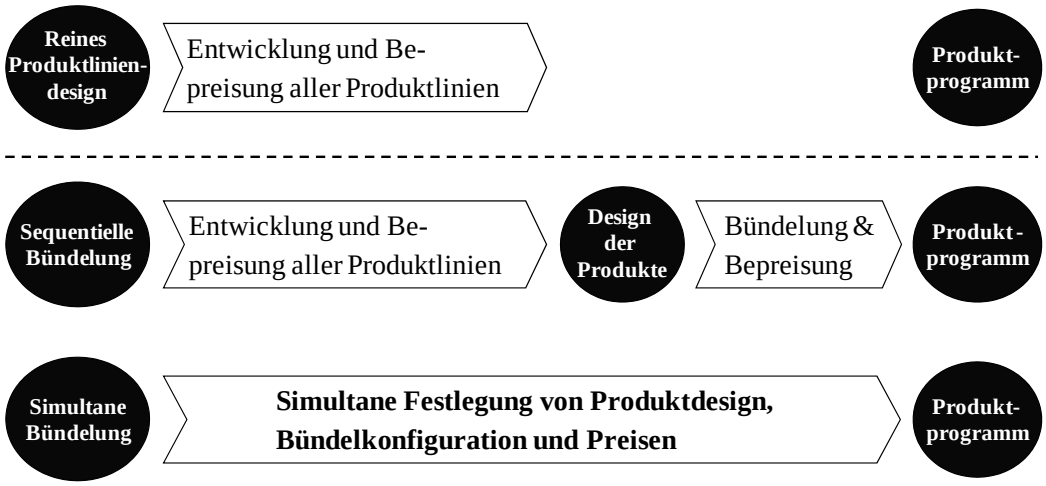

Abbildung 2.4: Integrationsformen der Preisbündelung

Ohne Berücksichtigung von Preisbündelung wird jede Produktlinie separat betrachtet und optimiert. Diese Optimierung wird im Weiteren als *reines Produktliniendesign* bezeichnet.[1] Der Kunde wird aus jeder Linie die Variante erwerben, welche ihm den größten Surplus bietet. Existiert in einer Linie keine Variante mit positivem Surplus, wird der Kunde hingegen diesen Produkttyp nicht kaufen. Durch Preisbündelung steigt der Gesamtdeckungsbeitrag, denn nur dann sollten sinnvollerweise Bündel gebildet und angeboten werden. Ist dies hingegen nicht der Fall, so ist auf das Angebot von Bündeln zu verzichten. Der Gesamtdeckungsbeitrag des reinen Produktliniendesigns stellt damit einerseits die Untergrenze, andererseits den Vergleichsmaßstab für die beiden Integrationsformen dar.

Als erste Möglichkeit zur Integration bietet sich ein sequentielles Vorgehen an, was daher im Weiteren als s*equentielle Bündelung* bezeichnet wird. Hierbei wird zunächst jede Produktlinie für sich betrachtet und das Produktdesign für jede Variante ermittelt. Auf Basis der in diesem ersten Schritt festgelegten Produktvarianten wird die Preisbündelung durchgeführt. Es werden optimale Bündel aus den bereits spezifizierten Varianten gebildet und optimale Preise bestimmt. Falls notwendig, werden in diesem Schritt auch die Preise der Einzelprodukte angepasst. Das Produktdesign kann hingegen nicht modifiziert werden.

Anstatt Bündel im Anschluss an eine Produktoptimierung zu konfigurieren, besteht zweitens die Möglichkeit, Produktliniendesign und Preisbündelung simultan durchzuführen. Dies wird im Folgenden daher als *simultane Bündelung* bezeichnet. Hierbei

[1] Streng genommen handelt es sich um ein Multi-Produktliniendesign, da mehrere Linien betrachtet werden.

wird das optimale Produktdesign jeder Variante in jeder Linie zusammen mit der optimalen Bündelkonfiguration und den Preisen bestimmt. Analog zur sequentiellen Bündelung besitzen Kunden die Möglichkeit, eine beliebige Kombination aus Einzelprodukten und Bündeln zu erwerben.

2.2 Modellannahmen

Die notwendigen Annahmen lassen sich in vier Kategorien einteilen: *Annahmen über die Kunden* und ihr Verhalten, *Annahmen über den Markt*, in dem das Unternehmen agiert, *Annahmen über die Produktion*, insbesondere die Kosten der Leistungserstellung, und *Annahmen über die zur Verfügung stehenden Informationen*. Sie lauten:

1. Rationale, nutzenmaximierende Kunden
2. Kein Nutzen aus zusätzlichen Varianten von erworbenen Produkttypen
3. Additive Zahlungsbereitschaft für Bündel
4. Kundensegmente mit homogener Zahlungsbereitschaft
5. Kostenlose Entsorgung und kostenlose Bündelung
6. Gültigkeit der Preise für alle Kunden
7. Kein Weiterverkauf von erworbenen Leistungen
8. Gesamtdeckungsbeitragsmaximierung als Unternehmensziel
9. Keine Reaktion von Wettbewerbern
10. Keine technischen Restriktionen der Produktgestaltung
11. Fixkosten sind unabhängig von Produktprogramm und Produktdesign
12. Additive Kosten der Bündel
13. Im Umfang streng monoton steigende Bündelkosten
14. Vollständige Kenntnis aller benötigten Informationen

Annahmen über die Kunden

Die Kunden verhalten sich rational und versuchen, ihren Nettonutzen zu maximieren. Dies erreichen sie, indem sie Leistungen derart kombinieren, dass ihr Surplus maximal ist.

Kunden können pro Produktlinie nur aus einer Variante einen Nutzen ziehen und besitzen dementsprechend für weitere Varianten eines bereits erworbenen Produkttyps keine Zahlungsbereitschaft. Es können jedoch Situationen auftreten, in denen Kunden auch für weitere Varianten einen Nutzen haben. Zur Erfassung solcher Situationen ist der Kauf in zwei unabhängige Kaufprozesse zu separieren und der Kunde als zwei ge-

trennte Kunden mit identischer Zahlungsbereitschaft zu behandeln, die jeweils eine Leistung erwerben. Sollte aus irgendwelchen Gründen ein Kunde zwei Varianten eines Produkttyps erwerben, so wird er die Variante nutzen, die ihm den höheren Bruttonutzen bietet. Die Zahlungsbereitschaft R_{kl} eines Kunden k für einen Produkttyp l ist daher das Maximum der Zahlungsbereitschaften für alle Varianten, bzw. formal:

$$R_{kl} := \max_{v} \left\{ \sum_{m=1}^{M_l} \sum_{a=1}^{A_m} R_{klma} \cdot \delta_{lvma} \right\} \qquad (2.1)$$

Es wird angenommen, dass sich die Zahlungsbereitschaft für Bündel additiv aus den Zahlungsbereitschaften der enthaltenen Varianten zusammensetzen. Formal ergibt sich die Zahlungsbereitschaft von Kunde k für Bündel b gemäß:

$$R_{kb} := \sum_{l=1}^{L} \max_{v} \left\{ \sum_{m=1}^{M_l} \sum_{a=1}^{A_m} R_{klma} \cdot \delta_{blvma} \right\} \qquad (2.2)$$

Zahlungsbereitschaften innerhalb eines Kundensegments sind vollständig homogen, d.h., Kunden desselben Segments besitzen identische Zahlungsbereitschaften. Kunden können nicht gewünschte Produkte kostenlos entsorgen und kostenlos Einzelprodukte oder Bündel zu komplexeren Bündeln zusammensetzen.

Annahmen über den Markt

Ermittelte Preise gelten für alle Kunden, sodass keine kundenspezifische Preisdifferenzierung möglich ist. Ein direkter Weiterverkauf von erworbenen Produkten ist nicht möglich. Dies verhindert, dass angebotene Produkte zeitgleich zu günstigeren Preisen verkauft werden. Der Anbieter verfolgt eine Abschöpfungsstrategie und möchte seinen Gesamtdeckungsbeitrag maximieren. Alternativ sind Umsatz- oder Marktanteilsmaximierung möglich.

Vorhandene Wettbewerber reagieren nicht auf Veränderungen des Produktprogramms durch den Anbieter. Durch reagierende Wettbewerber würden mit großer Wahrscheinlichkeit Programmanpassungen des Anbieters notwendig werden, wonach anschließend die Wettbewerber ihrerseits auf die neue Situation reagieren usw. In solchen Situationen können mit Hilfe spieltheoretischer Modelle Gleichgewichtspunkte bestimmt werden.[2] Spieltheoretische Modelle besitzen zum einen den Nachteil, dass sie meist nur für kleine Produktprogramme und sehr wenige Wettbewerber eine Lösung ermitteln können, zum anderen setzen sie die Kenntnis der Kostenstruktur der Wettbewerber voraus, was in der Praxis nur in den wenigsten Fällen gegeben sein dürfte.

[2] Vgl. u.a. Steiner (1999), S. 173-192.

Annahmen über Produktion und Herstellkosten

Es existieren keine technischen Restriktionen hinsichtlich der Gestaltung von Produktvarianten. Eine Variante ist vollständig beschrieben, wenn in jedem Merkmal genau eine Ausprägung ausgewählt wird, wobei Merkmalsausprägungen beliebig miteinander kombiniert werden dürfen.

Fixkosten sind unabhängig vom angebotenen Produktprogramm und vom Produktdesign der Varianten. Sie sind daher im Kontext der Produktprogrammoptimierung nicht entscheidungsrelevant und können vernachlässigt werden.

Die Kosten eines Bündels setzen sich analog zu den Zahlungsbereitschaften additiv aus den Kosten der enthaltenen Produkte zusammen. Alternativ sind durch Bündelung Synergieeffekte möglich. Dadurch sind die Kosten eines Bündels subadditiv. Superadditive Kosten können hingegen in additive oder subadditive Kosten überführt werden, indem die zusätzlichen Kosten in Form eines weiteren Einzelprodukts, z.B. Montage der übrigen Produkte, erfasst werden. Formal ergeben sich die variablen Kosten VK_b eines Bündels b aus den variablen Kosten der enthaltenden Produkte abzüglich der im Bündel b realisierbaren Synergieeffekte Ω_b:

$$VK_b := \sum_{l=1}^{L} \sum_{v=1}^{V_l} \sum_{m=1}^{M_l} \sum_{a=1}^{A_m} VK_{lma} \cdot \delta_{blvma} - \Omega_b \qquad (2.3)$$

Die Kosten der Bündel sind im Bündelumfang streng monoton steigend, was bedeutet, dass umfangreichere Bündel höhere Kosten aufweisen. Enthält ein Bündel B_2 alle Bestandteile eines Bündels B_1 und darüber hinaus weitere Bestandteile, so wird Bündel B_2 als umfangreicher gegenüber B_1 bezeichnet, bzw. formal $B_2 \supset B_1$. Dann gilt für die variablen Herstellkosten beider Bündel:

$$VK_2 > VK_1 \qquad \text{, falls } B_1 \subset B_2 \qquad (2.4)$$

Dies ist für additive und superadditive Bündelkosten stets erfüllt, bei Subadditivität genau dann, wenn die Synergieeffekte geringer als die Kosten des zusätzlich enthaltenen Produktes sind.

Annahmen über verfügbare Informationen

Der Anbieter besitzt vollständige Kenntnis aller benötigten Informationen. Aufgrund dessen handelt es sich um eine Entscheidungssituation unter Sicherheit.

2.3 Grundmodell zur Produktprogrammoptimierung mit simultaner Preisbündelung

Im Rahmen der Produktprogrammoptimierung mit simultaner Preisbündelung sind Produktdesign, Bündelkonfiguration und Preise simultan zu bestimmen. Zur Angabe des Modells werden folgende Indizes, Parameter und Entscheidungsvariablen benötigt.

Indizes und Indexmengen

$k = 1, \ldots, K$	Kundensegmente
$l = 1, \ldots, L$	Produktlinien
$v = 1, \ldots, V_l$	Varianten der Produktlinie l
$m = 1, \ldots, M_l$	Merkmale, die Produktlinie l beschreiben
$a = 1, \ldots, A_m$	Ausprägungen von Merkmal m
$q = 1, \ldots, Q$	Kaufbare Leistungsverbunde, $Q = 2^k - 1$
$\Phi(q)$	Menge der kundenindividuellen Bündel k, die in Leistungsverbund q enthalten sind

Parameter

R_{klma}	Reservationspreis / Zahlungsbereitschaft von Kunde k für Produkttyp l mit Merkmal m in Ausprägung a
SG_k	Segmentgröße von Kundensegment k
VK_{lma}	Variable Kosten zur Realisierung in Produkttyp l das Merkmal m in Ausprägung a

Variablen

p_k Preis, den Kunde k für das von ihm erworbene Bündel bezahlt

$$x_{klvma} = \begin{cases} 1, & \text{falls Kunde k aus Linie l Variante v} \\ & \text{mit Merkmal m in Ausprägung a kauft} \\ 0, & \text{sonst} \end{cases}$$

$$\delta_{lvma} = \begin{cases} 1, & \text{falls in Linie l Variante v Merkmal m in Ausprägung a enthält} \\ 0, & \text{sonst} \end{cases}$$

Um die Übersichtlichkeit des Modells zu wahren, wird der Surplus eines Kunden k bei Wahl des für Kunden k´ vorgesehenen Bündel wie folgt bezeichnet:

$$s_{kk'} := \sum_{l=1}^{L} \sum_{v=1}^{V_l} \sum_{m=1}^{M_l} \sum_{a=1}^{A_m} R_{klma} \cdot x_{k'lvma} - p_{k'}$$

Mit den eingeführten Indizes, Parametern und Variablen kann das folgende nichtlineare gemischt-ganzzahlige Modell *SimBun* zur Produktprogrammoptimierung mit simultaner Preisbündelung aufgestellt werden:

$$\max \quad \sum_{k=1}^{K} SG_k \cdot \left(p_k - \sum_{l=1}^{L} \sum_{v=1}^{V_l} \sum_{m=1}^{M_l} \sum_{a=1}^{A_m} VK_{lma} \cdot x_{klvma} \right) \tag{2.5}$$

$$\text{s.d.} \quad s_{kk} \geq s_{kk'} \qquad \forall k, k' \tag{2.6}$$

$$s_{kk} \geq \sum_{l=1}^{L} \left(\max_{\substack{v \\ k' \in \Phi(q)}} \sum_{m=1}^{M_l} \sum_{a=1}^{A_m} R_{klma} \cdot x_{k'lvma} \right) - \sum_{k' \in \Phi(q)} p_{k'} \qquad \forall k, q \tag{2.7}$$

$$s_{kk} \geq 0 \qquad \forall k \tag{2.8}$$

$$\sum_{v=1}^{V_l} \sum_{a=1}^{A_m} x_{klvma} \leq 1 \qquad \forall k, l, m \tag{2.9}$$

$$\sum_{a=1}^{A_m} x_{klvma} = \sum_{a=1}^{A_{m'}} x_{klvm'a} \qquad \forall k, l, v, m, m' \tag{2.10}$$

$$\delta_{lvma} \geq x_{klvma} \qquad \forall k, l, v, m, a \tag{2.11}$$

$$\sum_{a=1}^{A_m} \delta_{lvma} = 1 \qquad \forall l, v, m \tag{2.12}$$

$$x_{klvma} \in \{0,1\} \qquad \forall k, l, v, m, a \tag{2.13}$$

$$p_k, \delta_{lvma} \geq 0 \qquad \forall k, l, v, m, a \tag{2.14}$$

Das Modell ist wie folgt aufgebaut: Die Zielfunktion (2.5) maximiert den Gesamtdeckungsbeitrag. Die Restriktionen (2.6) und (2.8) modellieren die First-Choice Regel, Restriktion (2.7) sichert die Surplussuperadditivität. Restriktion (2.9) verbietet zeitgleich den Kauf von mehr als einer Variante pro Linie und die Wahl von mehr als einer Ausprägung pro Merkmal. Durch Restriktion (2.10) können nur vollständig definierte Varianten in den Bündeln enthalten sein, während (2.11) und (2.12) nur kundeneinheitlich definierte Varianten ermöglichen. Die Variablendefinition ist mit (2.13) und (2.14) gegeben.

Die Lösung des Modells erfolgt, indem zunächst Restriktion (2.7) fallen gelassen wird, wodurch sich das Modell *SimBun Relax* ergibt. Nach erfolgreicher Lösung des Modells wird überprüft, ob alle angebotenen Bündel surplussuperadditiv sind. Ein für einen Kunden vorgesehenes Bündel ist surplussuperadditiv, wenn durch dessen Erwerb der Surplus des Kunden größer oder gleich dem Surplus aus jeder anderen durch die angebotenen Bündel möglichen Kombination ist. Nur in diesem Fall wird ein Kunde das für ihn bestimmte Bündel erwerben. Bei gegebener Surplussuperadditivität ist die Lösung des relaxierten Modells gleichzeitig optimale Lösung des Ausgangsmodells *SimBun*. Ist die Surplussuperadditivität hingegen verletzt, wird das entsprechende

Bündel und die anstelle dessen gekauften Bündel identifiziert und als zusätzliche Restriktion dem Modell hinzugefügt. Dadurch ergibt sich ein erweitertes Modell *SimBun Relax* wie folgt:

Zusätzliche Indizes und Indexmengen

c = 1, …, C	Restriktionen zur Beachtung der Surplussuperadditivität
k(c)	Kunde k, für den Restriktion c gilt
Γ(c)	Menge der kundenindividuellen Bündel k, die in Restriktion c enthalten sind

Zusätzliche Variable

r_{cl}	Maximale Zahlungsbereitschaft von Kunde k(c) für eine Variante aus Linie l bei Kauf aller der in Restriktion c vorgesehenen Bündel

Zusätzliche Restriktionen

$$r_{cl} \geq \sum_{v=1}^{V_l} \sum_{m=1}^{M_l} \sum_{a=1}^{A_l} R_{k(c)lma} \cdot x_{k'lvma} \qquad \forall c, l, k' \in \Gamma(c) \qquad (2.15)$$

$$s_{k(c)} \geq \sum_{l=1}^{L} r_{cl} - \sum_{k' \in \Gamma(c)} p_{k'} \qquad \forall c \qquad (2.16)$$

$$r_{cl} \geq 0 \qquad \forall c, l \qquad (2.17)$$

Restriktion (2.15) bestimmt pro Produktlinie aus allen verletzenden Bündeln pro Linie die Variante, für die Kunde k(c) die größte Zahlungsbereitschaft aufweist. Durch Restriktion (2.16) wird in der nächsten Iteration sichergestellt, dass der Surplus aus dem für Kunde k(c) bestimmten Bündel größer ist als der Gesamtsurplus der den Konflikt verursachenden Bündel. Abschließend erweitert Restriktion (2.17) die Variablendeklaration.

Dieses Verfahren führt zum Optimum, da in jedem Schritt eine zusätzliche Restriktion hinzugefügt wird, bis schließlich alle durch (2.7) gegebenen Restriktionen eingehalten sind. In realistischen Problemstellungen werden nur wenige Iterationen notwendig sein. Trotzdem kann die Optimierung vorzeitig terminiert werden, sollte für den Anwender die mit einer Fortführung der Optimierung maximal mögliche Verbesserung, zu gering sein. Die maximal mögliche Verbesserung wird üblicherweise als GAP bezeichnet, der angibt, um wie viel Prozent die optimale Lösung maximal von der besten bisher erreichten zulässigen Lösung entfernt ist. Einen abschließenden Überblick über den Ablauf gibt Abbildung 2.5.

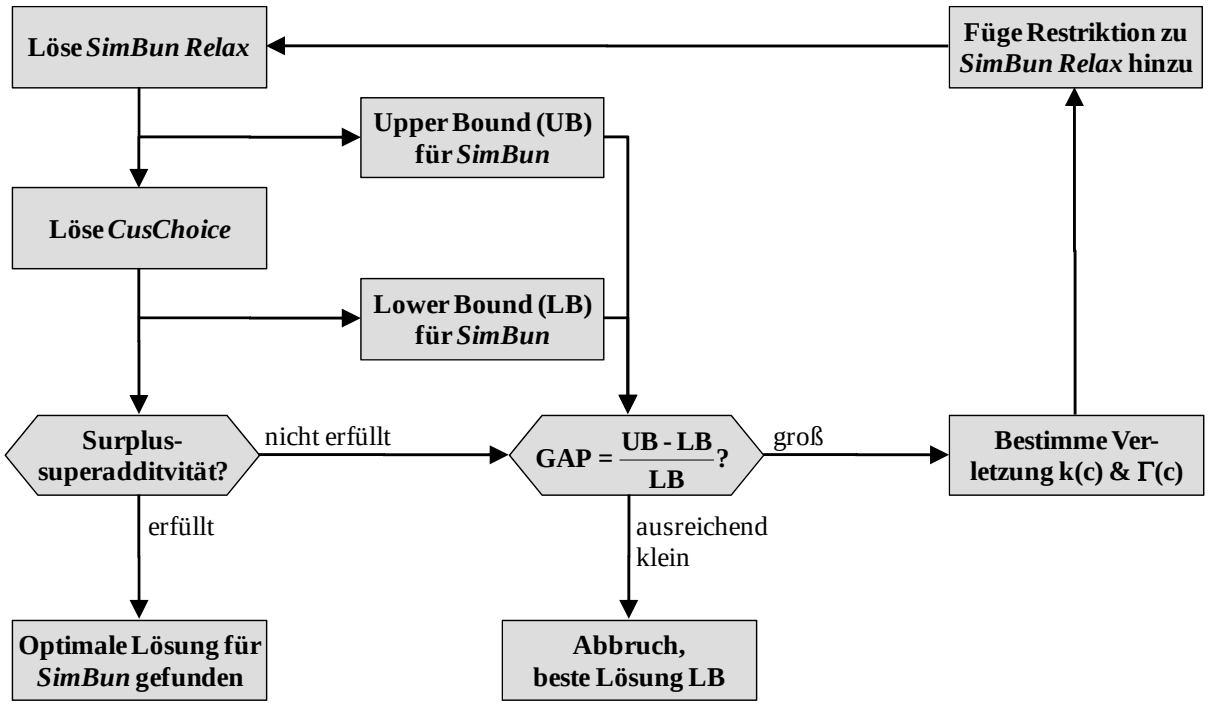

Abbildung 2.5: Ablauf der Produktprogrammoptimierung mit simultaner Preisbündelung

3 Entwicklung eines hybriden evolutionären Algorithmus

3.1 Grundlegendes

Zur Entwicklung eines hybriden evolutionären Algorithmus als heuristisches Lösungsverfahren für schwierige Probleminstanzen wird zunächst knapp geklärt, was unter Heuristik, evolutionärem Algorithmus und Hybridisierung zu verstehen ist. Im Folgenden werden die für das Verständnis der weiteren Arbeit notwendigen Ideen und Konzepte erläutert, für detailliertere Erläuterungen wird auf die angegebene Literatur verwiesen.

3.1.1 Grundideen heuristischer Lösungsverfahren und ihrer Bewertung

Im Bereich des Operations Research werden Heuristiken seit Jahren mit großem Erfolg zur Lösung mathematischer Modelle eingesetzt.[102] Sehr konkret und umfassend können Heuristiken im Kontext mathematischer Optimierung wie folgt definiert werden:

Heuristiken sind (meist) iterative Algorithmen, die aufgrund von Erfahrung oder vernünftiger Überlegung in der Lage erscheinen, gute, d.h. nahoptimale Lösungen eines mathematischen Optimierungsmodells in vertretbarer Zeit und akzeptablem Rechenaufwand zu finden, ohne die Optimalität der gefunden Lösung garantieren zu können.[103]

Heuristiken basieren meist auf einem einfachen, nachvollziehbaren Vorgehen zur Lösung eines spezifischen Problems, wodurch ein Einsatz für andere Probleme nahezu ausgeschlossen ist. Heuristiken werden immer dann eingesetzt, wenn Modelle so komplex[104] sind, dass sie mit exakten bzw. konvergierenden Algorithmen entweder nicht oder nicht in der zur Verfügung stehenden Zeit gelöst werden können.[105] Dazu zählen insbesondere kombinatorische Probleme.[106]

Modelle sind im Rahmen des modellorientierten Problemlösungsprozesses[107] so einfach wie möglich zu konstruieren.[108] Diesem Prinzip folgend sollten Modelle einfach

---

[102] Vgl. Müller-Merbach (1981), S. 1.

[103] Vgl. Hillier/Lieberman (2005), S. 617; Silver (2004), S. 936; Barr et al. (1995), S. 10; Reeves/Beasley (1993), S. 6; Müller-Merbach (1981), S. 1.

[104] Für eine Einführung in die Komplexitätstheorie etwa Wegener (2003). Für einen Überblick über komplexe Probleme der Klasse NP vgl. Garey/Johnson (1979).

[105] Vgl. Domschke/Drexl (2005), S. 127; Neumann/Morlock (2002), S. 402; Rardin/Uzsoy (2001), S. 261.

[106] Vgl. Zimmermann (2008), S. 278; Müller-Merbach (1981), S. 1.

[107] Vgl. u.a. Werners (2008), S. 2f.; Taha (2006), S. 8–10; Hillier/Lieberman (2005), S. 8–23; Winston (2004), S. 5f.; Jensen/Bard (2003), S. 1–9.

genug gehalten sein, um mit exakten Algorithmen gelöst werden zu können. Demgegenüber steht das Prinzip der Realitätsnähe, durch welches alle in der Realität relevanten Zusammenhänge zu erfassen und im Modell zu berücksichtigen sind.[109] Durch weitere evtl. realitätsferne Annahmen kann eine größere Einfachheit der Modelle zu Lasten der Realitätsnähe erreicht werden. In vielen Fällen wird jedoch eine gute, evtl. nicht optimale Lösung für ein komplexeres aber realitätsnahes Modell eine bessere Entscheidungsunterstützung bieten als eine optimale Lösung für ein einfacheres, aber realitätsfernes Modell.[110]

Als weitere Vorteile von Heuristiken werden u.a. die einfache Nachvollziehbarkeit der Lösungsermittlung und damit größere Akzeptanz bei Entscheidungsträgern, eine geringere Sensitivität gegenüber Datenqualität und Variationen von Problemparametern sowie deren ergänzender Einsatz in exakten Verfahren genannt.[111]

Problemspezifische Heuristiken besitzen jedoch auch einige Nachteile. Zum einen können sie nur schwierig auf neue Probleme übertragen werden und sind daher jeweils neu zu entwickeln, zum anderen tendieren viele einfache Heuristiken dazu, lokale statt globale Optima zu identifizieren.[112] Zur Überwindung dieser Probleme wurden allgemeine Lösungsstrategien, sogenannte Meta-Heuristiken, entwickelt:

Meta-Heuristiken sind generische, im Wesentlichen nicht problemspezifische Lösungsstrategien, die vorgeben, wie untergeordnete lokale Verbesserungsheuristiken und übergeordnete Strategien zu kombinieren sind, um lokale Optima zu verlassen und bessere Lösungen zu erzielen.[113]

Die bekanntesten Meta-Heuristiken sind von der Natur inspiriert und werden daher als naturanaloge Verfahren bezeichnet.[114] Dazu zählen Simulated Annealing[115] (in Anlehnung an physikalische Abkühlungsprozesse), evolutionäre Algorithmen[116] (basierend auf dem Prinzip der Evolution), Tabu Search[117] (unter Einbeziehung eines Gedächtnisses) sowie Ameisen-[118] und Schwarm-[119] Algorithmen (aus dem Bereich der Ethnolo-

---

[108] Vgl. Pidd (1999), S. 120–123; Ravindran et al. (1987), S. 7.
[109] Vgl. Hillier/Lieberman (2005), S. 13, dort mit „Tractability“ und „Precision“ bezeichnet sowie Jensen/ Bard (2003), S. 5.
[110] Vgl. u.a. Michalewicz/Fogel (2004), S. 18f.; Silver (2004), S. 937. Ähnlich Fink/Voß (2003), S. 395.
[111] Vgl. Silver (2004), S. 937.
[112] Vgl. Dréo et al. (2006), S. 4f.; Hillier/Lieberman (2005), S. 620.
[113] Vgl. Hillier/Lieberman (2005), S. 620; Silver (2004), S. 945; Fink/Voß (2003), S. 396. Für einen Definitionenüberblick vgl. Blum/Roli (2008), S. 5.
[114] Für einen allgemeinen Überblick vgl. Prins et al. (2010); Dréo et al. (2006); Glover et al. 2003; Reeves 1993.
[115] Vgl. Laarhoven/Aarts (1987).
[116] Vgl. Bäck et al. (2000b); Bäck et al. (2000a); Hertz/Kobler (2000); Weicker (2007); Nissen (1997); Goldberg (1989).
[117] Vgl. Glover/Laguna (1997).
[118] Vgl. Dorigo/Stützle (2004).
[119] Vgl. Eberhart et al. (2001).

gie). Meta-Heuristiken können u.a. nach der Art ihres Suchprozesses (deterministisch/stochastisch) und nach Anzahl gleichzeitig betrachteter Lösungen (Einzellösung/Population von Lösungen) klassifiziert werden.[120] Für den konkreten Einsatz sind die heuristischen Operatoren der Meta-Heuristiken an das konkrete Problem anzupassen.

Da für jedes Problem unendlich viele heuristische Lösungsverfahren möglich sind, ist zu klären, wie ein konkret vorgeschlagenes Verfahren zu bewerten ist. Während bei exakten Verfahren die Laufzeit das entscheidende Kriterium darstellt, sind bei Heuristiken mit Laufzeit und Lösungsqualität zwei Kriterien heranzuziehen.[121] Ein Ansatz zur Bewertung der Lösungsqualität ist eine Abschätzung der Worst Case Qualität.[122] Hierbei wird eine Untergrenze bestimmt, die mindestens von der Heuristik erreicht wird, z.B. mindestens 80% der optimalen Lösung. Allerdings spielt diese Art der Qualitätsbestimmung nur eine untergeordnete Rolle. Zum einen ist die Bestimmung der Worst Case Qualität schwierig und in manchen Fällen nicht möglich, zum anderen ist in praktischen Problemstellungen nicht die Worst Case, sondern die durchschnittlich zu erwartende Lösungsgüte von größerer Bedeutung.[123] Diese kann jedoch nicht durch Berechnung, sondern nur durch umfangreiche empirische Rechenexperimente erfolgen.[124] Zur Bewertung der Average Case Qualität kommen grundsätzlich drei Vergleiche in Betracht.[125] Die heuristisch ermittelte Lösung kann erstens mit oberen bzw. unteren Schranken, zweitens mit der optimalen Lösung und drittens mit den Ergebnissen anderer Heuristiken verglichen werden. Der Vergleich mit einer optimalen Lösung kann nur erfolgen, sofern diese auch bestimmbar ist. Ist die optimale Lösung nicht für alle Probleminstanzen bestimmbar, so kann ein Vergleich für optimal lösbare Instanzen erfolgen. Mit statistischen Methoden ist zu prüfen, ob ein Zusammenhang zwischen Lösungsqualität und Umweltfaktoren existiert.[126] Im Idealfall ist die Lösungsqualität unabhängig von den Umweltfaktoren und somit für alle betrachteten Instanzen identisch. Dann kann davon ausgegangen werden, dass die Heuristik auch für größere Instanzen ähnlich gute Ergebnisse erzielt.

[120] Für ein umfangreicheres Klassifikationsschema vgl. Müller-Merbach (1981), S. 10-19.
[121] Vgl. Rardin/Uzsoy (2001), S. 273; Hooker (1995), S. 39; Berens (1992), S. 114–124.
[122] Vgl. Fisher (1980); Werners (2008), S. 12.
[123] Vgl. Reeves (1993), S. 306; Werners (2008), S. 11f..
[124] Vgl. Dréo et al. (2006), S. 243. Zur Evaluation von Heuristiken etwa Rardin/Uzsoy (2001); Barr et al. (1995); Hooker (1995); Berens (1992).
[125] Vgl. Silver (2004), S. 952.
[126] Vgl. Reeves (1993), S. 309.

3.1.2 Grundidee evolutionärer Algorithmen

Evolutionäre Algorithmen sind stochastische Suchverfahren, die der Darwinschen Evolution[127] nachempfunden sind.[128] Ausgehend von einer Vielzahl von Lösungen werden schrittweise Verbesserungen auf Basis von Variation und Selektion erzielt. Heutzutage wird unter evolutionären Algorithmen die früher getrennten Bereiche genetische Algorithmen[129], Evolutionsstrategien[130], evolutionäre Programmierung[131] und genetische Programmierung[132] verstanden.[133] Da diese jeweils sehr unterschiedlich ausgestaltet sein können, kann nicht von dem einen evolutionären Algorithmus gesprochen werden.[134] Selbst ein Überblick über die gängigsten Implementierungsmöglichkeiten würde den Rahmen dieses Kapitels sprengen, weswegen auf Einführungen in der Literatur verwiesen wird.[135] Trotz ihrer Unterschiede basieren alle Algorithmen auf demselben Prinzip und weisen einen ähnlichen Ablauf auf (vgl. Abbildung 3.1).

Wie bereits erwähnt, arbeiten evolutionäre Algorithmen nicht mit einer, sondern mit mehreren Lösungen.[136] Eine Lösung des Optimierungsproblems wird als *Individuum*[137], die Menge aller Lösungen als *Population* bezeichnet. Jedes Individuum wird durch ein *Genom* bzw. *Chromosom* charakterisiert, welches als String die konkreten Werte aller Entscheidungsvariablen enthält. Ein Zeichen eines Chromosoms heißt *Gen*, welches verschiedene Ausprägungen (*Allele*) annehmen kann. Die durch eine solch kodierte Lösung gegebene Handlungsalternative wird als *Phänotyp* bezeichnet, welcher von der formalen Kodierung, dem sogenannten *Genotyp*, abweichen kann. Die Gesamtheit aller zulässigen Lösungen stellt den *Suchraum* dar, in dem das an eine *künstliche Umgebung*, gegeben durch eine Zielfunktion, bestangepasste Individuum zu

---

127 Vgl. Darwin (1859).
128 Vgl. VDI/VDE-Richtlinie 3550, Blatt 3; Beyer et al. (2001), S. 3, Pohlheim (2000), S. 7.
129 Vgl. u.a. Holland (1975); Goldberg (1989); Michalewicz (1996) für die Grundlagen und u.a. Davis (1991) für die Anwendung.
130 Vgl. Rechenberg (1973); Schwefel (1977) für die Grundlagen und Beyer (2001) für aktuellere Entwicklungen.
131 Vgl. Fogel et al. (1966) für die Grundlagen undFogel (1992) für eine Weiterentwicklung.
132 Im Gegensatz zu den anderen Bereichen entwickelten sich die genetische Programmierung erst nach Etablierung des Begriffs Evolutionäre Algorithmen als eines seiner Teilgebiete. Vgl. Koza (1992); Koza (1994); Koza et al. (1999).
133 Vgl. Dréo et al. (2006), S. 76 und S. 113; Gerdes et al. (2004), S. 2; Weicker (2007), S. 127f. Für einen kurzen historischen Überblick über die Entwicklung vgl. etwa Weicker (2007), S. 44–46 und S. 180-182 sowie de Jong (2006), S. 23–32.
134 Vgl. Pohlheim (2000), S. 12.
135 Für eine praktisch orientierte Einführung vgl. etwa Pohlheim (2000); Eiben/Smith (2003); Weicker (2007). Für einen tiefergehenden wissenschaftlich Einblick vgl. etwa Bäck et al. 1997; Bäck et al. (2000b); Bäck et al. (2000a); Rothlauf (2006); de Jong (2006). Für eine literaturorientierte Einführung vgl. etwa Nissen (1997).
136 Die folgende kurze Beschreibung des Ablaufs basiert auf der genannten einführenden Literatur.
137 Für eine weitergehende Erläuterung kursiv gedruckte Begriffe vgl. VDI/VDE-Richtlinie 3550, Blatt 3; Beyer et al. (2001), S. 3 oder die einführende Literatur.

identifizieren ist. Dies geschieht in mehreren Iterationen (sogenannten *Generationen*) durch Wettbewerb der Individuen untereinander.

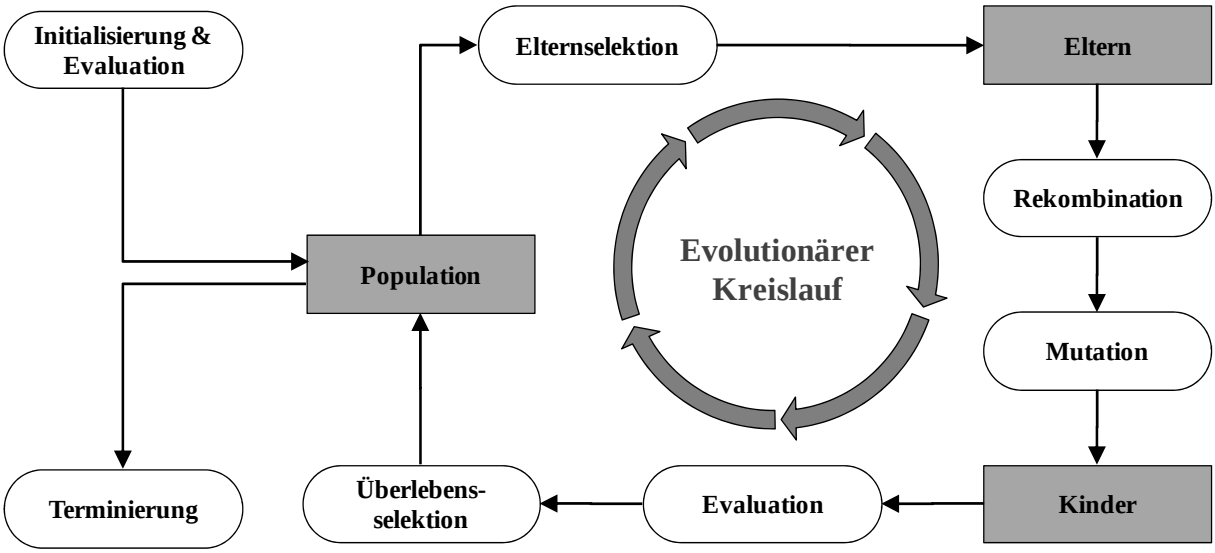

Abbildung 3.1: Schematische Darstellung des Ablaufs von evolutionären Algorithmen
Quelle: In Anlehnung an Eiben/Smith (2003), S. 17.

Dazu ist zu Beginn der Optimierung die Ausgangspopulation zu initialisieren. Die Gene jedes Chromosoms werden dabei auf ein konkretes Allel festgelegt. Dies geschieht meist zufällig, um viele unterschiedliche Individuen zu erhalten und damit möglichst alle Bereiche des Suchraums abzudecken. Anschließend ist die Anpassungsgüte (*Fitness*) jedes Individuums zu bestimmen. Dies geschieht mit Hilfe einer *Fitnessfunktion*, die meist, aber nicht notwendigerweise identisch mit der *Zielfunktion* ist und jedem Individuum eine Zahl als Fitnesswert zuweist. Sie muss in der Lage sein, Individuen in eine Reihenfolge zu bringen und sollte aufgrund häufiger Aufrufe schnell auszuwerten sein.

Ausgehend von der Startpopulation findet eine Selektion statt, bei der Individuen als *Eltern* zur *Fortpflanzung* ausgewählt werden. Diese Selektion wird daher auch als *Elternselektion* oder *Paarungsselektion* bezeichnet. Die Selektion erfolgt meist stochastisch, indem jedem Individuum auf Basis seiner Fitness eine Fortpflanzungswahrscheinlichkeit zugeordnet wird. Dadurch besitzen auch Individuen mit niedriger Fitness eine, wenn auch geringe Chance ausgewählt zu werden. Individuen können mehrfach ausgewählt werden, wodurch Individuen mit hohen Fitnesswerten überproportional häufig ihre Gene an ihre *Nachkommen* bzw. *Kinder* übergeben können. Dadurch wird ein *Selektionsdruck* auf die Population ausgeübt und die Suche auf aussichtsreiche Bereiche des Suchraums gelenkt.

Durch *Rekombination*, teilweise als *Crossover* bezeichnet, werden aus Eltern Nachkommen erzeugt, deren Chromosomen aus den Chromosomen der Eltern zusammen-

gesetzt sind und damit u.U. eine höhere Fitness besitzen. Die zum Einsatz kommenden stochastischen Rekombinationsoperatoren hängen stark von dem Problem, der Kodierung und dem Einsatzzweck ab und sind daher immer problemspezifisch zu konstruieren. Im Gegensatz zur biologischen Evolution besteht die Möglichkeit, Nachkommen nicht nur aus zwei, sondern auch aus mehreren Eltern zu erzeugen. Durch Rekombination wird die Suche in der Nähe von Individuen mit hoher Fitness intensiviert, die *Exploitation* (Verwertung) aussichtsreicher Bereiche des Suchraums demnach verstärkt.

Durch Rekombination werden die Nachkommen einander ähnlicher sein als ihre Eltern. Die *genetische Vielfalt* oder *Diversität* nimmt ab, was dazu führen kann, dass der Suchprozess ein lokales Optimum nicht mehr verlässt, was *vorzeitige Konvergenz* genannt wird. Zum Erhalt der genetischen Vielfalt werden die Nachkommen daher einer *Mutation* unterzogen. Hierbei handelt es sich um kleine Veränderungen des Chromosoms, die mit einer geringen Wahrscheinlichkeit auftreten. Dadurch wird die *Exploration* gestärkt, indem neue bisher nicht erreichte und durch Rekombination nur schwierig erreichbare Bereiche des Suchraums untersucht werden.

Im Anschluss werden aus den Individuen der bisherigen Population und der erzeugten Nachkommen entsprechend der Güte ihrer Umweltanpassung die Individuen ausgewählt, welche die aktuelle Generation überleben und in die nächste Population (wieder-)eingefügt werden. Diese Selektion wird daher als *Wiedereinfügen* oder *Umwelt- bzw. Überlebensselektion* bezeichnet. Dazu sind im Vorfeld alle Individuen mit einer Fitnessfunktion zu bewerten, die nicht notwendigerweise mit der Fitnessfunktion für die Elternselektion identisch sein muss. Wie die Elternselektion erfolgt auch die Überlebensselektion meist stochastisch, indem jedem Individuum analog eine Überlebenswahrscheinlichkeit zugeordnet wird. Ein Individuum kann jedoch nur einmal für die neue Population ausgewählt werden.

Bevor eine neue Generation und damit eine neue Elternselektion beginnt, wird das Eintreten von *Abbruch- bzw. Terminierungsbedingungen* geprüft. Im Gegensatz zur biologischen Evolution haben evolutionäre Algorithmen ein festgelegtes Ende. Neben einer festgelegten Anzahl an Generationen werden meist aus dem Lauf abgeleitete Kriterien wie laufende Mittelwerte verwendet. Nach Abbruch können neben dem Individuum mit der höchsten Fitness auch die weiteren Individuen der Population ausgegeben werden.

3.1.3 Grundidee der Hybridisierung von Heuristiken

Der Einsatz evolutionärer Algorithmen weist einige Vorteile auf. So wird in der Literatur, neben der einfachen Implementierbarkeit und des flexiblen Einsatzes für unterschiedliche Probleme, vor allem die Fähigkeit zur Identifizierung aussichtsreicher Bereiche selbst in hochdimensionalen Suchräumen hervorgehoben.[138] Die Auswertung aussichtsreicher Bereiche hingegen erfolgt durch die künstliche Evolution langsamer als bei trajektorischen[139] Verfahren, die jeweils eine Ausgangslösung verbessern.[140] Schon früh nach Entwicklung der Metaheuristiken kam daher die Idee auf, die Stärken verschiedener Suchverfahren zu nutzen und die Verfahren zu integrieren.[141] Dies wird als Hybridisierung bezeichnet, die insbesondere in den letzten Jahren nicht zuletzt aufgrund der Erkenntnisse des „No free lunch"-Theorems[142] an Bedeutung zugenommen hat.[143] Prinzipiell soll durch Hybridisierung die Performance (Konvergenzgeschwindigkeit) und/oder die Lösungsqualität verbessert werden.[144]

Inzwischen existieren auch einige theoretische Arbeiten zur Klassifizierung der verschieden Hybridisierungsformen, von denen sich aber noch keines endgültig durchgesetzt hat.[145] Ein akzeptiertes Unterscheidungsmerkmal ist die *Kopplungsstärke*.[146] Bei starker Kopplung verlieren die Verfahren ihre Eigenständigkeit und verschmelzen zu einem neuen Verfahren, meist indem auf unterster Ebene Funktionen des einen Verfahrens in das andere eingebettet (embedded) bzw. integriert werden. Es wird daher auch von low-level Hybridisierung[147] bzw. von embedded oder integrated combinations[148] gesprochen. Bei schwacher Kopplung hingegen sind beide Verfahren eigenständig. Die Kooperation erfolgt auf einer höheren Ebene, indem die Verfahren

---

138 Vgl. Grosan/Abraham (2007), S. 1; Blum/Roli (2008), S. 1f. und 17.

139 Lösungsverfahren, die schrittweise eine Lösung verbessern, bilden einen Lösungspfad durch den Suchraum (im engl. Trajectory), vgl. Blum/Roli (2008), S. 6; Blum et al. (2005), S. 8–18

140 Vgl. Krasnogor/Smith (2005), S. 474; Blum/Roli (2008), S. 18.

141 Mit Bosworth et al. (1972) als einer der ersten Ansätze.

142 Vgl. Wolpert/Macready (1997). Speziell für evolutionäre Algorithmen vgl. Culberson (1998). Das No free lunch-Theorem besagt, dass allgemein anwendbare Algorithmen bei Zugrundelegung aller mathematisch möglichen Optimierungsproblemen im Durchschnitt gleich gute Ergebnisse erzielen. Daraus kann abgeleitet werden, möglichst viel problemspezifisches Wissen bei der Entwicklung von Algorithmen zu verwenden.

143 Vgl. Talbi (2002), S. 541; Glover et al. (2003), S. 2; Raidl (2006), S. 2; Blum/Roli (2008), S. 17. Zudem existiert mit „Hybrid Metaheuristics" inzwischen eine eigenständige Konferenzenreihe, die sich ausschließlich dieser Thematik widmet und deren Proceedings einen Überblick über aktuelle Entwicklungen geben, vgl. Blesa et al. (2008); Bartz-Beielstein et al. (2007); Almeida et al. (2006); Blesa Aguilera et al. 2005; Blum et al. 2004. Allein für die Keywords „hybrid evolutionary" und „hybrid genetic" haben Grosan und Abraham mehr als 5000 bzw. mehr als 11000 Artikel in den wissenschaftlichen Datenbanken ScienceDirect und SpringerLink gefunden, vgl. Grosan/Abraham (2007), S. 3.

144 Vgl. Grosan/Abraham (2007).

145 Weitere Klassifizierungsschemata teilweise mit Einordnung zahlreicher Artikel werden vorgeschlagen von Talbi (2002); Cotta et al. (2005); Grosan/Abraham (2007); Jourdan et al. (2009).

146 Vgl. Raidl (2006), S. 3–5.

147 Vgl. Talbi (2002), S. 543; Jourdan et al. (2009), S. 622f..

148 Vgl. Puchinger/Raidl (2005), S. 42; Blum/Roli (2008), S. 2.

Informationen und Ergebnisse austauschen, was daher auch als high-level Hybridisierung[149] bzw. collaborative combinations[150] bezeichnet wird.

Bei starker Kopplung bestehen zwei *Einbettungsformen*, wie Funktionen von Verfahren B in Verfahren A integriert werden können. Verfahren A kann um Funktionen von B ergänzt werden (Funktionsergänzung) oder Funktionen von A werden durch die Einbettung von B modifiziert (Funktionsmodifikation).[151]

Bei schwacher Kopplung können die eigenständigen Verfahren gleichberechtigt sein (keine *Hierarchie*) oder ein Verfahren steuert das andere (Hierarchie). In letzterem Fall führt meist das untergeordnete Verfahren die Optimierung durch, während das übergeordnete Verfahren (auf Basis der Zwischenergebnisse) die Parametereinstellungen des untergeordneten vornimmt.

Bei gleichberechtigten Verfahren ist die *Einsatzreihenfolge* festzulegen. Diese kann entweder sequentiell, iterativ oder simultan erfolgen.[152] Beim sequentiellen Einsatz dient ein Verfahren zur Vor- (Preprocessing) oder Nachoptimierung (Fine Tuning) des anderen. Beim iterativen Einsatz werden die beiden Verfahren abwechselnd eingesetzt und die Ergebnisse jeweils ausgetauscht, während beim simultanen Einsatz beide Verfahren gleichzeitig zum Einsatz kommen. Dabei wird teilweise auch von Parallelisierung gesprochen, was inzwischen ein eigenständiges Forschungsgebiet darstellt.[153]

Weitere Klassifizierungsmerkmale sind *Homogenitätsgrad*, *Spezialisierung* und *Suchraumaufteilung*.[154] Dabei wird unterschieden, ob gleich- (homogene) oder verschiedenartige (heterogene) Verfahren kombiniert werden, ob die Verfahren in der Lage sind, das Gesamtproblem (Generalisten) zu lösen oder auf die Lösung von Teilproblemen (Spezialisten) spezialisiert sind und ob die Verfahren den gesamten Suchraum (global) oder nur einen Teil (partial) durchsuchen.

Im Kontext evolutionärer Algorithmen hat sich besonders die Hybridisierung mit lokalen Suchverfahren etabliert, welche von einigen Autoren als memetische Algorithmen bezeichnet werden.[155] Der Begriff *Mem* geht auf den Biologen Richard Dawkins zurück, der damit Verhaltenselemente bezeichnet, die sich im Gegensatz zu Genen indi-

[149] Vgl. Talbi (2002), S. 543; Jourdan et al. (2009), S. 623f..
[150] Vgl. Puchinger/Raidl (2005), S. 42; Blum/Roli (2008), S. 2.
[151] Talbi hingegen spricht von Relay und Teamwork, vgl. Talbi (2002), S. 543–546.
[152] Vgl. Puchinger/Raidl (2005), S. 2; Raidl (2006).
[153] Vgl. Alba (2005); Crainic/Toulouse (2003).
[154] Vgl. Talbi (2002), S. 548f.; Jourdan et al. (2009), S. 624f..
[155] Vgl. Moscato (1999), S. 219; Hart et al. (2005), S. 3; Krasnogor/Smith (2005), S. 475; Grosan/Abraham (2007), S. 5. Auch wenn sich dieser von Moscato geprägte Begriff noch nicht abschließend durchgesetzt hat, erfährt er doch zunehmende Resonanz, vgl. Hart et al. (2004); Yew-Son Ong et al. (2007); Ong et al. (2009). 2010 wird die erste Ausgabe des neuen „Journals of Memetic Computing“ erscheinen. Für eine Einführung vgl. u.a. Moscato (1999); Moscato/Cotta (2003); Hart et al. (2005); Krasnogor/Smith (2005).

viduell im Leben eines Individuums ändern können, indem sie beispielsweise durch Nachahmung erworben werden.[156] Die Individuen erhalten damit die Fähigkeit zu lernen und sich individuell zu verbessern. Algorithmisch geschieht dies durch die lokale Optimierung eines neu erzeugten Individuums vor dessen Fitnessbewertung.[157] Evolutionstheoretisch basieren memetische Algorithmen meist auf der Theorie von Lamark[158], die Veränderungen im Genotyp auf individuelles Lernen zurückführt.[159] Biologisch wurde dieser direkte Zusammenhang im Gegensatz zum indirekt wirkenden Baldwin-Effekt[160] zwar widerlegt, die zunehmende Verbreitung belegt jedoch die Praxistauglichkeit dieses Konzepts.[161]

3.2 Kodierung und Hybridisierung

3.2.1 Kodierung und Problemdekomposition

Für die Entwicklung eines evolutionären Algorithmus ist zunächst die Kodierung bzw. Repräsentation der Lösung festzulegen. Dies ist eine bedeutsame, wenn nicht sogar die für den erfolgreichen Einsatz bedeutsamste Entscheidung.[162] Im Rahmen der Produktprogrammoptimierung sind unterschiedliche Elemente und ihre Beziehungen zueinander zu kodieren. So existieren mehrere Produktlinien, deren Varianten durch unterschiedliche Merkmale und Ausprägungen beschrieben werden und die wiederum zu Bündel kombiniert und anschließend bepreist werden können. Hierfür sind zwei interdependente, der Produkt- bzw. Preispolitik zugeordnete Entscheidungskomplexe abzugrenzen. Zum einen ist produktpolitisch das Design, zum anderen preispolitisch das Preissystem festzulegen. Im Rahmen der Produktprogrammoptimierung wird Design wie folgt definiert:

Design bezeichnet die Angabe aller gestalterischen Entscheidungen, insbesondere die konkrete Angabe aller gewählten Merkmalsausprägungen.

In diesem Zusammenhang bezeichnet Produkt- bzw. Variantendesign die Festlegung der Merkmalausprägungen einer betrachteten Variante, während Produktlinien-Design die Angabe des Produktdesigns aller Varianten einer Linie umfasst. Unter Bündel-Design wird die Angabe, der in einem Bündel enthaltenden Varianten sowie deren

[156] Vgl. Dawkins (1976), S. 192ff.
[157] Vgl. Krasnogor/Smith (2005), S. 475.
[158] Vgl. de Lamarck (1809).
[159] Vgl. Krasnogor/Smith (2005), S. 484.
[160] Vgl. Baldwin (1896).
[161] Vgl. Weicker (2007), S. 14f. für eine Kurzbeschreibung beider Theorien. Für einen empirischen Vergleich lamarkscher und baldwinischer Implementierungen vgl. Whitley et al. (1994) und für einen Überblick erfolgreicher Anwendungen vgl. u.a. Moscato (1999), S. 220; Moscato (2002).
[162] Vgl. Gerdes et al. (2004).

Produktdesigns festzulegen, verstanden. Abschließend definiert Design allgemein das gesamte Produktprogramm. Demgegenüber steht das Preissystem bzw. verkürzt die Preise:

Das Preissystem bezeichnet die konkrete Angabe aller Bündelpreise.

Kunden werden maximal ein Bündel erwerben, weswegen nur für diese ein Preis zu bestimmen ist. Es sei angemerkt, dass es sich bei den Bündeln auch um „unechte" Bündel handeln kann, die nur aus einem Produkt bestehen, wodurch auch Einzelprodukte nachgefragt werden können.

Bei der Kodierung der Lösungen kann das Genom entweder aus einem String fester oder variabler Länge bestehen.[163] Bei einem Genom mit fester Länge sind die Anzahl der Gene und damit die Anzahl kodierter Entscheidungsvariablen konstant. Bei Genomen variabler Länge hingegen variieren diese, wodurch die Entwicklung und Kalibrierung von Rekombinationsoperatoren deutlich schwieriger ist, weswegen Genome fester Länge üblicher sind.

In vielen Fällen werden die Begriffe Chromosom und Genom synonym verwendet, was impliziert, dass ein Genom aus genau einem Chromosom besteht. Biologisch gesehen besteht hingegen ein Genom aus mehreren Chromosomen.[164] Aufgrund der hier betrachteten Vielzahl unterschiedlicher Entscheidungskomplexe bietet es sich an, dieser Auffassung zu folgen und ein Genom als Gesamtheit mehrerer Chromosomen zu begreifen. Im entwickelten Algorithmus existieren zwei unterschiedlich aufgebaute Arten von Chromosomen; zum einen Bündel-Chromosomen, zum anderen Produktlinien-Chromosomen. Das sich ergebende Genom kann als String hintereinandergelegter Chromosomen interpretiert werden, wie es Abbildung 3.2 veranschaulicht.

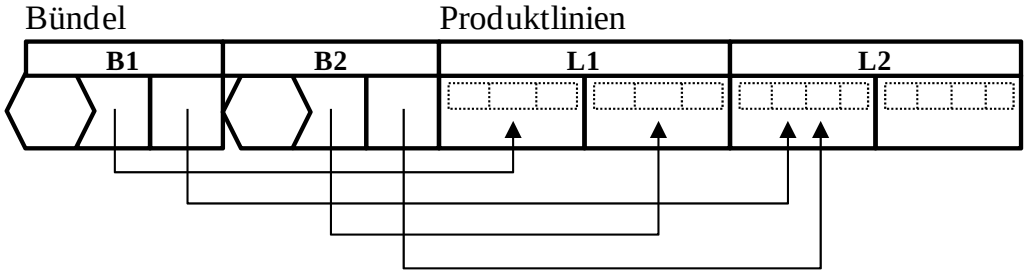

Abbildung 3.2: Schematische Darstellung eines Genoms mit 2 Bündel- und 2 Produktlinien-Chromosomen

Die Bündel-Chromosomen enthalten die Informationen über das Bündel-Design sowie die Bündelpreise. Es besteht als erstes aus einem Gen für den Preis des Bündels. Um Chromosomen mit variabler Länge zu vermeiden, enthält es im Weiteren nicht ein Gen

[163] Vgl. de Jong (2006), S. 72–75; Weicker (2007), S. 37.
[164] Vgl. Janning/Knust (2004), S. 11f.; Passarge/Kohlhase (2006), S. 5.

für jedes im Bündel enthaltene Produkt, sondern ein Gen für jede Produktlinie, unabhängig vom Vorhandensein einer Variante dieser Linie. Die Gene sind geordnet, zunächst das Gen für die erste Produktlinie, dann das für die zweite Linie usw. Das Preis-Gen enthält einen ganzzahligen Wert, der je nach Auflösung den Preis in Euro- oder Cent-Beträgen repräsentiert. Die Produktlinien-Gene sind entweder leer oder enthalten einen ganzzahligen Wert. Im ersteren Fall ist keine Variante der entsprechenden Produktlinie im Bündel enthalten, im letzteren Fall gibt der Wert die ID der enthaltenen Variante an. Dadurch wird eine Verbindung zwischen Bündel- und Produktlinien-Chromosomen hergestellt, die in Abbildung 3.2 als Verbindungspfeile dargestellt sind.

Produktlinien-Chromosomen enthalten hingegen keine Preis-, sondern ausschließlich Design-Informationen. Sie bestehen aus mehreren identisch aufgebauten Teil-Chromosomen bzw. Chromosomenfragmenten, die jeweils eine Variante der Produktlinie beschreiben. Jedes Fragment besteht aus einem Gen für jedes den Produkttyp beschreibende Merkmal. Die möglichen Allele des Gens sind jeweils die möglichen Ausprägungen des Merkmals. Durch die separate Kodierung der Produktlinien ist es möglich, die Anzahl verschiedenartiger Varianten pro Linie zu begrenzen. Wird im Weiteren eine Variante modifiziert, wirkt sich dies auch auf alle Bündel aus, welche die Variante enthalten.

Zum besseren Verständnis sollte hier jedoch das Genom nicht als flacher String, sondern vielmehr als semantische Hierarchie interpretiert werden, wie sie in Abbildung 3.3 dargestellt ist.[165]

Ein Individuum besteht aus mehreren Bündeln, welche aus jeder Produktlinie maximal eine Variante enthalten, die durch Merkmale und deren Ausprägungen beschrieben wird. Varianten hingegen sind nicht bündelspezifisch, sondern können in mehreren Bündeln enthalten sein. Ebenso ist es nicht erforderlich, dass aus jeder Produktlinie eine Variante in den Bündeln enthalten ist. Da als Fitnessfunktion der Gesamtdeckungsbeitrag verwendet wird, werden nur die Produktdesigns der in gekauften Bündeln enthaltenen Varianten bewertet. Besitzt ein Individuum ein optimal konfiguriertes Bündel mit optimal gestalteten Varianten, so wird dieses nicht bewertet, falls durch ein falsches Preissystem das Bündel nicht gekauft wird. Damit dies geschieht, bedarf es eines passenden Preissystems.

[165] Eine ähnliche Interpretation ist bei Jiao et al. (2007) und Man et al. (1999), S. 65–74 zu finden.

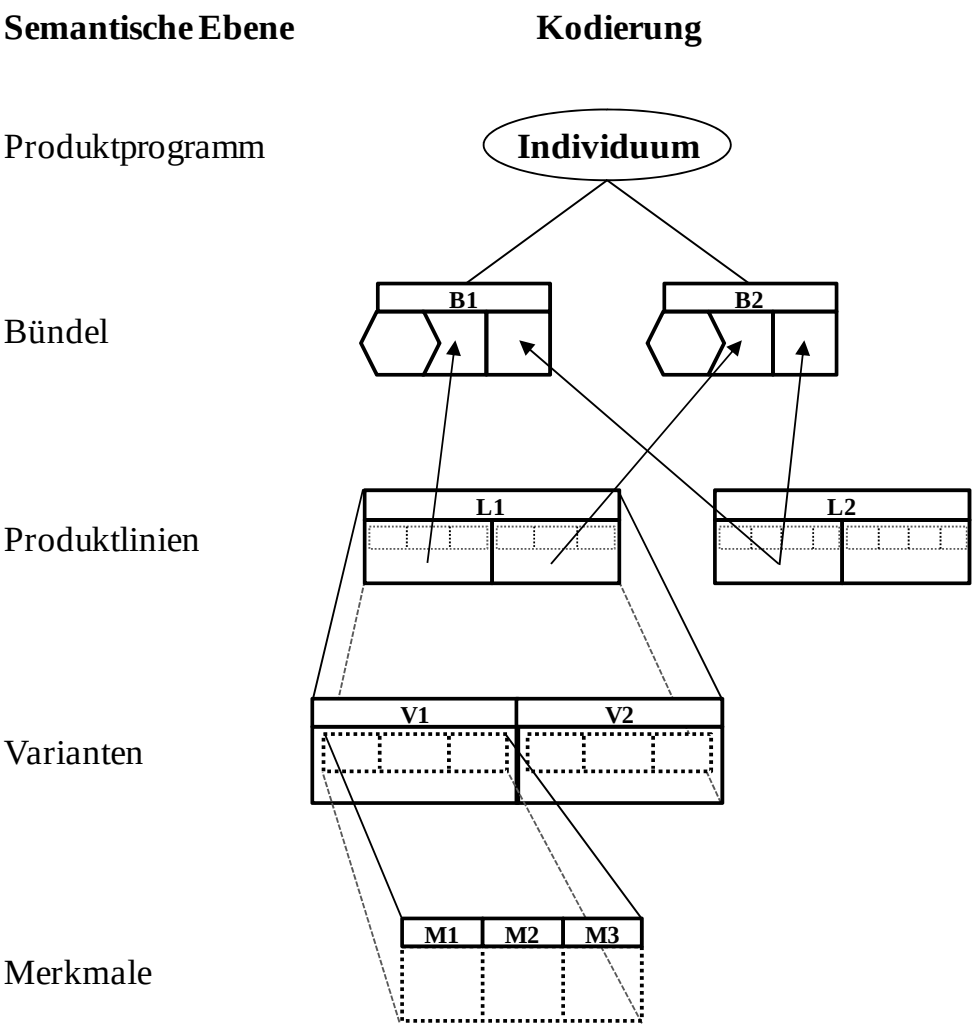

Abbildung 3.3: Schematische Darstellung der semantischen Hierarchie eines Genoms

Die Wichtigkeit eines passenden Preissystems für die Fitnessbewertung sei an folgenden Überlegungen verdeutlicht. Zu jedem Design Ψ existiert ein Preissystem $\Pi^*(\Psi)$, welches den mit diesem Design erzielbaren Deckungsbeitrag maximiert. Ein solches Individuum wird durch die Fitnessfunktion Γ einen größeren Wert erhalten als jedes andere Individuum mit demselben Design jedoch einem anderen Preissystem. Es gilt: $\Gamma[\Pi^*(\Psi)] \geq \Gamma[\Pi(\Psi)]$. Dies stellt jedoch ein lokales Optimum dar, welches nur durch Wahl eines anderen Designs verlassen werden kann. Gleichzeitig existiert ein optimales Design, welches zusammen mit dem passenden Preisschema das globale Optimum angibt. Es gilt: $\Gamma[\Pi^*(\Psi^*)] \geq \Gamma[\Pi^*(\Psi)] \geq \Gamma[\Pi(\Psi)]$. Problematisch hingegen ist, dass ein Individuum mit dem optimalen Design bei Verwendung eines anderen Preissystems nicht besser bewertet werden muss als ein Individuum mit einem anderen Design. Es kann gelten: $\Gamma[\Pi(\Psi^*)] < \Gamma[\Pi^*(\Psi)]$. Dies führt dazu, dass das Individuum mit dem eigentlich überlegenen Design eine geringere Paarungs- und Überlebenswahrscheinlichkeit zugeordnet wird. Zur „richtigen“ Bewertung eines Individuums muss dieses daher

über ein zu seinem Design passendes Preissystem verfügen. Ansonsten besteht die Gefahr, die falschen Individuen zu selektieren und das globale Optimum zu verfehlen.

Die Folgen einer solchen falschen Bewertung waren die Ursache dafür, dass die erste Implementierung in Form eines reinen evolutionären Algorithmus nicht zu den gewünschten Ergebnissen geführt hat. Der Algorithmus konvergierte zu schnell und erreichte nur lokale Optima. Das Hauptproblem bestand in der mangelnden Fähigkeit reiner evolutionärer Algorithmen, Preissystem und Design gleichzeitig zu optimieren. Durch evolutionäre Operatoren wie Rekombination und Mutation werden Design **und** Preise verändert. In den seltensten Fällen jedoch so, dass sie aufeinander abgestimmt sind. Durch die Fitnessbewertung erhielt der evolutionäre Algorithmus jedoch nicht die zur Eingrenzung der Suche notwendigen Informationen. Es konnte nicht unterschieden werden, ob eine schlechte Bewertung eines Individuums aufgrund eines schlechten Designs oder eines schlechten Preissystems erfolgte.

Zur Lösung des Problems kommt neben einem angepassteren Preissystem auch eine bessere Bewertung der Individuen durch eine andere Fitnessfunktion in Betracht. Die Fitnessfunktion müsste in der Lage sein, die Individuen auf Basis ihres Designs zu bewerten. Dabei ist sicherzustellen, dass Individuen umso besser bewertet werden, je näher sie am optimalen Design sind. Da dieses a priori jedoch nicht bekannt ist und zudem von den konkreten Parametern abhängt, erschien dieser Ansatz wenig erfolgreich und wurde verworfen. Stattdessen wurde eine Problemdekomposition durchgeführt, um das zu einem gegebenen Design optimale Preissystem bestimmen zu können. Bei einer Dekomposition wird das Gesamtproblem in zueinander in Beziehung stehende Teilprobleme zerlegt.[166] Konkret wird das Gesamtproblem in ein *Design-Masterproblem* und ein *Pricing-Subproblem* zerlegt. Das *Design-Masterproblem* stellt das übergeordnete Problem dar. Für ein konkretes Design soll dann innerhalb des *Pricing-Subproblem* das optimale Preissystem identifiziert werden.

3.2.2 Integration von evolutionärem Algorithmus und Preisbestimmung

Die Umsetzung der Hybridisierung erfolgte in Form eines memetischen Algorithmus. Basierend auf diesem Konzept besitzt ein Individuum in Genen kodierte Design-Informationen und „lernt“ das dazu passende Preissystem. Das gelernte Preissystem wird beim Individuum gespeichert, sodass von einer Implementierung der lamarkschen Theorie gesprochen werden könnte. Dies wäre jedoch falsch. Die Preis-Informationen sind zwar im Genom kodiert, werden aufgrund der Problemdekomposition nicht durch

[166] Vgl. Großmann/Terno (1997), S. 297–317; Williams (1999), S. 48–58; Kistner/Steven (2001), S. 311f.; Zimmermann (2008), S. 275f. und 313-322; Hagelschuer (1971).

den evolutionären Algorithmus verwendet und durch Rekombination und Mutation verändert. Die gelernten Informationen gehen damit nicht in die vom evolutionären Algorithmus verwendeten Teile des Genoms über. Es besteht daher im Gegensatz zur Grundidee memetischer Algorithmen eine deutlich stärkere Abgrenzung zwischen Genen und Memen. Für gewöhnlich dienen Meme zur Verbesserung des sich aus dem Genom ergebenden Phänotyps. Hier ergibt sich durch Lernen jedoch erst das zur Bildung des Phänotyps notwendige Preisschema. Lernen ist damit ein unverzichtbarer Bestandteil der Bewertung eines Individuums. Aus diesem Grund ist auch eine Einordnung in die baldwinsche Theorie nicht ganz zutreffend.

Der Ablauf des entwickelten hybriden Algorithmus weist gegenüber reinen evolutionären Algorithmen einen zusätzlichen Lernschritt auf. Wie in Abbildung 3.4 dargestellt, erfolgt dieser vor jeder Evaluierung, indem für jedes Individuum das zu seinem Design passende Preissystem bestimmt wird.

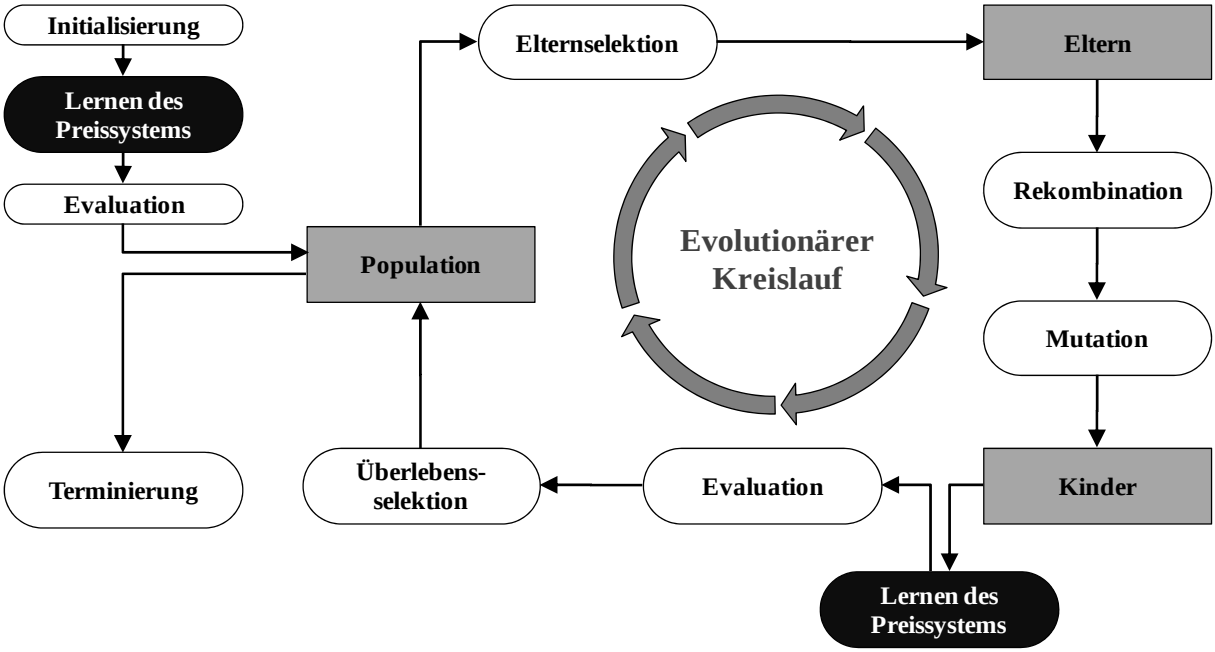

Abbildung 3.4: Schematische Darstellung des Ablaufs des hybriden evolutionären Algorithmus

Zur Bestimmung des passenden Preissystems wird dieselbe Zielfunktion, nämlich der Gesamtdeckungsbeitrag, verwendet. Dadurch ergibt sich durch die Bestimmung des Preissystems simultan der Gesamtdeckungsbeitrag. Es bietet daher einen Geschwindigkeitsvorteil, wenn der vom Preisbestimmungsverfahren ermittelte Gesamtdeckungsbeitrag als Zielfunktionswert verwendet wird und somit das Preisbestimmungsverfahren die Evaluation übernimmt. Im Rahmen der Selektion wird zur Bestimmung der Selektionswahrscheinlichkeit eine Fitnesstransformation durchgeführt. Der entwickelte Algorithmus führt evolutionäre Algorithmen und im Weiteren zu spezifizierende Preisbestimmungsverfahren zusammen. Er kann aufgrund der passenden Lernana-

logie zur Unterklasse der memetischen Algorithmen gezählt werden, auch wenn durch die stärkere Trennung und die nicht notwendigerweise auf lokalen Suchverfahren beruhende Preisbestimmung deutliche Unterschiede darstellen. Dies erklärt auch die unterschiedliche Einordnung. Allgemein werden memetische Algorithmen als stark gekoppelte Hybride klassifiziert, bei der die Funktionen eines evolutionären Algorithmus durch lokale Suchverfahren modifiziert werden.[167] Im vorgeschlagenen Algorithmus hingegen behalten evolutionäre Algorithmen und Preisbestimmungsverfahren ihre eigenständige Identität bei, wobei das Preisbestimmungsverfahren die lokalen Optima bestimmt bzw. approximiert und der evolutionäre Algorithmus auf der Menge der lokalen Optima operiert. Sie tauschen in schwacher Kopplung Design- bzw. Preisinformationen aus. Da keines der Verfahren das andere steuert, sind beide gleichberechtigt und es existiert keine Hierarchie. Die Verfahren werden abwechselnd iterativ eingesetzt. Die Verfahren sind heterogen und lösen als Spezialisten unterschiedliche Teilprobleme, wodurch sie zwangsläufig partial andere Ebenen des Suchraums betrachten. Abschließend ist daher festzuhalten, dass es sich bei dem entwickelten Verfahren um einen schwach gekoppelten, nicht hierarchischen hybriden Algorithmus handelt, bei dem iterativ heterogene, spezialisierte Verfahren unterschiedliche Ebenen des Suchraums durchsuchen.

3.2.3 Implementierung

Der hybride evolutionäre Algorithmus wurde in C# umgesetzt und ist Teil des modular aufgebauten Softwaretools SmartOpt©, welches zur Produktprogrammoptimierung bei Preisbündelung eingesetzt wird.[168] Durch den modularen Aufbau können einzelne Komponenten ersetzt und bei Bedarf durch andere bereits vorhandene Anwendungen ausgetauscht werden. SmartOpt© besteht aus drei Komponenten, die nacheinander angewendet werden (vgl. Abbildung 3.5).

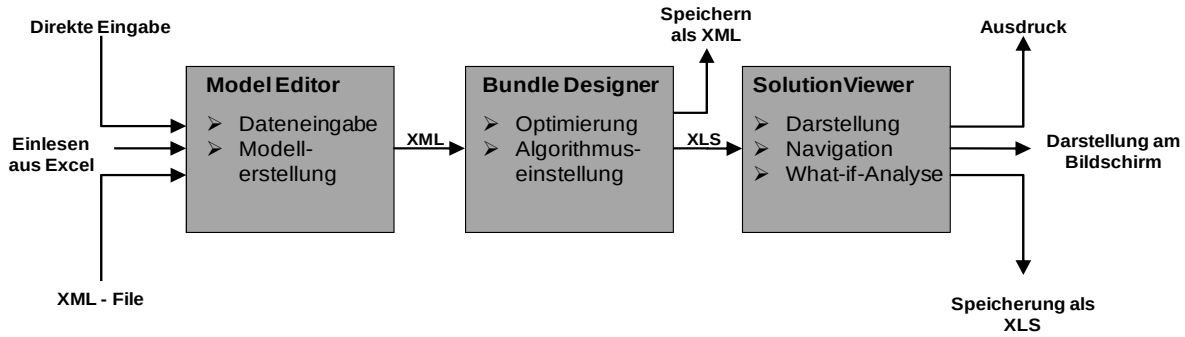

Abbildung 3.5: Aufbau des Softwaretools SmartOpt©

[167] Vgl. Krasnogor/Smith (2005), S. 483.
[168] Vgl. hierzu und Folgendem Werners et al. (2009), S. 70–76.

Zunächst sind alle Informationen im *Model Editor* zusammenzutragen. Dazu besteht die Möglichkeit, Daten per Hand einzugeben oder direkt aus den in vorherigen Schritten verwendeten Programmen zu importieren, ggf. über geeignete Austauschformate wie Excel- oder XML-Dateien. Seine Hauptaufgabe ist jedoch die Spezifikation von Settings, die im zweiten Schritt durch den *Bundle Designer* optimiert werden. Zur einfachen Handhabung werden alle Einstellungen in übersichtlichen Eingabemasken vorgenommen, die exemplarisch in Abbildung 3.6 dargestellt sind.

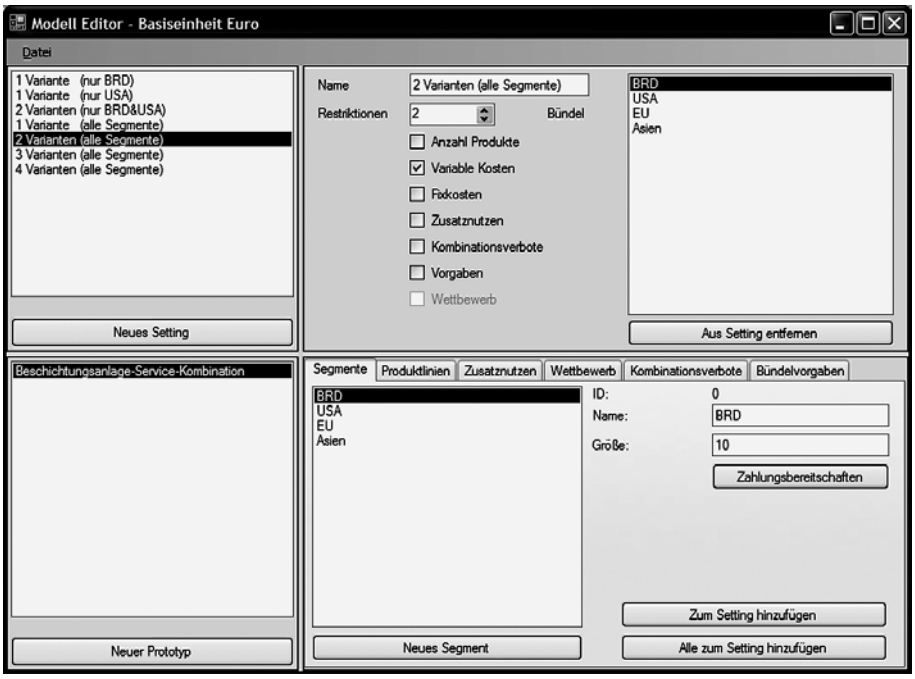

Abbildung 3.6: Eingabemaske des Model Editors

Die erstellten Settings werden anschließend an den *Bundle Designer* übergeben und durch diesen optimiert. Dazu kann dieser entweder direkt aus dem *Model Editor* aufgerufen werden oder er wird separat gestartet. Der *Bundle Designer* ist das Herzstück von SmartOpt© und optimiert mit Hilfe des entwickelten Algorithmus nacheinander alle formulierten Settings. Er wurde so konzipiert, dass Anwender Einblick in die Optimierung und ihren Fortschritt nehmen können. Der Benutzer hat jederzeit die Möglichkeit, die Optimierung anzuhalten und sich die bisher ermittelten Lösungen anzeigen zu lassen. Er kann so überprüfen, wie sich das Produktprogramm während eines Optimierungslaufes verändert und verbessert hat. Zusätzlich veranschaulichen mehrere Grafiken den Fortschritt der Optimierung. Die in Abbildung 3.7 dargestellte Grafik zeigt den Gesamtdeckungsbeitrag des besten gefundenen Produktprogramms in den einzelnen Optimierungsschritten. Es wird deutlich, dass der Algorithmus zu Beginn starke Verbesserungen erzielt und schließlich auch letzte Potenziale ausschöpft. Die erzielten Lösungen werden im *Bundle Designer* in der Tabelle zusammen mit dem erreichten Gesamtdeckungsbeitrag angezeigt. Zur besseren Visualisierung und Weiter-

verarbeitung können die optimalen Produktprogramme mit Angabe der Produktkonfigurationen und Preise nach Excel exportiert und vom *Solution Viewer* dargestellt werden. Mit ihm können die Auswirkungen postoptimaler Anpassung des Produktprogramms oder der Preise prognostiziert oder What-If-Analysen durchgeführt werden.

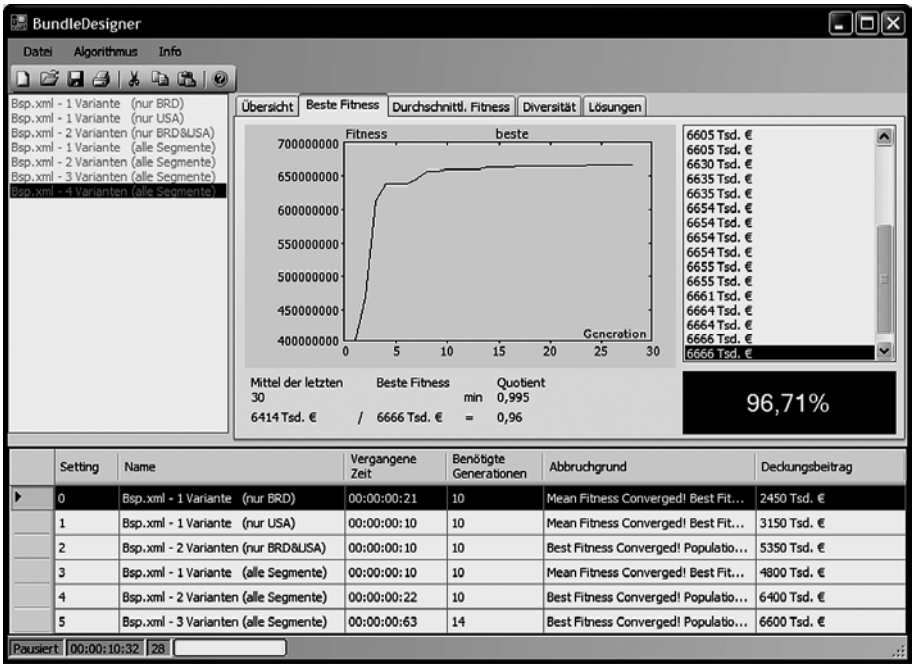

Abbildung 3.7: Darstellung des Bundle Designers

3.3 Heuristik I: Evolutionärer Algorithmus zur Designbestimmung

Bei der Konzipierung evolutionärer Algorithmen existieren einige Freiheitsgrade hinsichtlich der Ausgestaltung der einzelnen Elemente. Theoretische Untersuchungen im Hinblick auf die Vorteilhaftigkeit einzelner Ausgestaltungsmöglichkeiten können aufgrund ihres generischen Charakters nicht ohne weiteres auf alle Anwendungsprobleme übertragen werden, sodass bei der Entwicklung meist mehrere Möglichkeiten auszuprobieren sind, bevor ein zufriedenstellendes Ergebnis erzielt wird. Auf eine Vorstellung alternativer Ausgestaltungen wird hier aufgrund der vielfältigen Möglichkeiten verzichtet, vielmehr wird der nach mehrjähriger Entwicklung resultierende Algorithmus im Detail vorgestellt, dessen Erläuterung in der Reihenfolge des Ablaufs gemäß Abbildung 3.4 erfolgt.

3.3.1 Erzeugung der Startpopulation und Erstevaluation

Zu Beginn wird die Startpopulation zufällig erzeugt. Dazu wird ein neues Individuum generiert, bei dem zufällig für jede Variante jeder Produktlinie pro Merkmal eine Ausprägung festgelegt wird. Anschließend wird pro Bündel zufällig entschieden, ob ein bestimmter Produkttyp enthalten ist und ggf. welche Variante. Dieser Vorgang wird wiederholt, bis die vorgegebene *Populationsgröße* N_{ind} erreicht ist. Anschließend wird mit Hilfe der Preisheuristik für jedes Individuum das beste Preissystem bestimmt und der Gesamtdeckungsbeitrag berechnet. Durch dieses Vorgehen wird sichergestellt, dass alle Individuen zulässige Produktprogramme darstellen.

3.3.2 Fitnesszuweisung und Elternselektion

Der ermittelte Gesamtdeckungsbeitrag wird nicht direkt als Fitnessfunktion verwendet, sondern es erfolgt eine reihenfolgebasierte Fitnesszuweisung gemäß dem linearen Ranking.[169] Bei reihenfolgebasierten Fitnesszuweisungen hängt die Fitness eines Individuums von der Position (*Pos*) bzw. dem Rang des Individuums innerhalb der Population (N_{ind}) ab. Dazu werden die Individuen in eine aufsteigende Reihenfolge gebracht, sodass das Individuum mit dem geringsten Gesamtdeckungsbeitrag Position 1 einnimmt. Unter Angabe des gewünschten Selektionsdrucks *SP* berechnet sich die Fitness eines Individuums nach:[170]

$$\text{Fitness}(Pos) = 2 - SP + 2 \cdot (SP - 1) \cdot \frac{(Pos - 1)}{(N_{ind} - 1)} \qquad (3.1)$$

[169] Zu reihenfolgebasierten Fitnesszuweisungen im Allgemeinen und zum linearen Ranking im Speziellen vgl. Baker (1985); Baker (1987).
[170] Vgl. Pohlheim (2000), S. 19.

Der Vorteil reihenfolgebasierter gegenüber fitnessproportionaler Fitnesszuweisungen liegt in der besseren Kontrolle des Selektionsdrucks, durch welche die Gefahr vorzeitiger Konvergenz und Stagnation verringert werden kann.[171] Lineares Ranking erlaubt einen Selektionsdruck zwischen 1 und 2, wodurch, wie in Abbildung 3.8 ersichtlich, den Individuen unterschiedliche Fitnesswerte zugewiesen werden.

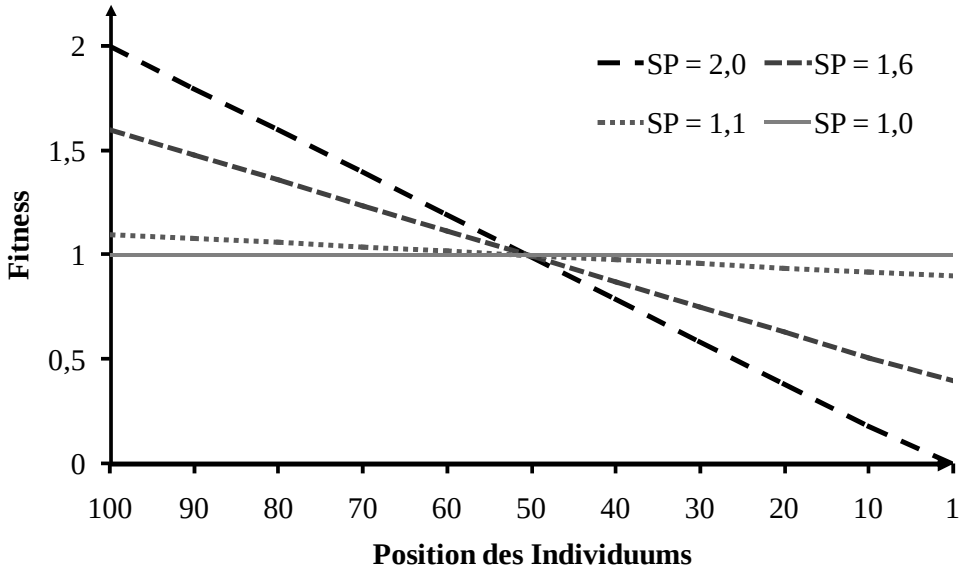

Abbildung 3.8: Fitness der Individuen in Abhängigkeit vom Selektionsdruck (Populationsgröße 100)
Quelle: In Anlehnung an Pohlheim (2000), S. 19.

Bei höchstem Selektionsdruck erhält das Individuum mit dem größten Gesamtdeckungsbeitrag eine Fitness von 2, die anderen linear abnehmende Fitnesswerte bis zum Individuum mit dem niedrigsten Gesamtdeckungsbeitrag, welches eine Fitness von 0 erhält. Mit abnehmendem Selektionsdruck reduziert sich der Abstand zwischen bestem und schlechtestem Fitnesswert bis bei einem Selektionsdruck von 1 alle Individuen Fitnesswerte von 1 und damit die gleiche Fortpflanzungswahrscheinlichkeit erhalten.

Basierend auf der Fitness werden die Individuen ausgewählt, die zur Zeugung von Nachkommen vorgesehen sind und damit Eltern werden. Anders als in Abbildung 3.4 angedeutet, werden jedoch nicht erst alle Eltern ausgewählt und anschließend Kinder erzeugt, sondern dies erfolgt iterativ in folgenden Schritten: Zunächst werden zwei Individuen zufällig ausgewählt. Dies geschieht durch die sogenannte Rouletteselektion, einem fitnessproportionalem Selektionsverfahren.[172] Die Rouletteselektion kann am einfachsten anhand eines Rouletterades veranschaulicht werden. Jeder Abschnitt des Rades entspricht einem Individuum, wobei die Abschnitte nicht gleich groß, son-

[171] Vgl. Blickle (1997); Pohlheim (2000), S. 17–18.
[172] Zur Rouletteselektion vgl. u.a. Davis (1991); Eiben/Smith (2003), S. 61f.

dern proportional zur Fitness der Individuen sind. Nach Drehen des Rades wird das Individuum ausgewählt, in dessen Abschnitt der Zeiger zeigt. Nachdem auf diese Weise zwei Eltern bestimmt wurden, zeugen diese durch Rekombination und Mutation genau ein Kind. Dieser Vorgang wird wiederholt bis die voreingestellte Anzahl an *Nachkommen K* erzeugt wurde.

3.3.3 Rekombination

Nachkommen werden durch Rekombination der Eltern-Chromosomen erzeugt, wobei darauf zu achten ist, dass entstehende Kinder zulässige Produktprogramme darstellen. Im Rahmen evolutionärer Algorithmen wird zur Rekombination häufig das Konzept des Crossovers verwendet.[173] Beim Single Point Crossover werden Allele bis zum Crossover-Punkt von der Mutter und die folgenden vom Vater übernommen. Das Konzept wurde weiterentwickelt zum Uniform Crossover[174], bei dem mit Wahrscheinlichkeit p ein Gen des Kindes das Allel der Mutter und mit Wahrscheinlichkeit $(1-p)$ das Allel des Vaters übernimmt, wodurch hier jedoch unzulässige Kinder entstehen können. Zur Vermeidung dieses Problems wird hier eine abgewandelte Form des Uniform Crossovers verwendet. Zunächst wird zufällig ein Elternindividuum als Mutter ausgewählt, die als Ausgangsquelle beginnt, ihre genetischen Informationen an das Kind zu übergeben. Bei jedem möglichen Crossover-Punkt besteht eine Wahrscheinlichkeit *MR*, dass dieser Crossover-Punkt aktiv ist, die Ausgangsquelle wechselt und nun die genetischen Informationen des Vater übergeben werden. Es kommt somit zu einem zufälligen Lauf zwischen beiden Eltern, aus dem sich das Kind ergibt. Die Stärke der Durchmischung beider Eltern ist abhängig von der Wahrscheinlichkeit *MR*. Beträgt diese 100%, wird an jedem möglichen Crossover-Punkt gewechselt und es findet eine sehr starke Vermischung der Eltern statt. Ist sie hingegen gering, so wird selten gewechselt und große zusammenhängende Bereiche der Elterngenome werden in das Kind übernommen. Beträgt sie gar 0%, so wird gar nicht gewechselt und die Informationen ausschließlich von der Mutter übernommen. Da diese Wahrscheinlichkeit die Durchmischungsstärke verändert, wird sie im Folgenden als *Mixing Rate* bezeichnet. Es kann gezeigt werden, dass in diesem Fall eine Mixing Rate von 50% einem Uniform Crossover mit einer Wahrscheinlichkeit von $p = 50\%$ entspricht.[175]

Um die Genome kombinieren zu können, ist zunächst aus den Elterngenomen jeweils ein String zu generieren, der das Bündeldesign darstellt. Dazu wird der Verweis in einem Bündel auf eine konkrete Variante in der Produktlinie durch das Produktdesign,

[173] Zum Crossover vgl. Pohlheim (2000), S. 39–41; Garus (2000), S. 259–268; Eiben/Smith (2003), S. 46–59.
[174] Vgl. Syswerda (1989).
[175] Vgl. de Jong/Spears (1992), S. 12.

d.h. die Merkmale und deren Ausprägungen, ersetzt. Jedes Gen dieses Strings stellt ein Produktmerkmal und der ganzzahlige Wert die gewählte Ausprägung dar. Zwischen zwei Genen existiert ein Crossover-Punkt, falls mit dem neuen Gen ein neues Bündel oder ein neues Produkt beginnt oder falls im aktuell betrachteten Bündel und Produkttyp sowohl beim Vater als auch der Mutter eine Variante vorhanden ist. Ist hingegen nur bei einem Elternteil eine Variante spezifiziert, kann nicht gewechselt werden, da sonst unvollständig definierte Produkte entstehen würden. Ob ein Produkt in ein Bündel des Kindes aufgenommen wird, hängt folglich davon ab, ob beim ersten Produktmerkmal in der Ausgangsquelle eine Ausprägung vorliegt.

Ein durch diese Rekombination entstandenes Bündeldesign kann jedoch unter Umständen unzulässig sein, da mehr Produktvarianten vorkommen können als in den einzelnen Linien erlaubt sind. Bei der Rücktransformation des Bündeldesign-Strings in das Kindgenom wird daher ggf. eine Reparatur vorgenommen. Bei der Rücktransformation wird das konkrete Bündeldesign wieder durch einen Verweis auf eine Variante innerhalb der Produktlinie ersetzt. Dazu wird, sofern möglich, eine entsprechend dem Bündeldesign-String gestaltete Variante in die Produktlinie aufgenommen. Ist dort kein Platz vorhanden, so kann in dem betreffenden Bündel keine Produktvariante mit diesen Spezifikationen enthalten sein. In dem Fall wird die ähnlichste bereits in der Linie vorhandende Variante gewählt, d.h. die Variante mit minimaler Distanz zur vorgegebenen Spezifikation. Die Distanz wird in Analogie zur Hamming-Distanz[176] bestimmt durch:

$$\text{Distanz}_{vv'} = \sum_{m=1}^{M} \sum_{a=1}^{A_m} \frac{\left| \delta_{vma} - \delta_{v'ma} \right|}{2} \qquad (3.2)$$

Besitzen zwei Varianten in einem Merkmal die gleiche Ausprägung, so ergibt sich ein Wert von 0, ansonsten von 1. Es ist die Variante zu wählen, bei der die Summe dieser Unterschiede, also deren Distanz am geringsten ist. Sind im Kindgenom in einer Linie weniger Varianten als möglich spezifiziert worden, so werden die restlichen Plätze zufällig durch Varianten der Eltern aufgefüllt, um die genetischen Informationen nicht zu verlieren.

Zum besseren Verständnis wird im Folgenden der entwickelte Rekombinationsoperator an einem Beispiel mit drei Bündeln, zwei Linien und zwei Varianten, die durch zwei Merkmale und drei Ausprägungen beschrieben werden, erläutert. In Abbildung 3.9 sind beispielhaft die Genome zweier Elternindividuen dargestellt.

[176] Vgl. u.a. Hamming (1950); Ernst (2008).

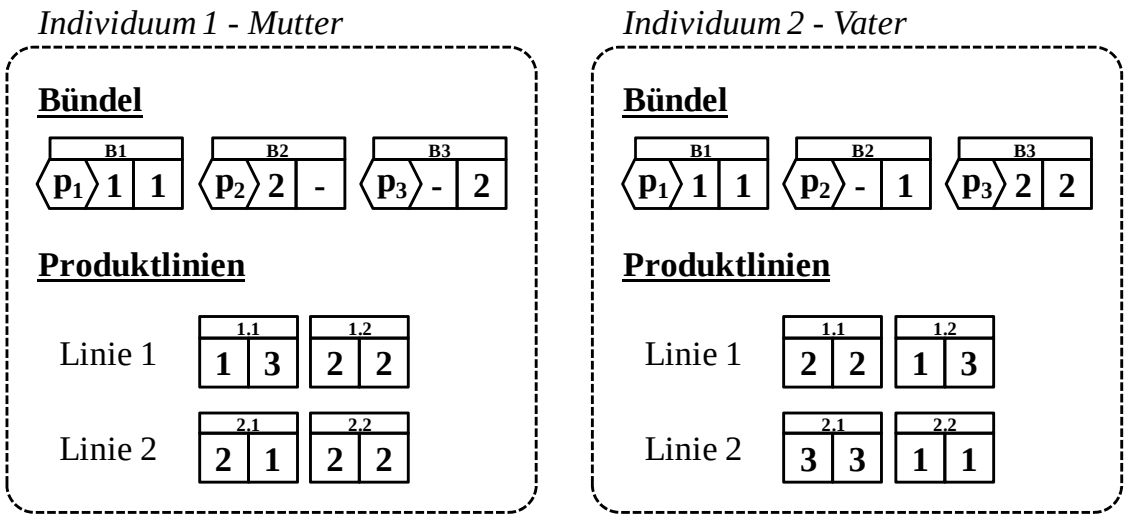

Abbildung 3.9: Beispielhafte Darstellung zweier Elternindividuen

Die sich ergebenden Bündeldesigns der Eltern können Abbildung 3.10 entnommen werden. Da Bündel 1 der Mutter aus Linie 1 und 2 jeweils Variante 1 enthält, ergibt sich aufgrund der Variantenspezifikationen der Teil-String 1,3,2,1. Für die restlichen Bündel erfolgt die Transformation analog.

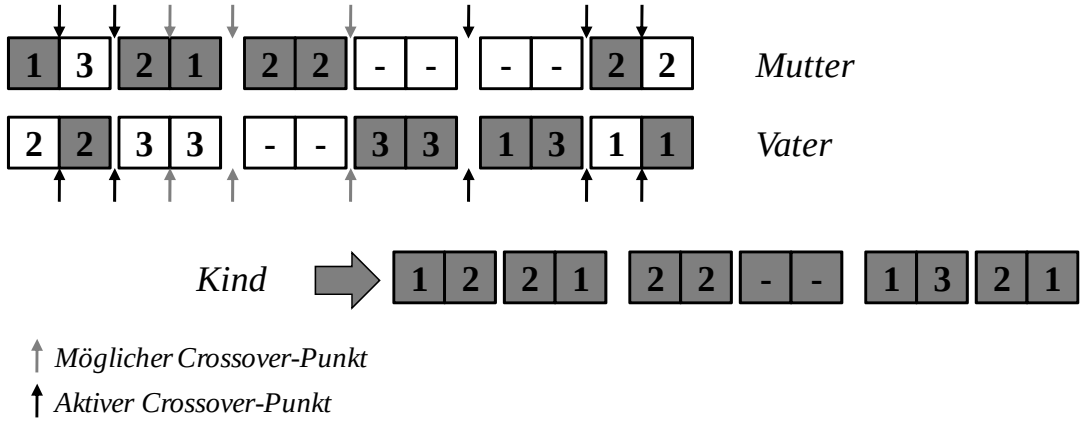

Abbildung 3.10: Bündeldesign des Kindes nach Rekombination der Elterngenome

Abbildung 3.10 veranschaulicht zudem alle möglichen Crossover-Punkte. Zwischen dem 5. und 6. Gen kann kein Crossover stattfinden, da Bündel 2 zwar im Muttergenom eine Variante aus Linie 1 besitzt, nicht jedoch im Vatergenom. Ob Bündel 2 des Kindes eine Variante aus Linie 1 enthält, ist damit davon abhängig, welcher der beiden Eltern als Ausgangsquelle zu Beginn des Produktes bzw. des 5. Gens im String ausgewählt wurde.

Angenommen, die schwarz markierten Crossover-Punkte sind aktive Punkte, so resultiert der dargestellte String. Für die Rücktransformation ist zu prüfen, ob alle Restriktionen eingehalten sind. Aus Abbildung 3.11 ist ersichtlich, dass im Kindergenom in jedem Bündel eine andere Variante aus Produktlinie 1 enthalten ist. Demnach besitzt das Kind drei Varianten aus Linie 1, was annahmegemäß nicht erlaubt ist. Die dritte

Variante kann daher nicht aufgenommen werden und wird stattdessen durch die ähnlichste Variante ersetzt. Dies ist Variante 1, da die Distanz zur Variante 1 mit 1 kleiner als zur Variante 2 mit einer Distanz von 2 ist. Auf der anderen Seite wird nur eine identisch konfigurierte Variante aus Linie 2 in den Bündeln verwendet. Da somit ein Platz in der Linie frei ist, wird dieser zufällig besetzt, z.B. mit Variante 1 des Vaters.

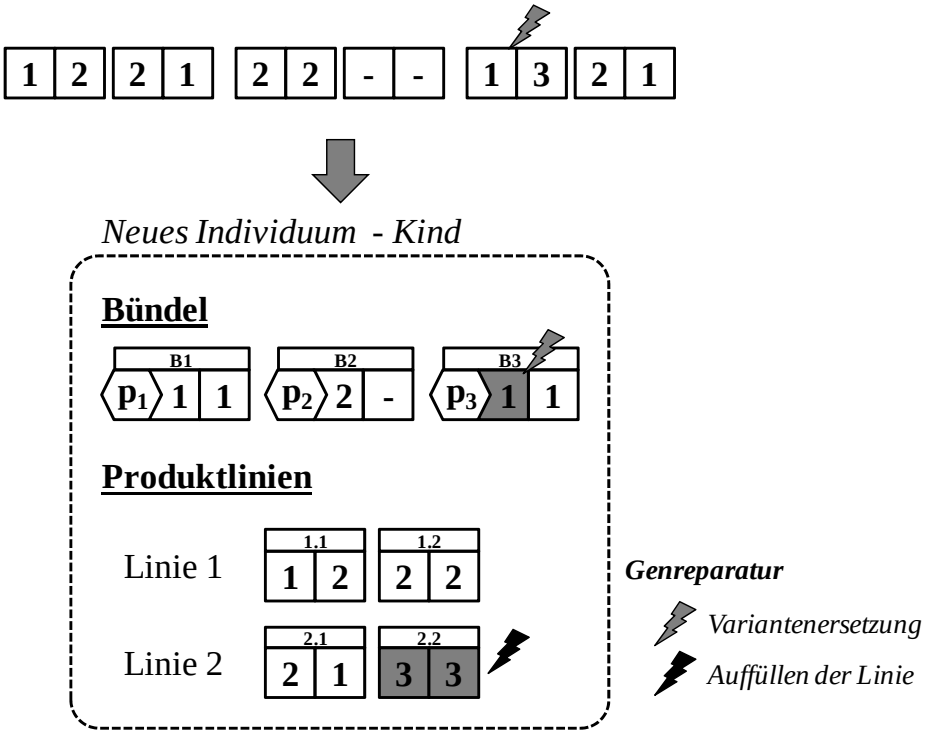

Abbildung 3.11: Genreparatur zur Vermeidung unzulässiger Nachkommen

3.3.4 Mutation

Die genetische Vielfalt wird durch drei Mutationsoperatoren auf den verschiedenen semantischen Ebenen (Bündel, Produkte, Merkmale) sichergestellt. Da die Operatoren unterschiedlich stark das Genom verändern, kann die Wahrscheinlichkeit ihres Auftretens, welche als *Mutationsrate* bezeichnet wird, unabhängig voneinander eingestellt werden. Die Mutationsraten geben jeweils an, mit welcher Wahrscheinlichkeit ein Element der betrachteten Ebene einer Mutation unterliegt.

Auf der Merkmalsebene wird zufällig für ein Merkmal eine andere Ausprägung gewählt, weswegen der Operator als *Change-Product-Specification* bezeichnet wird. Die Festlegung einer anderen Ausprägung führt zu einer vergleichsweise geringen Änderung am Genom, die sich u.U. jedoch auf sehr viele Bündel auswirkt, je nachdem, in wie vielen Bündeln das entsprechende Produkt enthalten ist.

Auf Produktebene wird durch den Operator *Replace-Product-In-Line* eine komplette Variante zufällig neu spezifiziert. Dieser Operator führt zu großen Veränderungen am

Genom. Zum einen ändert er eine komplette Produktspezifikation, zum anderen hat diese Änderung Einfluss u.U. auf sehr viele Bündel. Die Wahrscheinlichkeit, dass eine Produktvariante dieser Mutation unterliegt, sollte daher sehr gering sein.

Auf der Bündelebene wird in einem Bündel eine Variante zufällig durch eine andere Variante derselben Linie ersetzt oder komplett aus dem Bündel entfernt. Der Operator wird daher als *Replace-Product-In-Bundle* bezeichnet. Er ist der einzige Operator, der einerseits auf jeden Fall Auswirkungen auf Bündel hat, andererseits hingegen auch nur ein Bündel verändert. Die Mutationsrate gibt an, mit welcher Wahrscheinlichkeit ein Produkt innerhalb eines Bündels ersetzt wird. Abschließend gibt Tabelle 3.1 einen Überblick über die drei Mutationsoperatoren.

Tabelle 3.1: Überblick über Mutationsoperatoren

Mutationsoperator	Betrachtete Elemente	Mutations-rate	Auswirkungen auf das Genom
Change-Product-Specification	Merkmale	0,02	Ein Merkmal, u.U. mehrere Bündel
Replace-Product-In-Line	Varianten in Linien	0,005	Mehrere Merkmale, u.U. mehrere Bündel
Replace-Product-In-Bundle	Produkte in Bündeln	0,05	Mehrere Merkmale, 1 Bündel

3.3.5 Überlebensselektion und Wiedereinfügen

Im Rahmen der Überlebensselektion ist festzulegen, welche Individuen in die Population der nächsten Generation aufgenommen werden. Dazu ist zunächst zu klären, welche Individuen dafür überhaupt in Betracht kommen. Mögliche Kandidaten sind alle Kinder und die mit Hinblick auf den Gesamtdeckungsbeitrag E-besten Eltern, die sogenannten *Elitisten*.[177] Je nach Einstellung von E werden unterschiedlich viele Individuen betrachtet. Ist E = 0, so können nur Kinder in die neue Population übernommen werden und alle Eltern sterben. Ist E = 1, so existiert ein Elitist, der neben den Kindern, die Chance zu überleben hat. Ist E = Populationsgröße, so sind alle Eltern gleichzeitig Elitisten und Kinder- und Elternpopulation werden gemeinsam berücksichtigt. Die Existenz eines Elitisten besitzt den Vorteil, dass das beste Individuum in der neuen Population niemals eine geringere Fitness aufweist als das beste Individuum der vorhergehenden Population. Aus diesem Grund wird ein Elitist, also E = 1 empfohlen.

[177] Zu Elitismus vgl. u.a. Dréo et al. (2006), S. 91.

Ausgehend von den berücksichtigten Individuen erfolgt eine *Truncation*-Selektion auf Basis des Gesamtdeckungsbeitrags, bei der die Populationsgröße-besten Individuen ausgewählt und in die nächste Population übernommen werden.[178]

3.3.6 Abbruchkriterien

Abbruchkriterien sollten so definiert sein, dass sie einerseits die Optimierung erst beenden, wenn ein ausreichend gutes Ergebnis erreicht wurde und andererseits nicht mehr Berechnungen durchführen lassen als zur Erreichung des ersten Ziels erforderlich sind.[179] Zudem sollte mindestens ein Abbruchkriterium verwendet werden, welches einen Abbruch garantieren kann, was hier durch Vorgabe einer *maximalen Anzahl von Generationen* erreicht wird. Um auf der anderen Seite eine Mindestlaufzeit sicherzustellen, kann zudem eine *minimale Anzahl von Generationen* vorgegeben werden. Sinnvollerweise wird das garantierte Abbruchkriterium durch weitere Kriterien ergänzt, die prüfen, ob ein Fortsetzen der Optimierung sinnvoll ist, um unnötig lange Laufzeiten zu vermeiden. Dies geschieht meist durch Konvergenzprüfung,[180] wofür im vorgeschlagenen Algorithmus zwei unterschiedliche Arten von Abbruchkriterien eingesetzt werden, nämlich verbesserungsbezogene und diversitätsbezogene Kriterien.

Die verbesserungsbezogenen Abbruchkriterien untersuchen, ob in den letzten Generationen Verbesserungen erzielt wurden. Haben lange keine Verbesserungen stattgefunden, so ist dies ein Indiz dafür, dass durch den Algorithmus auch im Weiteren keine Verbesserungen mehr erzielt werden können. Es werden *laufende Mittelwerte* (*running mean*) der besten (GDB^{best}) und der durchschnittlichen Zielfunktionswerte ($GDB^{\varnothing}$) berechnet, wozu die Anzahl an Generationen ($RunMeanGen^{best}$ bzw. $RunMeanGen^{\varnothing}$) festzulegen sind, die zur Bildung der Mittelwerte herangezogen werden.[181] Der relative Running Mean für die aktuelle Generation *Gen* berechnet sich durch:

$$\text{runmean}_{\text{Gen}}^{\text{best}} = \frac{\frac{1}{\text{RunMeanGen}^{\text{best}}} \sum_{i=\text{Gen}-\text{RunMeanGen}^{\text{best}}}^{\text{Gen}-1} \text{GDB}_i^{\text{best}}}{\text{GDB}_{\text{Gen}}^{\text{best}}} \tag{3.3}$$

$$\text{runmean}_{\text{Gen}}^{\varnothing} = \frac{\frac{1}{\text{RunMeanGen}^{\varnothing}} \sum_{i=\text{Gen}-\text{RunMeanGen}^{\varnothing}}^{\text{Gen}-1} \text{GDB}_i^{\varnothing}}{\text{GDB}_{\text{Gen}}^{\varnothing}} \tag{3.4}$$

Die laufenden Mittelwerte sind definiert von 0 bis 1. Hat innerhalb der betrachteten Generationen keine oder nur eine sehr geringe Verbesserung stattgefunden, so ent-

[178] Vgl. Dréo et al. (2006), S 90; Pohlheim (2000), S. 27f.
[179] Vgl. Pohlheim (2000), S. 63.
[180] Vgl. de Jong (2006), S. 78f.
[181] Vgl. Balakrishnan/Jacob (1996), S. 1111.

spricht der Durchschnitt der letzten Generationen in etwa dem Wert der aktuellen Generation, womit der laufende Mittelwert gleich 1 bzw. nahe 1 ist. Für beide laufenden Mittelwerte kann eine Schwelle vorgegeben werden, bei deren Überschreitung das Abbruchkriterium als erfüllt gilt.

Die diversitätsbezogenen Abbruchkriterien bewerten das genetische Potenzial der aktuellen Population im Hinblick auf zukünftige Verbesserungen. Sind sich die Individuen sehr ähnlich, so ist die Population konvergiert und durch Rekombination dürften keine Verbesserung mehr erreicht werden. Wird die Evolution trotzdem fortgeführt, so können Verbesserungen ausschließlich durch Mutation erreicht werden, wodurch der evolutionäre Algorithmus degeneriert und einer zufälligen Suche gleicht. Eine Möglichkeit, die Konvergenz festzustellen, ist die Bildung des Quotienten aus bestem und durchschnittlichem Zielfunktionswert der aktuellen Generation gemäß:

$$\text{mean / best}_{\text{Gen}} = \frac{\text{GDB}_{\text{Gen}}^{\varnothing}}{\text{GDB}_{\text{Gen}}^{\text{best}}} \tag{3.5}$$

Im Laufe der Generationen werden sich die Individuen tendenziell ähnlicher, wodurch der Quotient gegen 1 konvergiert.[182]

Eine weitere Möglichkeit der Konvergenzprüfung ist die Berechnung der Anzahl verschiedener Zielfunktionswerte innerhalb einer Population. Sie stellt eine Approximation für die Anzahl genetisch unterschiedlicher Individuen dar, deren Bestimmung weitaus aufwändiger wäre.[183] Unter der Annahme, dass Individuen mit identischen Fitnesswerten zumindest im Hinblick auf die relevanten genetischen Informationen identisch sind, kann die Populationsdiversität bestimmt werden durch:

$$\text{Diversity}_{\text{Gen}} = \frac{\text{Anzahl verschiedener GDB}_{\text{Gen}}}{\text{N}_{\text{ind}}} \tag{3.6}$$

Da die Produktprogrammoptimierung nicht zeitkritisch ist, sind insbesondere vorzeitige Abbrüche zu vermeiden. Aus diesem Grund werden verbesserungs- und diversitätsbezogene Abbruchkriterien gemeinsam eingesetzt. Der Algorithmus terminiert, sofern aus beiden Kategorien mindestens ein Kriterium erfüllt ist, also wenn in den letzten Generationen kaum Verbesserungen stattgefunden haben und die Diversität der aktuellen Population so gering ist, dass zukünftige Verbesserungen unwahrscheinlich sind. Abschließend veranschaulicht Abbildung 3.12 die Prüfung der Abbruchkriterien.

[182] Zur Problematik der Verwendung von Mittelwerten vgl. Pohlheim (2000), S. 65–67.
[183] Für weitere Möglichkeiten der Diversitätsbestimmung vgl. Weicker (2007), S. 62–64.

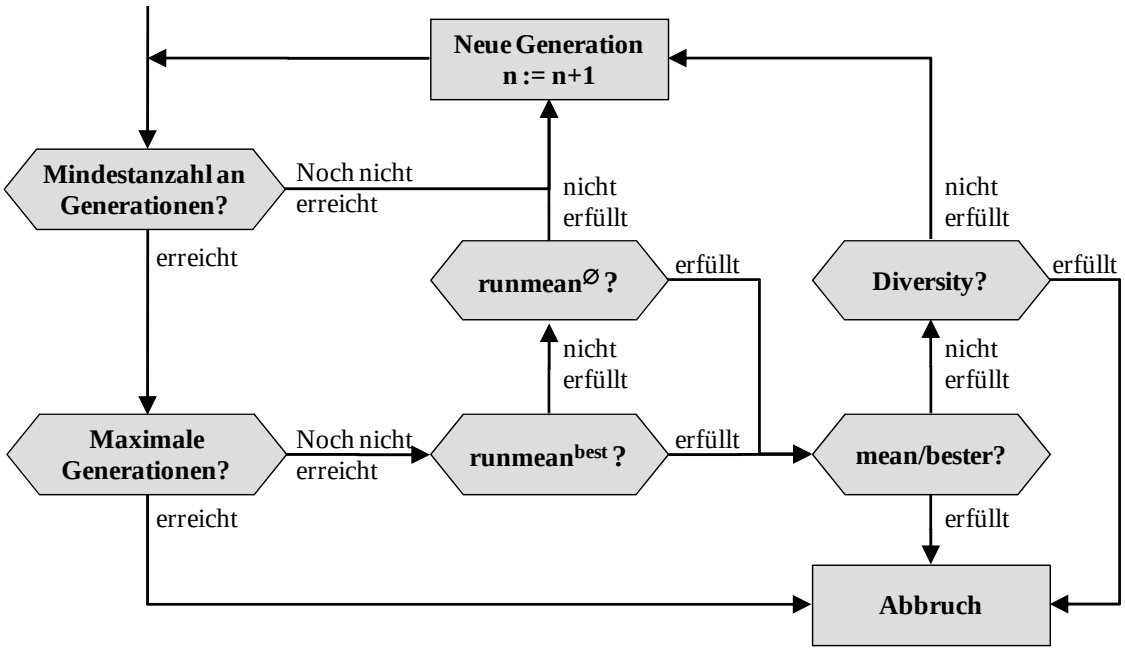

Abbildung 3.12: Ablauf der Prüfung der Abbruchkriterien

3.3.7 Überblick über Einstellungsparameter

Der vorgeschlagene Algorithmus kann durch diverse Parameter der einzelnen Elemente und Operatoren eingestellt und angepasst werden. Tabelle 3.2 stellt alle bisher beschriebenen Parameter noch einmal zusammenfassend dar.

Tabelle 3.2: Überblick über Parameter des evolutionären Algorithmus

Bereich	Bezeichnung	Kurzbezeichnung
Allgemein	Populationsgröße	N_{ind}
	Anzahl Nachkommen	K
Fitnesszuweisung & Elternselektion	Selektionsdruck	SP
Rekombination	Mixing Rate	MR
Mutation	Mutationsrate Replace-Product-In-Bundle	MR_{bun}
	Mutationsrate Replace-Product-In-Line	MR_{line}
	Mutationsrate Change-Product-Specification	MR_{spec}
Überlebensselektion & Wiedereinfügen	Anzahl Elitisten	E
Abbruchkriterien	Minimale Anzahl von Generationen	MinGen
	Maximale Anzahl von Genrationen	MaxGen
	Anzahl Generationen für laufenden Mittelwert des Besten	$RunMean^{best}$
	Anzahl Generationen für laufenden Mittelwert des Durchschnitts	$RunMean^{\varnothing}$
	Abbruchschwelle laufender Mittelwert des Besten	TH_{best}
	Abbruchschwelle laufender Mittelwert des Durchschnitts	TH_{mean}
	Abbruchschwelle Quotient Durchschnitt/Bester	$TH_{mean/best}$
	Abbruchschwelle Diversität	Th_{div}

3.4 Heuristik II: Heuristiken zur Preisfindung bei gegebenem Design

3.4.1 Das Pricing Subproblem

Sind Anzahl und Konfigurationen der anzubietenden Bündel festgelegt, reduziert sich das Gesamtproblem und es ist das zum Bündeldesign deckungsbeitragsmaximale Preissystem zu bestimmen.[184] Dieses Problem ähnelt dem Problem der *Product Line Selection*, bei dem jedoch Produkte mit ihren gegebenen Preisen auszuwählen sind. Zur Bestimmung des optimalen Preissystems kann ein Optimierungsmodell eingesetzt werden, wozu folgende Indizes, Parameter und Variablen benötigt werden:

Indizes und Indexmengen

$b = 0, \ldots, B$	Bündel
$k = 1, \ldots, K$	Kundensegmente

Parameter

R_{kb}	Reservationspreis / Zahlungsbereitschaft von Kunde k für Bündel b mit $R_{k0} = 0$ für alle Kunden k
SG_k	Segmentgröße von Kundensegment k
VK_b	Variable Kosten von Bündel b

Variablen

p_k Preis, den Kunde k für das von ihm erworbene Bündel bezahlt

$$\theta_{kb} = \begin{cases} 1, & \text{falls Kunde k Bündel b erwirbt} \\ 0, & \text{sonst} \end{cases}$$

Das gemischt-ganzzahlige lineare Modell *Pricing i.w.S.* lautet wie folgt:

$$\max \sum_{k=1}^{K} SG_k \cdot \left(p_k - \sum_{b=0}^{B} VK_b \cdot \theta_{kb} \right) \tag{3.7}$$

$$\text{s.d.} \quad \sum_{b=0}^{B} R_{kb} \cdot \theta_{kb} - p_k \geq \sum_{b=0}^{B} R_{kb} \cdot \theta_{k'b} - p_{k'} \qquad \forall k, k' \tag{3.8}$$

$$\sum_{b=0}^{B} R_{kb} \cdot \theta_{kb} - p_k \geq 0 \qquad \forall k \tag{3.9}$$

$$\sum_{b=0}^{B} \theta_{kb} = 1 \qquad \forall k \tag{3.10}$$

[184] Es sei angenommen, dass die Angebotsentscheidung für alle Bündel fixiert ist und durch die Preissetzung nicht verändert wird. Dann sind eventuelle Fixkosten der Bündel entscheidungsirrelevant und können vernachlässigt werden.

$$\{\theta_{k0}, \theta_{k1}, ..., \theta_{kB}\} \text{ is SOS1} \qquad \forall k \qquad (3.11)$$

$$p_k \geq 0 \qquad \forall k \qquad (3.12)$$

Die Zielfunktion (3.7) maximiert den Gesamtdeckungsbeitrag, der sich aus den Deckungsbeiträgen der von den Kunden gekauften Bündel multipliziert mit der Segmentgröße zusammensetzt. Restriktionen (3.8) und (3.9) bilden die Kundenwahl gemäß der First-Choice Regel ab, während Restriktion (3.10) den Kunden zwingt, ein Bündel oder das 0-Bündel B_0 zu erwerben und somit auf einen Kauf zu verzichten. Die Restriktionen (3.11) und (3.12) beinhalten die Variablendefinitionen. Bei den Entscheidungsvariablen θ werden k Mengen vom Typ Special Ordered Sets 1 (SOS1) definiert. Bei Special Ordered Sets vom Typ 1 kann aus einer Menge von Variablen maximal eine Variable einen von Null verschiedenen Wert annehmen.[185] Alternativ ist eine Deklaration als Binärvariablen möglich.[186].

Das Modell lässt sich auch für größere Instanzen zügig optimal lösen. Die Berechnungszeit beträgt meist ca. 1 Sekunde, was allerdings für eine Hybridisierung mit evolutionären Algorithmen zu langsam ist. Besteht beispielsweise die Population aus 300 Individuen und beträgt die Zeit zur Lösung des Pricing-Subproblems im Durchschnitt 1 Sekunde pro Individuum, so benötigt die Berechnung bereits 300 Sekunden bzw. 5 Minuten. Eine neue Generation kann daher frühestens nach 5 Minuten erreicht werden, wodurch sich eine extrem lange Laufzeit ergibt. Zur Reduktion der Berechnungsdauer werden heuristische Lösungsverfahren untersucht und im Weiteren vorgestellt.

Zur heuristischen Lösung wird das *Pricing-Problem i.w.S.* in zwei Teilprobleme zerlegt, und zwar in ein *Assignment-Problem*, bei dem die Kunden den Bündeln zugeordnet werden und ein *Pricing-Problem i.e.S.*, bei dem das deckungsbeitragsmaximale Preissystem zu bestimmen ist. Diese Bestimmung erfolgt unter der Nebenbedingung, dass die Kunden gemäß der First-Choice Regel die im Rahmen des Assignments zugeordneten Bündel erwerben. Wird das Kunden k zugeordnete Bündel mit b(k) bezeichnet und bezeichne p_b den Preis von Bündel b, so ergeben sich folgende Änderungen der Indizes und Variablen.

Zusätzliche Indizes und Indexmengen

b(k)	Bündel b, dem Kunde k zugeordnet ist

Variablenänderung

p_b	Preis von Bündel b

[185] Vgl. Williams (1999), S. 165f.; Zimmermann (2008), S. 263. Für einen Literaturüberblick zur Entwicklung Special Ordered Sets vgl. Forrest/Tomlin (2007).

[186] Es zeigte sich jedoch, dass XPress-Ive das Modell bei Verwendung des SOS-Ansatzes schneller löst.

Das lineare Modell *Pricing i.e.S.* lautet dann:[187]

$$\max \sum_{k=1}^{K} SG_k \cdot \left(p_{b(k)} - VK_{b(k)} \right) \tag{3.13}$$

$$\text{s.d.} \quad R_{kb(k)} - p_{b(k)} \geq R_{kb} - p_b \qquad \forall b,k \tag{3.14}$$

$$p_0 = 0; \qquad p_b \geq 0 \qquad \forall b > 0 \tag{3.15}$$

Die Zielfunktion maximiert den Gesamtdeckungsbeitrag von allen gekauften Bündeln. Restriktion (3.14) stellt sicher, dass das einem Kunden zugeordnete Bündel b(k) für ihn einen größeren Surplus besitzt als jedes andere Bündel inklusive dem 0-Bündel. (3.15) gibt die Nichtnegativitätsbedingung und verbietet einen positiven Preis für das 0-Bündel. Da nicht zugeordnete Bündel in der Zielfunktion nicht erfasst werden, kann für diese ein nahezu beliebiger Preis gesetzt werden. Er muss nur ausreichend hoch sein, damit kein Kunde dieses Bündel erwirbt und so von seiner Zuordnung abweicht. Alternativ können diese Bündel im Vorfeld aus der Betrachtung ausgeklammert werden.[188]

Sind einem Bündel mehrere Kunden zugeordnet, wird genau ein Kunde Restriktion (3.14) in Bezug zu einem anderen Bündel mit Gleichheit erfüllen, während für alle anderen Kunden die Restriktion nicht bindend ist. Die Preisdifferenz entspricht demnach der minimalen Differenz der Reservationspreise, sodass das Modell um die nicht bindenden Restriktionen reduziert werden kann. Dazu bezeichne $G_b = \{k \mid \theta_{kb} = 1\}$ die Menge aller Bündel b zugeordneten Kunden k und $B' = \{b \mid G_b \neq \varnothing\}$ die Menge aller gekauften Bündel sowie $N_b = |G_b|$, die Anzahl Kunden, die Bündel b erwerben. Damit kann (3.13) – (3.15) umformuliert werden zu:[189]

$$\max \sum_{b=1}^{B} N_b \cdot \left(p_b - VK_b \right) \tag{3.16}$$

$$\text{s.d.} \quad p_i - p_b \leq \min_{k \in G_b} \left(R_{ki} - R_{kb} \right) \equiv a_{bi} \qquad \forall i \neq b, \forall i \in B', \forall b \tag{3.17}$$

$$p_0 = 0; \qquad p_b \geq 0 \qquad \forall b > 0 \tag{3.18}$$

Wird die im Optimum erfüllte Nichtnegativitätsbedingung (3.18) relaxiert und (3.16) sowie (3.17) unter Hinzufügen von Dualvariablen f_{bi} dualisiert[190], so erhält man

[187] Vgl. Dobson/Kalish (1988), S. 112; Dobson/Kalish (1993), S. 173f.; Stauß (2006), S. 76–78; Shioda et al. (2009), S. 7–9. Ebenso für die weiteren Umformungen.
[188] Vgl. Shioda et al. (2009), S. 7; Dobson/Kalish (1993), S. 174.
[189] Vgl. Dobson/Kalish (1988), S. 112.
[190] Vgl. zur Dualisierung linearer Modelle u.a. Werners (2008), S. 102–109; Neumann/Morlock (2002), S. 76–85; Kistner (2003), S. 40–52.

$$\min \quad \sum_{b=0}^{B} \sum_{i \in B'} a_{bi} f_{bi} \tag{3.19}$$

$$\text{s.d.} \quad \sum_{i \neq b} f_{bi} - \sum_{i \neq b} f_{ib} = N_b \qquad \forall i \in B', \forall b \tag{3.20}$$

$$\sum_{i \in B'} f_{bi} = 0 \qquad \forall b \in \{0,...,B\} \setminus B' \tag{3.21}$$

$$f_{bi} \geq 0 \qquad \forall b, \forall i \in B' \tag{3.22}$$

Da durch Restriktion (3.21) zusammen mit (3.22) alle in Restriktion (3.21) enthaltenen f_{bi} gleich 0 sind, können diese aus der Modellierung entfernt und die Restriktion (3.21) weggelassen werden.

3.4.2 Heuristiken zur Lösung des Pricing-Problems

Dieses Problem kann graphentheoretisch interpretiert und wie in Abbildung 3.13 veranschaulicht werden.[191] Es handelt sich um ein kostenminimales Fluss- bzw. im Englischen Minimum Cost Flow Problem.[192] Der resultierende gerichtete Graph besteht aus einem Knoten für jedes angebotene, d.h. mindestens einem Kunden zugeordnete Bündel sowie dem 0-Bündel und einem Pfeil von Knoten i zum Knoten j mit der Länge $a_{ij.}$[193]. Da keine Kapazitätsbeschränkungen für die Kostenflüsse existieren, ist optimalerweise jeder Knoten von der nächsten Quelle zu beliefern.

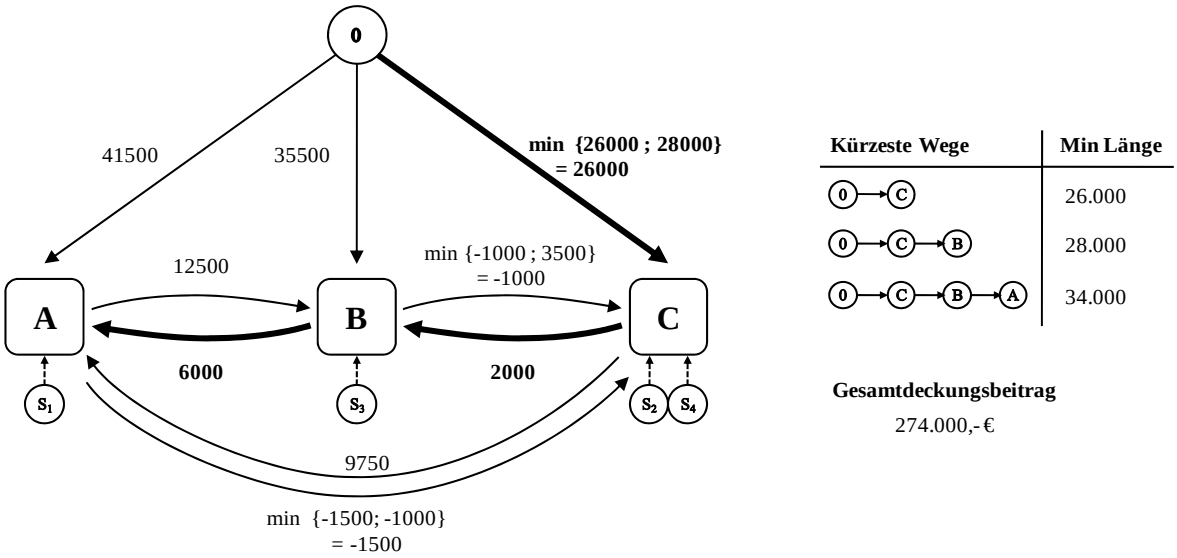

Abbildung 3.13: Graphentheoretisches Äquivalent für ein Produktprogramm mit 3 Bündeln

[191] Für eine Einführung in die Graphentheorie vgl. etwa Krumke/Noltemeier (2005).
[192] Zum Minimum Cost Flow Problem vgl. u.a. Neumann/Morlock (2002), S. 275–282; Winston (2004), S. 450f.; Hillier/Lieberman (2005), S. 396–401.
[193] Mit $j \neq 0$, d.h. keine Pfeil von einem Bündel zum 0-Bündel.

Das Problem reduziert sich damit zu mehreren kürzesten Wege-Problemen vom 0-Bündel als einziger Quelle[194] zu jedem angebotenen Bündel.[195] Die Kantenbewertung a_{ij} zwischen zwei Knoten i und j stellt wegen (3.17) die minimale Reservationspreisdifferenz zwischen Bündel i und j für alle Bündel j zugeordnete Kunden dar. Diese Differenz ist positiv, sofern alle Bündel j zugeordneten Kunden einen größeren Reservationspreis für Bündel j als für i besitzen. Der gesamte Graph besitzt genau dann ausschließlich positive Kantenbewertungen, wenn alle Kunden dem angebotenen Bündel zugeordnet wurden, für welches ihr Reservationspreis maximal ist. Es sei hier nochmal erwähnt, dass bei der Maximumsbestimmung Bündel ignoriert werden, denen keine Kunden zugeordnet wurden. Liegt eine Lösung für (3.19) – (3.22) vor, so können daraus direkt die optimalen Bündelpreise bestimmt werden. Da der Preisunterschied zwischen zwei Bündeln niemals die minimale Reservationspreisdifferenz übersteigen darf, gilt im Optimum $p_j = \min_i (p_i + a_{ij})$. Dieses kann rekursiv gelöst werden und mit $p_0 = 0$ entspricht der optimale Bündelpreis p_b genau der ermittelten Länge des kürzesten Weges von der Quelle[196] zu dem entsprechenden Knoten b. Existiert im Graph hingegen mindestens ein negativer Zyklus, so kann kein kürzester Weg bestimmt werden, da dieser beliebig reduziert werden kann. Gleichzeitig bedeutet dies, dass kein Preisschema existiert, welches zu der vorgegebenen Zuordnung von Kunden zu Bündeln führt. Die Zuordnung ist damit unzulässig.

Kürzeste Wege Probleme können sehr effizient gelöst werden. Sind alle Kantenbewertungen positiv, so kann u.a. der Algorithmus von Dijkstra eingesetzt werden.[197] Wie oben erläutert, kann jedoch nicht unbedingt von ausschließlich positiven Kantenbewertungen ausgegangen werden. In diesem Fall kann u.a. der Bellman-Ford-Algorithmus verwendet werden, der zugleich in der Lage ist, negative Zyklen und damit unzulässige Zuordnungen zu erkennen.[198] Da mit diesen Verfahren das zu einer gegebenen zulässigen Zuordnung optimale Preissystem schnell bestimmbar ist, bleibt die Frage, wie eine optimale Zuordnung gefunden werden kann. Dazu wurden unter anderem von Dobson und Kalish mehrere meist zweistufige Heuristiken vorgeschlagen und in umfangreichen Untersuchungen miteinander verglichen.[199] In Stufe I wird mit einer Initialisierungsheuristik eine gute, auf jeden Fall jedoch zulässige Startlösung erzeugt, die

[194] Bzw. mehreren Quellen, falls weitere Bündel am Markt existieren, deren Preise jedoch Parameter sind. Vgl. Dobson/Kalish (1993), S. 174.

[195] Vgl. zum Finden kürzester Wege in Graphen u.a. Neumann/Morlock (2002), S. 203–226; Krumke/Noltemeier (2005), S. 167–192.

[196] Bzw. der geringsten Länge aller kürzesten Wege von irgendeiner Quelle.

[197] Vgl. zum Algorithmus von Dijkstra Dijkstra (1959). Für eine modernere Darstellung u.a. Werners (2008), S. 197–205; Krumke/Noltemeier (2005), S. 175–181.

[198] Der Bellman-Ford-Algorithmus geht auf die Arbeiten von Bellman und Ford zurück. Vgl. Bellman (1958); Ford (1956). Für eine modernere Darstellung u.a. Krumke/Noltemeier (2005), S. 181–186.

[199] Vgl. Dobson/Kalish (1988); Dobson/Kalish (1993).

ggf. in Stufe zwei durch eine weitere Heuristik verbessert wird. Im Rahmen dieser Arbeit sind mit PH II und PH III die zwei besten Verfahren von Dobson und Kalish sowie mit PH I eine vorgeschlagene Abwandlung umgesetzt worden. Einen Überblick über den Aufbau und Zusammensetzung der implementierten Preisheuristiken gibt Tabelle 3.3.

Tabelle 3.3: Aufbau der implementierten Preisheuristiken

Stage	Preisheuristik		
	PH I	PH II	PH III
Stage I Initialisierungsheuristik	MaxR	MaxW	DK PriceGreedy
Stage II Verbesserungsheuristik	DK Reassignment	DK Reassignment	-

Bei den Initialisierungsheuristiken *MaxR*[200] und *MaxW*[201] ordnet die *MaxR*-Heuristik jedes Kundensegment dem Bündel zu, für das es den größten Reservationspreis besitzt, sodass gilt $b(k) = \max_b \{R_{kb}\}$. Ist dies bei einem Segment für mehrere Bündel gegeben, wird zufällig eines ausgewählt. Die Heuristik könnte dahingehend angepasst werden, dass in diesen Fällen das Kriterium der maximalen Wohlfahrt herangezogen wird oder dass das Segment dem Bündel zugeordnet wird, welches bisher noch keine Kundenzuordnung besitzt. Da in der Initialisierung nicht zugeordnete Bündel nur zu prohibitiv hohen Preisen angeboten werden, führt letztere Modifikation im weiteren Verlauf zu einer größeren Anzahl berücksichtigter Bündel. Die *MaxW*-Heuristik berechnet zunächst die Wohlfahrt als Differenz aus Zahlungsbereitschaft und Kosten und ordnet die Kunden entsprechend der maximalen Wohlfahrt zu, sodass gilt $b(k) = \max_b \{R_{kb} - VK_b\}$. Bei Gleichheit wird wiederum zufällig eine Zuordnung ausgewählt.

Beide Heuristiken führen zu zulässigen Zuordnungen, in dem Sinne, dass mindestens ein Preisschema existiert, bei dem sich alle Kunden gemäß der First-Choice Regel für die ihnen zugeordneten Bündel entscheiden. Im ersten Fall ist dieses gegeben durch Preise von 0, im zweiten durch Preise in Höhe der Kosten ($p_b = VK_b$). Neben der einfachen Berechnung fällt *MaxR* durch die Tatsache auf, dass im Graph der erzeugten Zuordnung ausschließlich positive Kantenbewertungen vorliegen, während *MaxW* Zuordnungen mit tendenziell größerer Wohlfahrt erzeugt. Da der Gesamtdeckungsbeitrag neben der Konsumentenrente einen Teil der Wohlfahrt darstellt, ist sinnvollerweise die Wohlfahrt im ersten Schritt zu maximieren, um in weiteren Schritten Zuordnungen

[200] Die Maximum Reservation Price Heuristik wurde vorgeschlagen in Shioda et al. (2009), S. 11f.
[201] Maximum Welfare Heuristik, vgl. Dobson/Kalish (1988), S. 115.

und Preise mit höheren Gesamtdeckungsbeiträgen erzeugen zu können. Dobson und Kalish selbst verweisen zusätzlich darauf, dass durch Wettbewerb eher wohlfahrtsmaximale Gleichgewichte erreicht werden, sodass es für Unternehmen sinnvoll sei, frühzeitig diese Wohlfahrtspositionen durch eigene Produkte zu besetzen und im Zeitablauf nur Preisanpassungen vorzunehmen.[202]

Im Anschluss an die Initialisierung wird mit Hilfe der *DK Reassignment* Heuristik, deren Ablauf in Abbildung 3.14 dargestellt ist, eine Verbesserung der Anfangszuordnung angestrebt.[203]

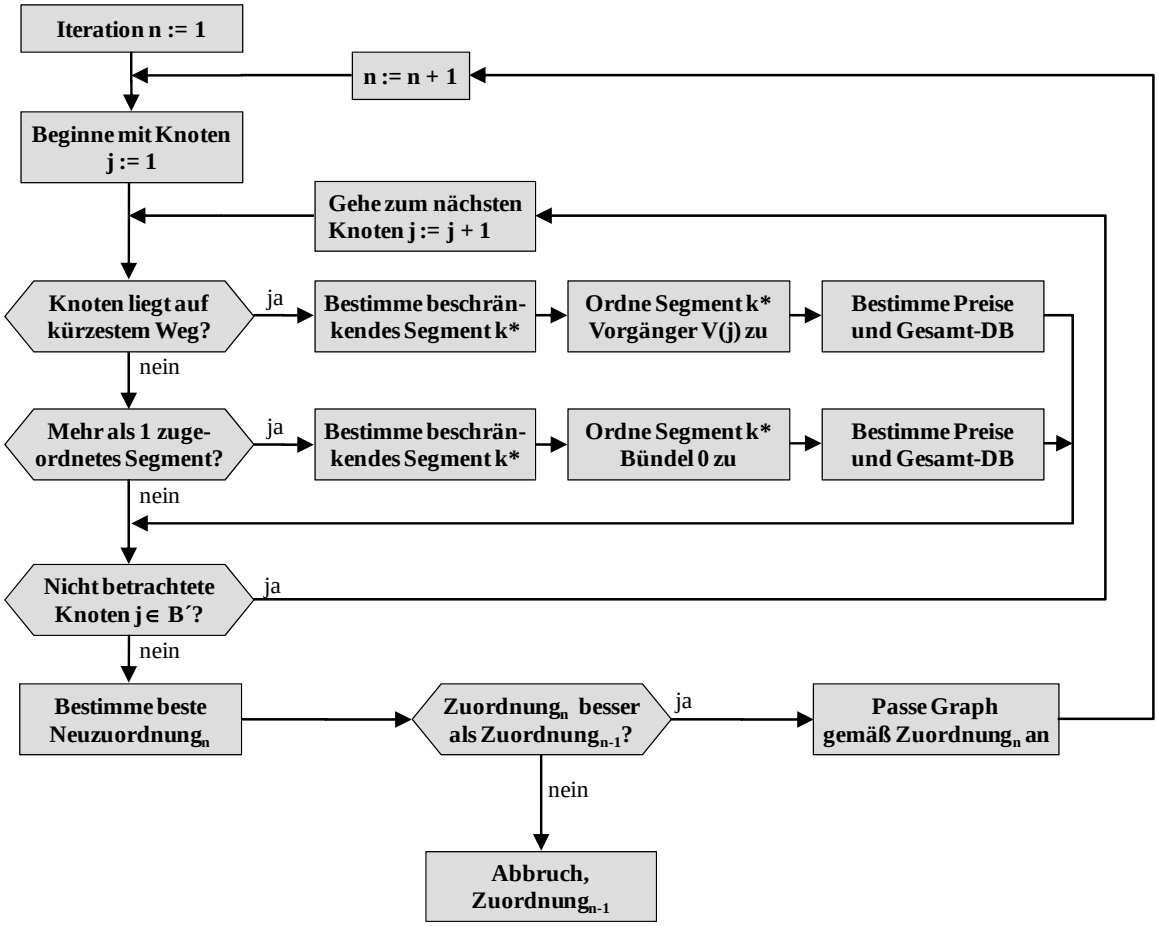

Abbildung 3.14: Ablauf der *DK Reassignment*-Heuristik zur Verbesserung der Zuordnung

Meist entsprechen die Bündelpreise nicht den Reservationspreisen der zugeordneten Segmente, können jedoch nicht erhöht werden, da sonst einige Segmente nicht mehr die ihnen zugeordneten Bündel kaufen. Die Idee ist nun, diejenigen Segmente zu identifizieren, die eine Preiserhöhung verhindern, und diese anderen Bündeln zuzuordnen. Dazu arbeitet die iterative Heuristik mit einem reduzierten Graphen, der nur Knoten mit zugeordneten Segmenten und Pfeile von kürzesten Wegen enthält. In jedem Iterationsschritt wird für jedes Bündel geprüft, ob eines der zugeordneten Segmente eine

[202] Vgl. Dobson/Kalish (1988), S. 115; Dobson/Kalish (1993), S. 164.
[203] Für einen Pseudocode vgl. Shioda et al. (2009), S. 12.

Preiserhöhung verhindert. Dazu sind zwei Punkte zu überprüfen. Erstens ist zu klären, ob der betrachtete Knoten $j \in B'$ auf einem kürzesten Weg liegt, d.h. ob mindestens ein ausgehender Pfeil (j,l) existiert. Ist dies der Fall, begrenzt eines der zugeordneten Segmente die Preise der nachfolgenden Bündel. Es ist daher für den eingehenden Pfeil[204] (i,j) zu prüfen, welches zugeordnete Segment $k \in C_j$ zur Kantenbewertung a_{ij} geführt hat, wozu $k^* = \arg\min_{k \in C_j} \left(R_{kj} - R_{ki} \right)$ zu bestimmen ist. Dieses Segment wird vorläufig Bündel i zugeordnet und die optimalen Preise erneut bestimmt. Die vorläufige Zuordnung wird zusammen mit dem erreichten Gesamtdeckungsbeitrag gespeichert. Anschließend wird die ursprüngliche Zuordnung wieder hergestellt und das Verfahren fortgesetzt.

Der zweite zu prüfende Punkt ist, ob einem nicht auf einem kürzesten Weg liegenden Knoten[205] mehrere Segmente zugeordnet sind. Dann ist Segment k^* mit der niedrigsten Zahlungsbereitschaft durch $k^* = \min_{k \in C_j} R_{kj}$ zu identifizieren und vorläufig dem 0-Bündel zuzuordnen. Wiederum sind optimale Preise zu bestimmen und die Zuordnung mit dem Gesamtdeckungsbeitrag zu speichern. Auf diese Weise werden alle Knoten bzw. Bündel untersucht. Anschließend wird die vorläufige Neuzuordnung ausgewählt, deren Gesamtdeckungsbeitrag am größten ist. Ist der Gesamtdeckungsbeitrag der gewählten Neuzuordnung größer als der Gesamtdeckungsbeitrag der aktuellen Zuordnung aus Iteration n-1, dann wird die Neuzuordnung zur aktuellen Zuordnung und der Algorithmus beginnt eine neue Iteration. Ist dies nicht der Fall, bricht das Verfahren ab und gibt als Lösung die Zuordnung aus Iteration n-1 aus.

Neben der *MaxW*-Eröffnungs- sowie der *Reassignment*-Verbesserungsheuristik schlagen Dobson und Kalish eine einstufige *Price Greedy*-Konstruktionsheuristik vor, deren Ablauf in Abbildung 3.15 dargestellt ist.[206] Dieses Verfahren versucht nicht, iterativ gegebene Zuordnungen zu verbessern, sondern in einem Durchlauf ein gutes Preissystem zu entwickeln.

Dazu wird zunächst heuristisch die wohlfahrtsmaximale Zuordnung bestimmt, indem in jedem Schritt gierig das Bündel hinzugefügt wird, welches die Wohlfahrt am stärksten steigert, was bei Vernachlässigbarkeit von Fixkosten zur gleichen Zuordnung wie *MaxW* führt. Anschließend wird der Beitrag jedes Bündels zur Gesamtwohlfahrt bestimmt und die Bündel in eine absteigende Rangfolge gebracht. Entsprechend dieser Rangfolge wird versucht, die Bündel in das Produktprogramm aufzunehmen.

---

[204] In einem aus kürzesten Wegen bestehender Graph hat jeder Knoten genau einen eingehenden Pfeil, sofern alle kürzesten Wege eindeutig sind, d.h. kein Knoten j über zwei identisch lange Wege von Knoten 0 aus erreicht werden kann.

[205] Der Knoten ist dann eine Senke.

[206] Für einen Pseudocode vgl. Dobson/Kalish (1993), S. 167.

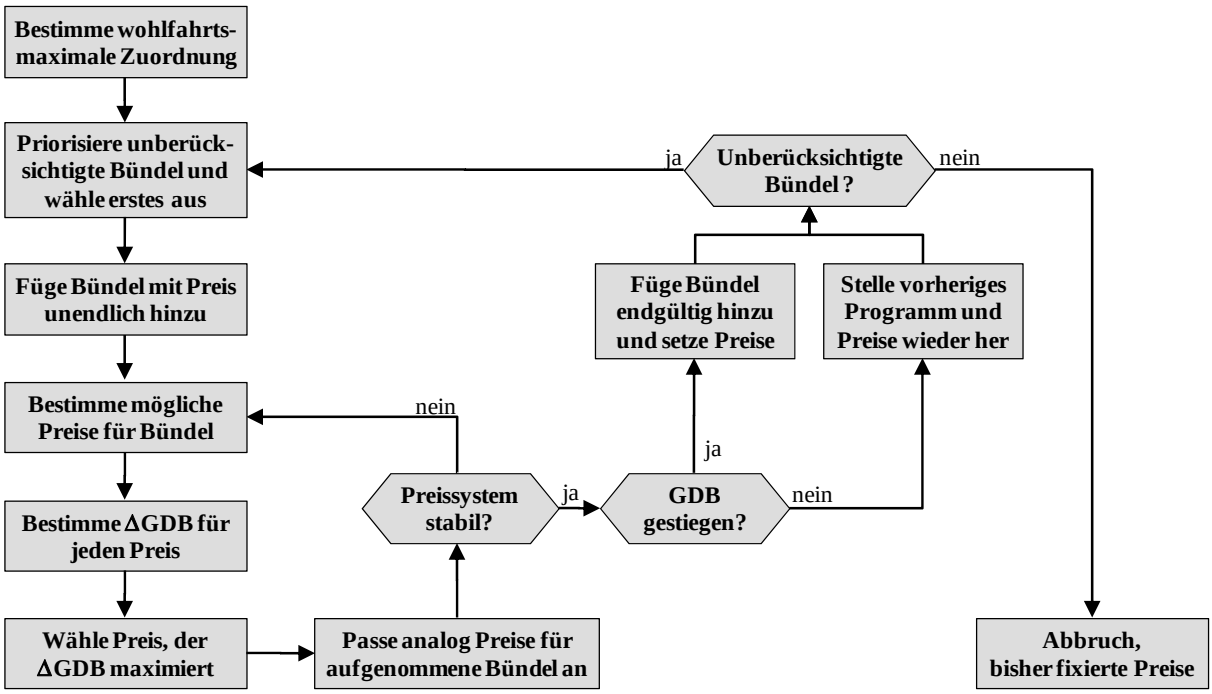

Abbildung 3.15: Ablauf der *DK Price Greedy*-Heuristik

Ein Bündel erhält zunächst einen unendlich hohen Preis, bevor weitere mögliche Preise bestimmt werden. Dazu wird für jedes Segment berechnet, wie hoch der Preis des neuen Bündels maximal sein darf, damit es von diesem Segment gekauft wird. Der maximale Preis, den Kunde k bereit ist für Bündel b zu bezahlen, ist der Preis seiner aktuellen Wahl zuzüglich der Differenz der Reservationspreise und somit gegeben durch $v_{kb} = p_{b(k)} + R_{kb} - R_{kb(k)}$. Danach werden die Veränderungen im Gesamtdeckungsbeitrag bei Wahl der verschiedenen Preise bestimmt und gierig der Preis festgelegt, der zur größten Steigerung führt. Anschließend wird analog für jedes bereits in das Produktprogramm aufgenommene Bündel geprüft, ob eine Preisanpassung den Gesamtdeckungsbeitrag erhöht. Ist dies bei mindestens einem Bündel der Fall, wird der Preis des hinzugefügten Bündels erneut geprüft und ggf. angepasst. Bei einer Preisanpassung werden wiederum die Preise aller bisher aufgenommenen Bündel geprüft. Kann durch Preisanpassungen keine Verbesserung mehr erzielt werden, ist das Preissystem stabil und der ermittelte Gesamtdeckungsbeitrag des Produktprogramms mit dem Bündel wird verglichen mit dem bisherigen Programm ohne dieses Bündel. Weist das Produktprogramm ohne das untersuchte Bündel einen höheren Gesamtdeckungsbeitrag auf, wird das vorherige Programm und die vorherigen Preise wieder hergestellt. Andernfalls wird das untersuchte Bündel endgültig in das Produktprogramm aufgenommen und alle Preise gemäß den Berechnungen angepasst. Existieren noch weitere bisher nicht betrachtete Bündel, wird nun das gemäß der Rangfolge nächste analog untersucht, ansonsten bricht das Verfahren ab. Als Lösung werden die bisher aufgenommenen Bündel mit den ermittelten Preisen ausgegeben.

3.4.3 Exemplarischer Einsatz der Heuristiken

Die Funktionsweisen der Heuristiken seien an einem Beispiel verdeutlicht. Gegeben sind drei Bündel, die gemäß der Tabelle A.4 im Anhang spezifiziert sind. Durch das gegebene Design und durch die in Tabelle A.5 bis Tabelle A.8 gegebenen Kosten und Zahlungsbereitschaften resultieren für die drei Bündel die in Tabelle 3.4 dargestellten Kosten und Zahlungsbereitschaften der 4 jeweils 10 Kunden umfassenden Segmente.

Tabelle 3.4: Kosten und Zahlungsbereitschaften für drei gegebene Bündel

Bündel	**Kosten**	**Zahlungsbereitschaft von Kundensegment**			
		S_1	**S_2**	**S_3**	**S_4**
Bündel A	23.600,-€	41.500,-€	29.500,-€	23.000,-€	27.000,-€
Bündel B	21.700,-€	35.500,-€	24.500,-€	35.500,-€	27.000,-€
Bündel C	20.650,-€	31.750,-€	28.000,-€	33.500,-€	26.000,-€

Die Anwendung der Initialisierungsheuristiken *MaxR* und *MaxW* führt zu den in Tabelle 3.5 dargestellten Startzuordnungen. Aufgrund seiner Reservationspreise kann Segment S_4 nach *MaxR* sowohl Bündel A als auch Bündel B zugeordnet werden. In diesem Fall sei zufällig A gewählt.

Tabelle 3.5: Gegenüberstellung der Zuordnungen unterschiedlicher Initialisierungsheuristiken

Kunde	**Initialisierungsheuristik**	
	MaxR	MaxW
S_1	A	A
S_2	A	C
S_3	B	B
S_4	A/B	C

Ausgehend von der von *MaxR* erzeugten Startzuordnung ergibt sich der in Abbildung 3.16 dargestellte Graph.

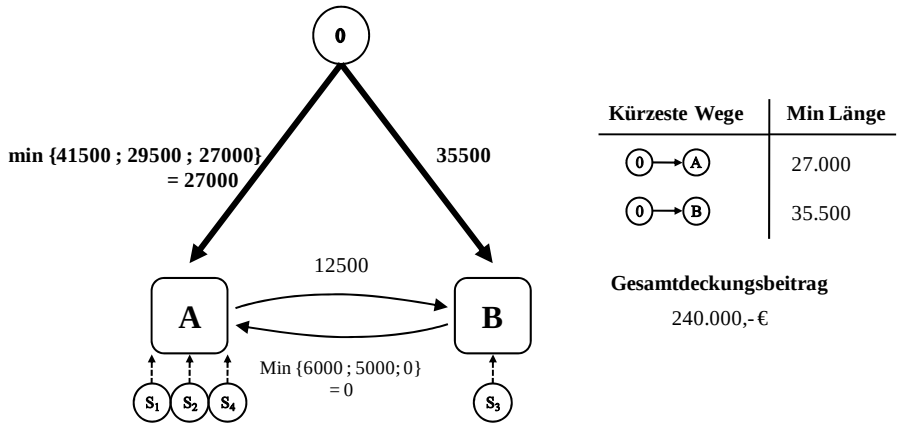

Abbildung 3.16: Kürzeste Wege nach *MaxR*-Zuordnung

Die kürzesten Wege fließen direkt von der Quelle zu den Bündeln. Dies signalisiert, dass Kunden bei Preiserhöhungen eines Bündels auf den Kauf verzichten und nicht auf andere Bündel ausweichen werden. Es besteht daher keine Kannibalisierungsgefahr zwischen den Bündeln. Entsprechend der Kantenbewertung sollte Bündel A zu einem Preis von 27.000,- € und Bündel B zu einem Preis von 35.500,- € angeboten werden. Der Preis von Bündel A ist jedoch aufgrund der geringen Zahlungsbereitschaft von Segment S_4 sehr niedrig. Im ersten Reassignment-Schritt wird daher geprüft, ob durch einer Zuordnung von Segment 4 zum 0-Bündel und damit dessen Nichtbedienung der Gesamtdeckungsbeitrag gesteigert werden kann. Es resultiert folgender Graph:

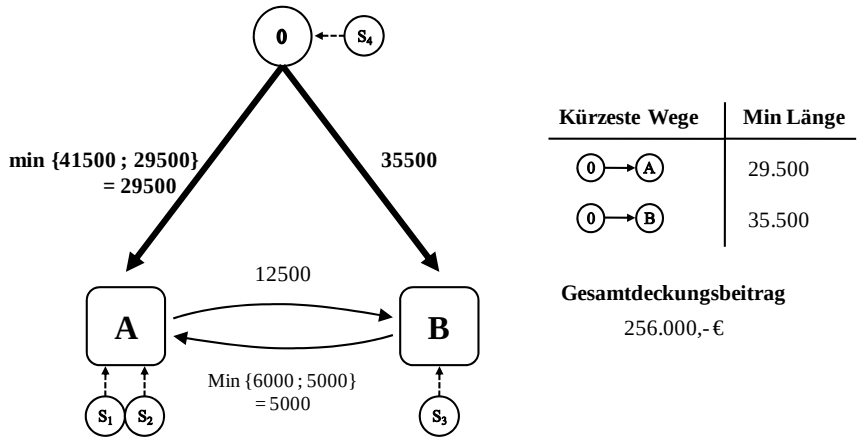

Abbildung 3.17: 1.Reassignment: Zuordnung von Segment S_4 zum 0-Bündel

Durch Nichtbedienung von Segment S_4 kann der Preis von Bündel A um 2.500,- € angehoben werden. Dadurch steigt der Gesamtdeckungsbeitrag durch die Segmente S_1 und S_2 um 50.000,- € während er sich jedoch durch die Nichtbedienung von S_4 um 34.000,- € verringert. Insgesamt verbleibt jedoch eine Nettosteigerung von 16.000,- €, sodass das Reassignment fixiert wird und weitere geprüft werden. Segment S_2 begrenzt den Preis von Bündel A, es wird daher geprüft, ob dessen Zuordnung zum 0-Bündel den Gesamtdeckungsbeitrag steigert. Es resultiert folgender Graph:

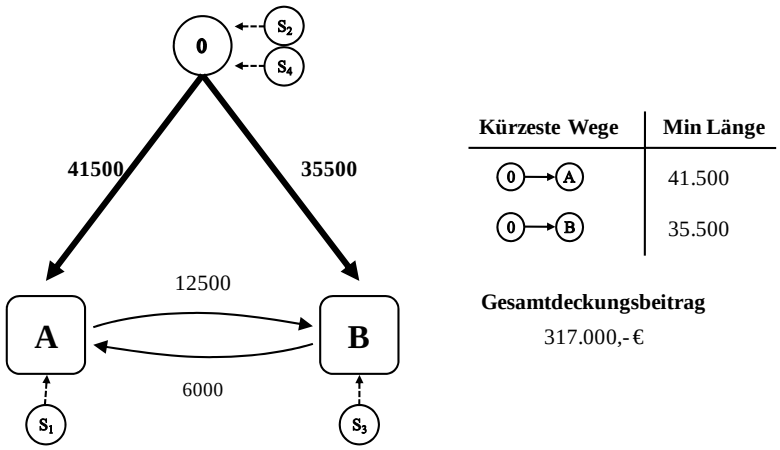

Abbildung 3.18: 2. Reassignment und finale Zuordnung: Zuordnung von Segment S_2 zum 0-Bündel

Durch das 2. Reassignment kann der Gesamtdeckungsbeitrag gesteigert werden. Da nun jedes Bündel genau von einem Kunden und die kürzesten Wege direkt vom 0-Bündel zu den Bündeln verlaufen, sind die finale Zuordnung und das finale Preissystem erreicht. Bündel A wird zu einem Preis von 41.500,- € an Segment S_1 und Bündel B zu einem Preis von 35.500,- € an Segment S_3 verkauft. Damit wird ein Gesamtdeckungsbeitrag von 317.000,- € realisiert.

Wird die *MaxW*-Heuristik zur Erzeugung der Startzuordnung eingesetzt, ergibt sich der in Abbildung 3.19 dargestellte Graph.

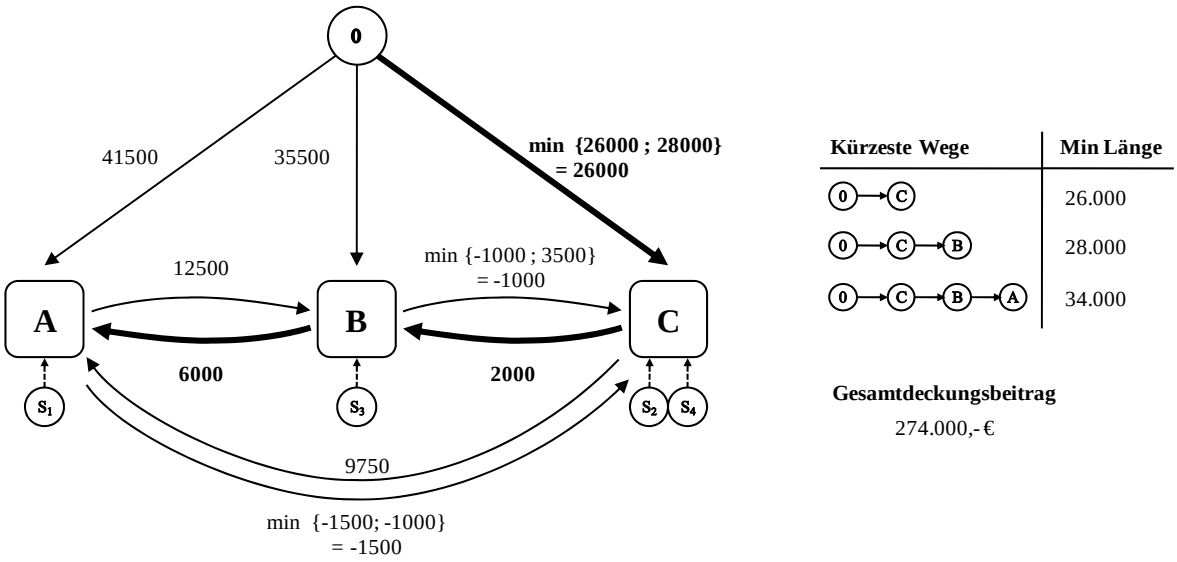

Abbildung 3.19: Kürzeste Wege nach *MaxW*-Zuordnung

Es wird deutlich, dass die kürzesten Wege nicht immer direkt vom 0-Bündel zu den Bündeln führen. Bei höheren Preisen von A würde Segment S_1 zu Bündel B und bei höheren Preisen von B Segment S_3 zu Bündel C wechseln. Es besteht daher die Gefahr der Kannibalisierung. In diesem Fall liegen Bündel B und C auf kürzesten Wegen, Bündel B liegt auf dem kürzesten Weg zu A und Bündel C liegt auf dem kürzesten Weg zu A und B. Es sind daher zwei Reassignments möglich. Erstens kann Segment

S_4 dem 0-Bündel zugeordnet werden, zweitens kann Segment S_3 Bündel C zugeordnet werden. Bei Reassignment von Segment S_4 ergibt sich der in Abbildung 3.20 dargestellte Graph, während das Reassignment von S_3 zum Graphen in Abbildung 3.21 führt.

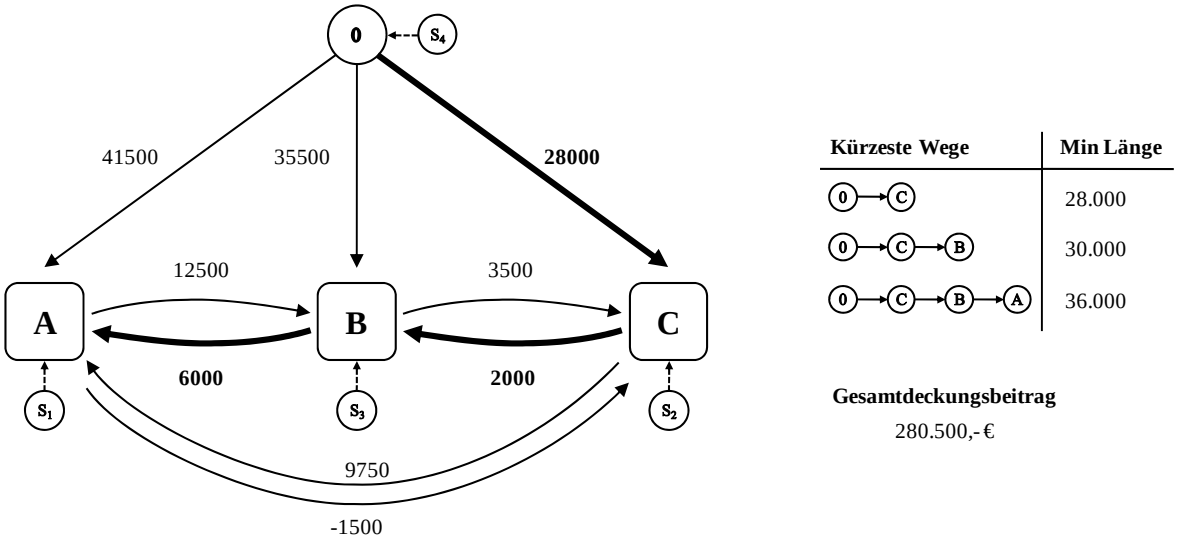

Abbildung 3.20: Mögliches 1. Reassignment: Zuordnung von Segment S_4 zum 0-Bündel

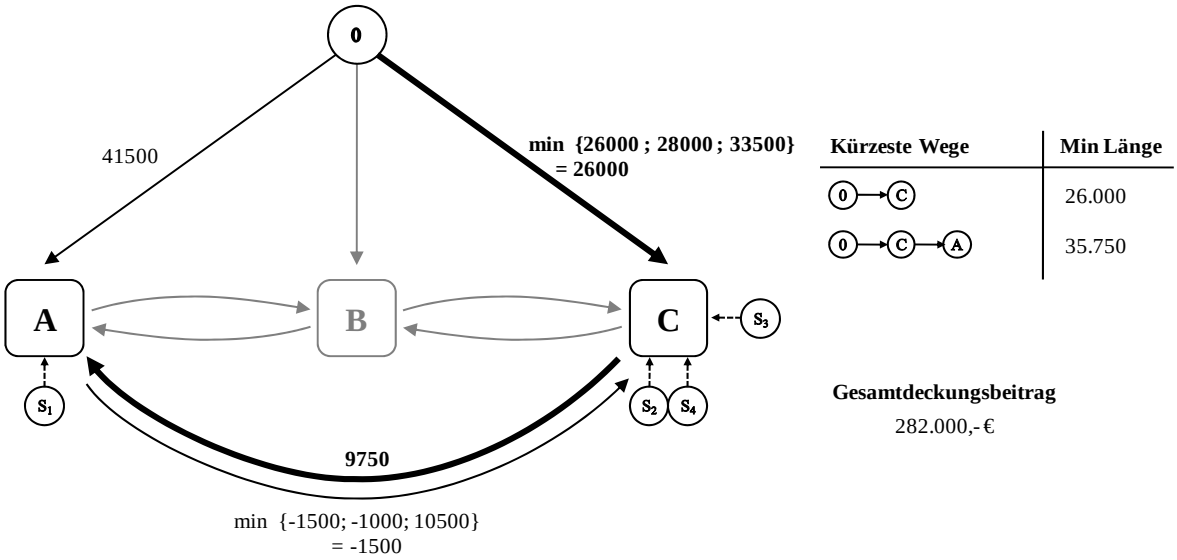

Abbildung 3.21: Mögliches 1. Reassignment: Zuordnung von Segment S_3 zu Bündel C

Das Reassignment von S_3 zu Bündel C steigert den Gesamtdeckungsbeitrag stärker, sodass dieses Reassignment ausgewählt und fixiert wird. Ausgehend von dieser Zuordnung kann Segment S_4 zum 0-Bündel zugeordnet werden, was in Abbildung 3.22 dargestellt ist.

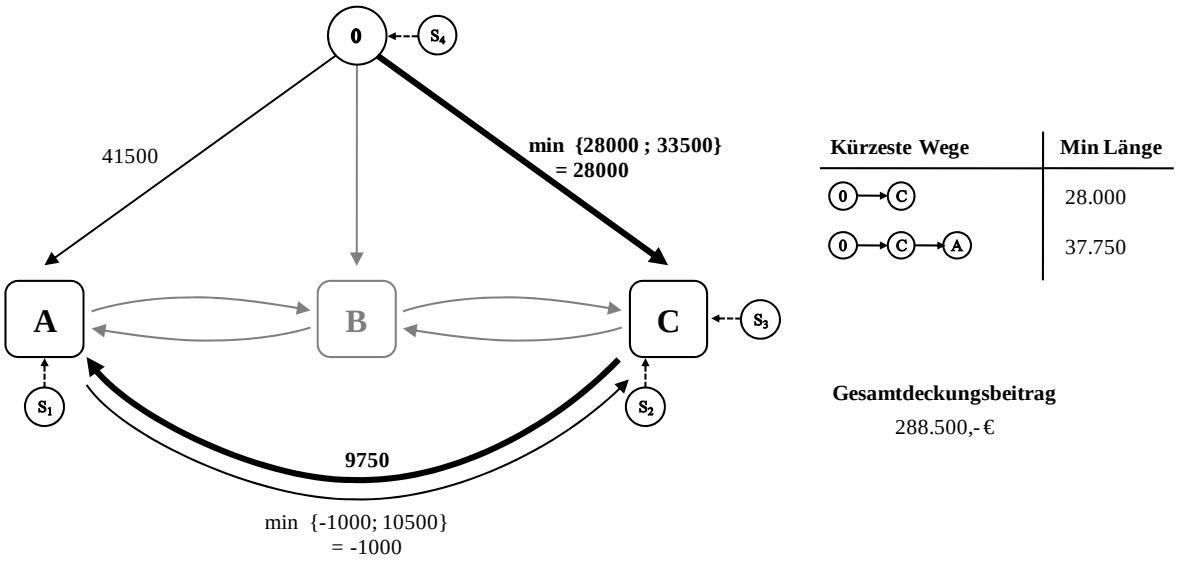

Abbildung 3.22: 2. Reassignment: Zuordnung von Segment S_4 zum 0-Bündel

Als drittes Reassignment kann, wie in Abbildung 3.23 ersichtlich, S_2 dem 0-Bündel zugeordnet werden. Auch dieses Reassignment führt zu einer Steigerung des Gesamtdeckungsbeitrags und wird daher fixiert. Da jedes Bündel von einem Segment gekauft wird und die kürzesten Wege wiederum direkt vom 0-Bündel zu den Bündeln verlaufen, kann kein weiteres Reassignment durchgeführt werden. Die finale Zuordnung führt zu einem gegenüber der *MaxR*-Startlösung leicht niedrigerem Gesamtdeckungsbeitrag von 307.500,- €.

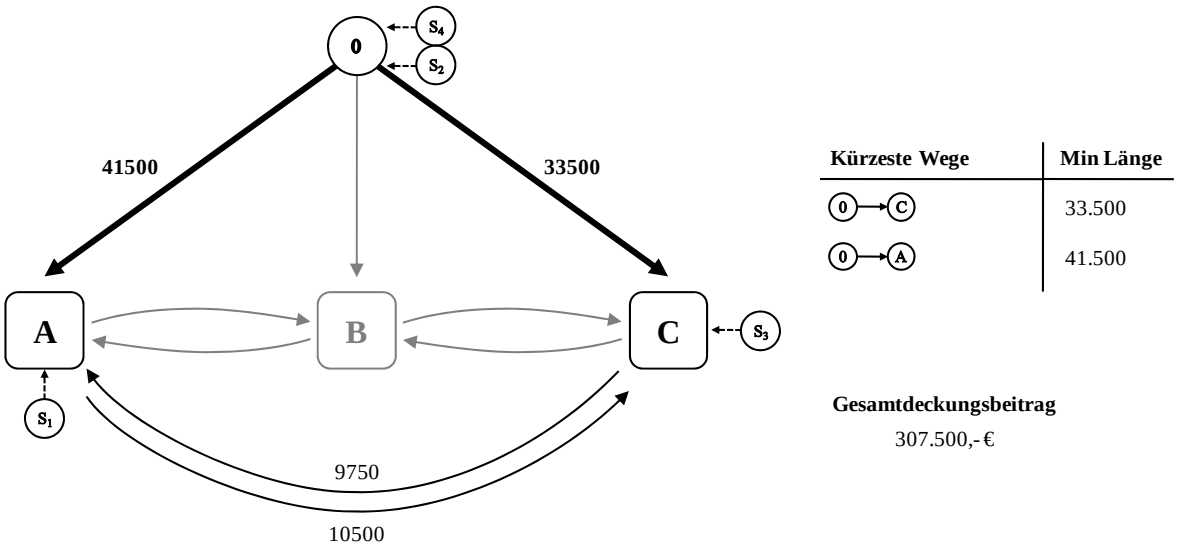

Abbildung 3.23: 3. Reassignment und finale Zuordnung: Zuordnung von Segment S_2 zum 0-Bündel

Dobson Kalish Price Greedy-Heuristik

Die Dobson Kalish *Price Greedy*-Heuristik beginnt mit der Bestimmung der wohlfahrtsmaximalen Lösung. Diese ist, wie bereits erläutert, bei Vernachlässigbarkeit von Fixkosten äquivalent zur *MaxW* Zuordnung. Daher wird auf eine Darstellung der Ermittlung der wohlfahrtsmaximalen Zuordnung verzichtet. Die erfolgte Zuordnung und die Wohlfahrt der einzelnen Segmente und der Wohlfahrtsbeitrag der Bündel sind in Tabelle 3.6 dargestellt.

Tabelle 3.6: Startzuordnung und Wohlfahrt durch die Price Greedy Heuristik

Bündel	**Wohlfahrt von Segment**				**Summe**
	S_1	**S_2**	**S_3**	**S_4**	
A	179.000,- €				179.000,-€
B			138.000,- €		138.000,-€
C		73.500,- €		53.500,-€	127.000,-€

Gemäß Tabelle 3.6 trägt Bündel A mehr als Bündel B und dieses wiederum mehr als Bündel C zur Gesamtwohlfahrt von 444.000,- € bei. Als Rangfolge zur Aufnahme in das Produktprogramm ergibt sich damit A, B, C. Bei Hinzufügen von Bündel A sind basierend auf den Reservationspreisen der Kunden vier verschiedene Preise möglich, wie sie in Tabelle 3.7 dargestellt sind.

Tabelle 3.7: Deckungsbeitragsänderungen bei unterschiedlichen Preisen von Bündel A

Mögliche Preise	**Käufer**	**GDB-Zuwachs**	**GDB-Einbuße**	**ΔGDB**
41.500,- €	**1**	**179.000,- €**	**0,- €**	**179.000,- €***
29.500,- €	1,2	118.000,- €	0,- €	118.000,- €
27.000,- €	1,2,4	102.000,- €	0,- €	102.000,- €
22.000,- €	1,2,3,4	-24.000,- €	0,- €	-24.000,- €

Es zeigt sich, dass der Gesamtdeckungsbeitrag am stärksten bei Wahl des höchsten Preises gesteigert wird. Gemäß der Rangfolge wird als nächstes Bündel B dem Produktprogramm hinzugefügt, siehe Tabelle 3.8.

Tabelle 3.8: Deckungsbeitragsänderungen bei unterschiedlichen Preisen von Bündel B und gegebenem Bündel A zum Preis von 41.500,- €

Mögliche Preise	Käufer	GDB-Zuwachs	GDB-Einbuße	ΔGDB
35.500,- €	**3**	**138.000,- €**	**0,- €**	**138.000,- €***
27.000,- €	1,3,4	159.000,- €	-179.000,- €	-20,000,- €
24.500,- €	1,2,3,4	112.000,- €	-179.000,- €	-67.000,- €

Nur bei Wahl des höchsten Preises wird der Gesamtdeckungsbeitrag gesteigert. Liegt der Preis bei 27.000,- € oder darunter, tritt Kannibalisierung auf. In diesem Fall werden nicht nur die beiden Nichtkäufer S_3 und S_4 Bündel B erwerben, sondern S_1 von Bündel A zu Bündel B wechseln, wodurch die Deckungsbeitragseinbußen stärker als die Deckungsbeitragszuwächse sind. Nach Wahl des höchsten Preises erfolgt eine Überprüfung der Preise von Bündel A, die hier aufgrund nicht notwendiger Anpassung nicht dargestellt ist. Als letzter Schritt wird probehalber Bündel C in das Produktprogramm aufgenommen.

Tabelle 3.9: Deckungsbeitragsänderungen bei unterschiedlichen Preisen von Bündel C und gegebenen Bündeln A und B

Mögliche Preise	Käufer	GDB-Zuwachs	GDB-Einbuße	ΔGDB
33.500,- €	3	128.500,- €	-138.000,- €	-9.500,- €
31.750,- €	1,3	222.000,- €	-317.000,-€	-95.000,-€
28.000,- €	1,2,3	220.500,- €	-317.000,- €	-96.500,- €
26.000,- €	1,2,3,4	214.000,- €	-317.000,- €	-103.000,- €

Durch Aufnahme von Bündel C verringert sich der Gesamtdeckungsbeitrag und da dies nicht durch eine Preisanpassung von Bündel A oder B kompensiert werden kann, wird Bündel C nicht aufgenommen und das Verfahren bricht ab. Das endgültige Produktprogramm besteht aus den Bündeln A und B, die zu den Preisen 41.500,- € bzw. 35.500,- € angeboten werden. Insgesamt wird dadurch ein Gesamtdeckungsbeitrag von 317.00,-€ erwirtschaftet. Die ermittelte Lösung entspricht der optimalen Lösung des Pricing-Problem i.w.S. gegeben durch (3.7) – (3.12).

3.4.4 Anpassung der Preisheuristiken für Bündel

Der Einsatz der Preisheuristiken setzt die Interpretation der Bündel als Produkte voraus. Werden annahmegemäß segmentviele Bündel angeboten, kann gezeigt werden, dass jeder Kunde maximal genau ein Bündel erwerben wird. Dies wird genau dann geschehen, wenn durch das Produktprogramm die Surplussuperadditivitätsbedingung eingehalten wird, d.h. die Kombination aus zwei Bündeln keinen größeren Surplus bietet als der Erwerb von nur einem Bündel. Im Produktliniendesign mit einer Produktlinien ist hingegen der gleichzeitige Erwerb mehrerer Produkte nicht vorgesehen, weswegen dieser Fall nicht von den Preisheuristiken berücksichtigt wird. Zur Gewährleistung der Gültigkeit des Preissystems und der Kundenzuordnung erfolgt daher im Anschluss an die Preisheuristik eine Überprüfung der Kundenwahl, deren Ablauf in Abbildung 3.24 dargestellt ist. Die Überprüfung beginnt mit Kombination der vorhandenen Bündel zu (überschneidungsfreien) Meta-Bündeln, wobei der Preis der Meta-Bündel auf die Summe der Preise der enthaltenden Bündel festgesetzt wird. Anschließend wird der Surplus jedes Kunden für alle Meta-Bündel bestimmt und geprüft, ob die Kunden bei ihrer Wahl bleiben oder stattdessen ein Meta-Bündel erwerben. Bei Abweichung von der durch die Preisheuristik bestimmten Zuordnung wird der Gesamtdeckungsbeitrag erneut unter Einbeziehung der Meta-Bündel berechnet und als Ergebnis ausgegeben.

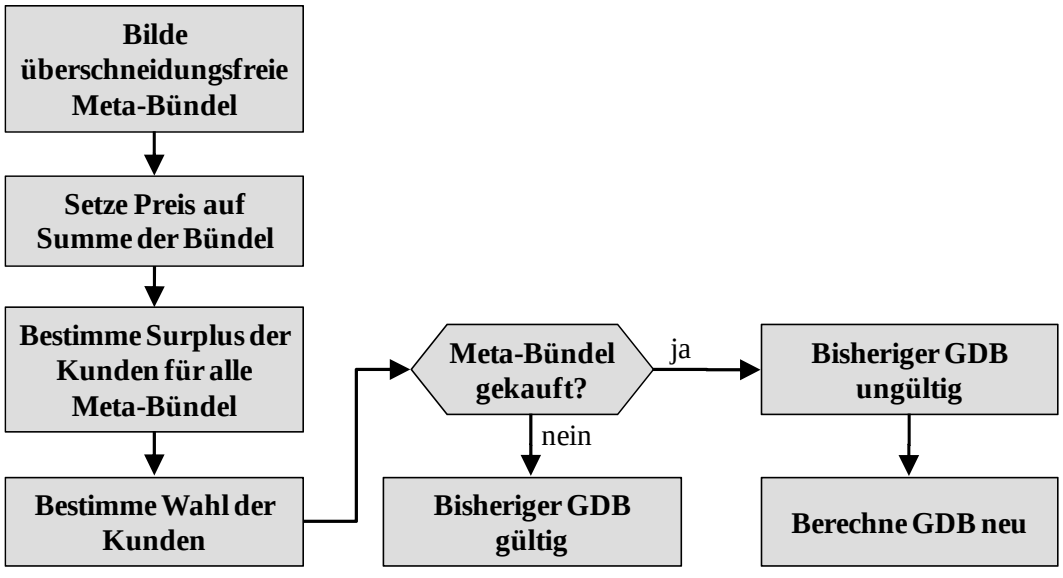

Abbildung 3.24: Ablauf des *MetaChecks* zur Überprüfung der Preisheuristiken

4 Experimentelle Evaluation des entwickelten Algorithmus

Der entwickelte Algorithmus wird im Weiteren durch eine experimentelle Studie evaluiert, wozu nacheinander folgende Fragen untersucht werden:

1. Wie lauten gute Parametereinstellungen für den Algorithmus?
2. Wie gut eignet sich der Algorithmus zur simultanen Bündelung?
3. Welchen Einfluss haben die Umweltfaktoren auf die Performance?

Zur Performancebewertung des entwickelten Algorithmus ist es unzureichend, anhand *eines* Beispiels die mit dem Algorithmus bestimmte Lösung mit der optimalen Lösung zu vergleichen. Vielmehr ist sicherzustellen, dass der Algorithmus in einer Vielzahl von Situationen zu guten, nahoptimalen Lösungen führt. Dazu sind verschiedene *Settings* mit jeweils unterschiedlichen *Realisationen* betrachteter *Umweltfaktoren* zu bilden. So sind mit den in Tabelle 4.1 dargestellten Umweltfaktoren und Realisationen $2 \cdot 3 \cdot 3 \cdot 3 = 54$ verschiedene Settings möglich. Pro Setting werden zufällig 20 *Instanzen* generiert. Anschließend wird die Performance des Algorithmus anhand der so entstandenen 1080 verschiedenen Instanzen bewertet.

Tabelle 4.1: Untersuchte Umweltfaktoren und betrachtete Realisationen

Umweltfaktor	**Betrachtete Realisationen**		
Anzahl Linien	2	4	
Anzahl Segmente	4	8	12
Produktkomplexität (#Merkmale, #Ausprägungen)	einfach (3, 2)	mittel (6, 3)	komplex (9, 4)
Zahlungsbereitschaftstyp	Typ I	Typ II	Typ III

Bei der Auswahl der Umweltfaktoren und Realisationen ist darauf zu achten, ein möglichst breites Spektrum realer Anwendungssituationen zu erfassen. Auf der anderen Seite können nicht beliebig viele Faktoren und Realisationen berücksichtigt werden, da die Anzahl zu berechnender Instanzen aufgrund der Kombinatorik mit zunehmender Anzahl dramatisch ansteigt. Es werden daher 2 und 4 Produktlinien betrachtet. Darüber hinaus werden mit 4, 8 und 12 betrachteten Segmenten unterschiedlich stark segmentierte Märkte untersucht. Der Faktor Produktkomplexität bezeichnet die zur Produktbeschreibung notwendige Anzahl von Merkmalen und Ausprägungen. Bei ein-

fach strukturierten Produkten werden bspw. die Varianten aller Linien durch 3 Merkmale und 2 Ausprägungen beschrieben. Dies gilt analog für die Realisationen „mittel“ und „komplex“. Mit dem Zahlungsbereitschaftstyp wird die Ähnlichkeit der Zahlungsbereitschaft der Kunden untereinander festgelegt. Für die Evaluation des Algorithmus ist die Kenntnis ausreichend, dass Zahlungsbereitschaften vom Typ 1 hauptsächlich auf einer kundenindividuellen Präferenzreihenfolge, Zahlungsbereitschaften vom Typ 3 hauptsächlich auf den Kosten der Ausprägungen und Zahlungsbereitschaften vom Typ 2 auf beiden gleichermaßen basieren. Typ 1 und 3 stellen somit Randfälle dar, bei denen die Zahlungsbereitschaften sich entweder ungewöhnlich stark voneinander unterscheiden bzw. ähneln. Zahlungsbereitschaften vom Typ 2 liegen zwischen den beiden Extremen und stellen damit am ehesten realitätsnahe Werte dar.

4.1 Ermittlung geeigneter Parametereinstellungen

Der entwickelte Algorithmus besitzt zahlreiche Einstellungsparameter[207], sodass vor einer Bewertung gute Parametereinstellungen auf Basis von Testinstanzen zu identifizieren sind.[208] Dazu werden für jede der durch Tabelle 4.1 gegebenen 54 Settings unabhängig von den zur Bewertung verwendeten Instanzen 10 Testinstanzen generiert, sodass insgesamt 540 Testinstanzen verwendet werden. Da einige Parameter reellwertig sind, existieren unendlich viele Parameterkonstellationen, sodass das Auffinden optimaler Einstellungen ein schwieriges Problem darstellt.[209] Selbst bei Beschränkung auf wenige Ausprägungen pro Parameter erweist sich ein vollständiges Ausprobieren aufgrund der vielfältigen Kombinationsmöglichkeiten und der benötigten hohen Anzahl von Algorithmenläufen als wenig praktikabel. Evolutionäre Algorithmen sind jedoch relativ robust gegenüber Änderungen ihrer Einstellungsparameter, sodass die Identifizierung des *Sweet Spots,* des Bereichs guter Einstellungen ausreichend ist.[210] Für ein systematisches Vorgehen wird hier eine heuristische Strategie verfolgt, die auf der Idee des Jacobi-Verfahrens[211] zur iterativen Lösung linearer Gleichungssysteme aufbaut. Der Idee folgend werden ausgehend von gewählten Starteinstellungen die Parameter in einem iterativen Prozess verbessert. Dazu werden die Parameter einzeln nacheinander untersucht, während die übrigen Parametereinstellungen konstant gehalten werden. Für den betrachteten Parameter werden mögliche Einstellungen analysiert und die beste Einstellung als zukünftige Standardeinstellung ausgewählt. Das Verfah-

[207] Vgl. Tabelle 3.2 auf S. 39.
[208] Dies wird auch als Parameter Tuning bezeichnet, vgl. Eiben et al. (1999), S. 124.
[209] Vgl. Eiben et al. (1999), S. 126.
[210] Vgl. de Jong (2007), S. 5.
[211] Vgl. Hanke-Bourgeois (2009), S. 77–82; Meister (2008), S. 69–74; Bollhöfer/Mehrmann (2004), S. 244f.

ren bricht ab, wenn in einem Iterationsschritt für keinen Parameter eine neue Standardeinstellung festgelegt wurde.

Die analysierten Parameter und untersuchten Einstellungen sind in Tabelle 4.2 dargestellt. Diese aufgrund subjektiver Einschätzung während der Entwicklungsphase als gut befundenen Standardeinstellungen werden jeweils für die restlichen Untersuchungen durch die beste innerhalb der Analyse gefundene Einstellung ersetzt. Da Parametereinstellungen jedoch nicht nur die Qualität, sondern auch die zur Ermittlung der Lösung benötigte Zeit beeinflussen, wird neben dem durchschnittlich erreichten Gesamtdeckungsbeitrag auch die Anzahl benötigter Generationen zur Beurteilung herangezogen. Die nach Abschluss aller Parameteruntersuchungen gefundenen Einstellungen werden bei den weiteren Berechnungen zur Evaluation des Algorithmus verwendet.

Tabelle 4.2: Untersuchte Parameter und Parametereinstellungen

Parameter	**Untersuchte Parametereinstellungen**			
Preisheuristik	MaxW + DK Reassignment	MaxR + DK Reassignment	DK Price Greedy	
Selektionsverfahren / Selektionsdruck	Linear Ranking 1,0	Linear Ranking 1,1	**Linear Ranking 1,6**	Linear Ranking 2,0
Mutationsraten (MR_{spec} , MR_{bun} , MR_{line})	*sehr niedrig* (0,5% ; 1,3% ; 0,1%)*	niedrig (1,0% ; 2,5% ; 0,3%)	**mittel (2,0% ; 5,0% ; 0,5%)**	hoch (4,0% ; 10,0% ; 1,0%)
Mixing Rate	*12,5%**	25%	**50%**	75%

Ausgangseinstellungen sind fett markiert

** In weiteren Iterationen geprüfte Einstellung*

Da dieselben Instanzen mit unterschiedlichen Parametereinstellungen gelöst werden, handelt es sich um verbundene Stichproben. Zur statistischen Prüfung, ob eine Einstellung zu signifikant besseren Ergebnisse führt, kommen in diesem Fall verschiedene parametrische (wie Differenzentest) und nichtparametrische Tests (wie Vorzeichentest oder Wilcoxon-Test) in Frage.[212] Aufgrund der unterschiedlichen Settings kann jedoch nicht davon ausgegangen werden, dass die resultierenden Gesamtdeckungsbeiträge symmetrisch, geschweige denn normalverteilt sind. Es wird daher mit dem Vorzeichentest ein nichtparametrischer, verteilungsfreier Test ausgewählt. Als Nullhypothese H_0 wird davon ausgegangen, dass zwei Parametereinstellungen X und Y zu gleich guten Ergebnissen führen. Dies ist dann der Fall, wenn X und Y in vielen Testinstanzen zu identischen Gesamtdeckungsbeiträgen führen, also $GDB_X = GDB_Y$ ist bzw. ungefähr gleich viele positive wie negative Vorzeichen der Differenz $GDB_X - GDB_Y$ vorkommen. Als Alternativhypothese H_1 wird im Weiteren stets geprüft, ob mit Einstellung X signifikant bessere Ergebnisse erzielt werden als mit Y. Zur Bewertung von

[212] Zu Tests bei verbunden Stichproben vgl. u.a. Bamberg et al. (2008), S. 186–191; Toutenburg/Heumann (2008), S. 145–147 und S. 179-186; Polasek (1997), S. 129–131 und S. 184-200.

Parametereinstellungen eignet sich der Test gut. Er ist robust gegenüber Ausreißern, also dem bei probabilistischen Heuristiken in einem Lauf möglichen zufälligen Erzielen sehr schlechter Ergebnisse. Dadurch reduziert sich die Gefahr der Überanpassung an die Testumgebung und die ermittelten Einstellungen sind mit größerer Wahrscheinlichkeit auch für in der Testumgebung nicht enthaltenen Settings geeignet.

4.1.1 Preisheuristik

Als erster Parameter wird die eingesetzte Preisheuristik untersucht. Möglich ist die Verwendung der *MaxR* bzw. *MaxW*-Initialheuristik kombiniert mit der Dobson-Kalish *Reassignment* Heuristik (Einstellung R1 bzw. W1) sowie die Dobson-Kalish *PriceGreedy* Heuristik (Einstellung P1). Mit den Einstellungen wird im Durchschnitt über alle Testinstanzen Gesamtdeckungsbeiträge von ca. 2,2 Mio. € (R1), 3,3 Mio. € (W1) bzw. 3,1 Mio. € (P1) erzielt. Die Standardabweichungen betragen ca. 3,4 Mio. € (R1), 3,7 Mio. € (W1) bzw. 3,2 Mio. € (P1). Die hohen Standardabweichungen zeigen die mit den Settings und Instanzen stark voneinander abweichenden Gesamtdeckungsbeiträge. Dies impliziert gleichzeitig, dass selbst bei Erfüllung der Anwendungsvoraussetzungen mit Hilfe parametrischer Tests keine signifikanten Unterschiede zwischen den Einstellungen festgestellt werden können. Im Vergleich führte jedoch Einstellung R1 nur 83-mal, Einstellung W1 hingegen 383-mal und Einstellung P1 96-mal zu den höchsten Gesamtdeckungsbeiträgen.[213] Die für den Differenzentest notwendige Anzahl positiver und negativer Differenzen sowie Gleichheit bzw. Bindungen im wechselseitigen Vergleich der drei Einstellungen sind in Tabelle 4.3 dargestellt.

Tabelle 4.3: Anzahl positiver und negativer Differenzen sowie Bindungen der drei Preisheuristiken im wechselseitigen Vergleich

pos. Diff.	R1 > W1:	67	R1 > P1:	169	W1 > P1:	435
Bindungen	R1 = W1:	22	R1 = P1:	2	W1 = P1:	0
neg. Diff.	R1 < W1:	451	R1 < P1:	369	W1 < P1:	105
Gesamt N	Gesamt:	540	Gesamt:	540	Gesamt:	540

Es zeigt sich, dass Einstellung R1 auch im wechselseitigen Vergleich seltener als W1 oder P1 zu besseren Ergebnissen führt. Unter Berücksichtigung eines Signifikanzniveaus von $\alpha = 0{,}05$ wird gemäß den Formeln des Vorzeichentests die Nullhypothese H_0 verworfen, sofern die Anzahl positiver Differenzen größer als 279 (Vergleich R1 und W1) bzw. 289 (Vergleich R1 und P1) ist. Da dies nicht der Fall ist, kann nicht davon ausgegangen werden, dass R1 zu besseren Ergebnissen führt als W1 oder P1. Um-

[213] Da in manchen Testinstanzen mehrere Einstellungen die höchsten Gesamtdeckungsbeiträge erzielten, ist die Summe mit 562 größer als die Anzahl der Testinstanzen (=540).

gekehrt sind die positiven Differenzen (W1-R1 bzw. P1-R1) mit 451 bzw. 369 deutlich größer als die Vergleichswerte. Einstellungen W1 und R1 führen damit zu signifikant besseren Ergebnissen als R1. Zum besseren Vergleich hat sich in der Statistik die Angabe der p-Werte anstelle konkreter Testentscheidungen eingebürgert. Die p-Werte geben das Signifikanzniveau an, bei dem die Nullhypothese gerade abgelehnt wird. Da z.B. der p-Wert im Vergleich R1-W1 größer als 0,999 ist, ist mit einer Wahrscheinlichkeit von über 99,9% die Ablehnung der Nullhypothese falsch. Die p-Werte aller Paarvergleiche können Tabelle 4.4 entnommen werden.

Tabelle 4.4: P-Werte des wechselseitigen Vorzeichentests der drei Preisheuristiken

X \ Y	R1	W1	P1
R1	-	>0,999	>0,999
W1	**<0,001**	-	**<0,001**
P1	**<0,001**	>0,999	-

Nach dem Vorzeichentest ergibt sich die statistisch stark abgesicherte Aussage, dass Einstellung W1 signifikant bessere Ergebnisse erzielt als P1 und beide wiederum signifikant bessere Ergebnisse als R1. Einstellung W1 (Preisheuristik *MaxW* plus *Reassignment*) wird daher als Standardeinstellung festgelegt. Im Hinblick auf die Laufzeit ist festzuhalten, dass mit Einstellung R1 durchschnittlich 132 (Std.Abw. 165,22), mit Einstellung W1 88 (Std.Abw. 124,82) und mit Einstellung P1 140 (Std.Abw. 170,09) Generationen bzw. Iterationen benötigt wurden. Einstellung W1 benötigte 378-mal, Einstellung R1 157-mal und Einstellung P1 117-mal die wenigsten Generationen.[214] Nach dem Vorzeichentest ergibt sich eindeutig, dass Einstellung W1 signifikant weniger Generationen benötigt als die beiden anderen Einstellungen. Anders als im Hinblick auf die Lösungsqualität, ist die Einstellung R1 der Einstellung P1 hinsichtlich der Laufzeit signifikant überlegen, wie auch Tabelle 4.5 belegt.

[214] Aufgrund von Gleichheit ist auch hier die Summe größer als der Stichprobenumfang.

Tabelle 4.5: Vergleich unterschiedlicher Preisheuristiken

Preisheuristik		N	Gesamtdeckungsbeitrag #Max	Ø (Std.Abw.)	Vorzeichentest (p-Werte)				Generationen (Iterationen) #Min	Ø (Std.Abw.)	Vorzeichentest (p-Werte)			
					X \ Y	R1	W1	P1			X \ Y	R1	W1	P1
R1	MaxR	540	83	2.248.251 (3.366.315)	R1	-	>0,999	>0,999	157	132 (165,22)	R1	-	>0,999	>0,999
W1	MaxW	540	383	3.340.892 (3.693.212)	W1	**<0,001**	-	**<0,001**	378	88 (124,82)	W1	**<0,001**	-	**<0,001**
P1	PriceGreedy	540	96	3.175.613 (3.611.622)	P1	**<0,001**	>0,999	-	117	140 (170,09)	P1	**<0,001**	**<0,001**	-

N = Anzahl von gerechneten Instanzen
#Max / #Min = Anzahl an Instanzen, bei denen mit dieser Einstellung der größte GDB erzielt bzw. die geringste Anzahl an Generationen benötigt wurden
Signifikante Unterschiede sind fett hervorgehoben
Gewählte Parametereinstellung ist grau hinterlegt

4.1.2 Selektionsdruck

Als zweiter Parameter wird der beim Linear Ranking verwendete Selektionsdruck SP untersucht. Wie Tabelle 4.6 zeigt, ist ein geringer Selektionsdruck von 1,1 zwar dem zufälligen Ziehen (SP 1,0) sowohl hinsichtlich Lösungsqualität als auch -geschwindigkeit überlegen, jedoch ist er seinerseits schlechter als der bisherige Standardwert 1,6 und der beim Linear Ranking maximal möglichen Selektionsdruck von 2,0. Aufgrund der signifikanten Unterschiede wird die Standardeinstellung des Parameters Selektionsdruck auf 2,0 geändert.

Tabelle 4.6: Anpassung des Selektionsdrucks nach Wahl der MaxW-Preisheuristik (Iteration 1)

Selektionsdruck		N	Gesamtdeckungsbeitrag #Max	Ø (Std.Abw.)	Vorzeichentest (p-Werte)					Generationen (Iterationen) #Min	Ø (Std.Abw.)	Vorzeichentest (p-Werte)			
					X \ Y	W1	W2	W3	W4			X \ Y	W1	W2	W3	W4
W1	1,6	540	239	3.340.892 (3.693.212)	W1	-	>0,999	**<0,001**	**<0,001**	157	132 (165,22)	W1	-	>0,999	**<0,001**	**<0,001**
W2	2,0	540	321	3.355.197 (3.720.301)	W2	**<0,001**	-	**<0,001**	**<0,001**	378	88 (124,82)	W2	**<0,001**	-	**<0,001**	**<0,001**
W3	1,1	540	170	3.312.139 (3.663.431)	W3	>0,999	>0,999	-	**<0,001**	378	88 (124,82)	W3	>0,999	>0,999	-	**<0,001**
W4	1,0	540	165	3.311.951 (3.658.900)	W4	>0,999	>0,999	0,988	-	117	140 (170,09)	W4	>0,999	>0,999	>0,999	-

N = Anzahl von gerechneten Instanzen
#Max / #Min = Anzahl an Instanzen, bei denen mit dieser Einstellung der größte GDB erzielt bzw. die geringste Anzahl an Generationen benötigt wurden
Signifikante Unterschiede sind fett hervorgehoben
Gewählte Parametereinstellung ist grau hinterlegt

4.1.3 Mutationsrate

Hinsichtlich der Mutationsrate lässt sich, wie aus Tabelle 4.7 ersichtlich, feststellen, dass eine Halbierung der bei der Entwicklung standardmäßig verwendeten Mutationsraten zu signifikant besseren Ergebnissen führt. Zudem lässt sich mit dieser Einstellung die durchschnittlich benötigte Anzahl an Generationen noch einmal deutlich von 70 auf 46 reduzieren.

Tabelle 4.7: Anpassung der Mutationsrate nach Wahl der MaxW-Preisheuristik (Iteration 1)

Mutationsrate		N	Gesamtdeckungsbeitrag #Max	Ø (Std.Abw.)	Vorzeichentest (p-Werte)			Generationen (Iterationen) #Min	Ø (Std.Abw.)	Vorzeichentest (p-Werte)		
W2	2,0/5,0/0,5	540	268	3.355.197 (3.720.301)	x \ Y	W2	W5	W6	194	70 (99,39)	x \ Y	W2	W5	W6
					W2	-	**<0,001**	>0,999			W2	-	**<0,001**	>0,999
W5	4,0/10,0/1,0	540	155	3.231.038 (3.525.725)	W5	>0,999	-	>0,999	50	252 (227,35)	W5	>0,999	-	>0,999
W6	1,0/2,5/0,3	540	358	3.376.922 (3.726.282)	W6	**<0,001**	**<0,001**	-	409	46 (16,40)	W6	**<0,001**	**<0,001**	-

N = Anzahl von gerechneten Instanzen

#Max / #Min = Anzahl an Instanzen, bei denen mit dieser Einstellung der größte GDB erzielt bzw. die geringste Anzahl an Generationen benötigt wurden

Signifikante Unterschiede sind fett hervorgehoben

Gewählte Parametereinstellung ist grau hinterlegt

4.1.4 Mixing Rate

Im Gegensatz zu den bisherigen Untersuchungen zeigt der Vergleich unterschiedlicher Mixing Raten ein weniger eindeutiges Bild. Hinsichtlich der Lösungsgüte kann auf einem 5% Signifikanzniveau ein signifikanter Unterschied zwischen einer Mixing Rate von 25% gegenüber 75% festgestellt werden. Zudem zeigen die Werte zwar keinen signifikanten Unterschied, jedoch eher einen Vorteil einer 25%-igen Mixing Rate gegenüber der Standardeinstellung. Da auch hinsichtlich der benötigten Lösungszeit kein signifikanter Unterschied festzustellen ist, kann die Mixing Rate beliebig 25% oder 50% betragen. Aufgrund der leichten Vorteile wird die Einstellung auf 25% geändert.

Tabelle 4.8: Anpassung der Mixing Rate nach Wahl der MaxW-Preisheuristik (Iteration 1)

Mixing Rate		N	Gesamtdeckungsbeitrag #Max	Ø (Std.Abw.)	Vorzeichentest (p-Werte)			Generationen (Iterationen) #Min	Ø (Std.Abw.)	Vorzeichentest (p-Werte)		
W6	50%	540	266	3.376.922 (3.726.282)	x \ Y	W6	W7	W8	239	46 (16,40)	x \ Y	W6	W7	W8
					W6	-	0,099	0,938			W6	-	0,172	0,242
W7	75%	540	238	3.376.213 (3.746.417)	W7	0,945	-	0,976	232	46 (16,52)	W7	0,892	-	0,774
W8	25%	540	286	3.391.058 (3.779.452)	W8	0,11	**0,048**	-	234	47 (25,40)	W8	0,837	0,32	-

N = Anzahl von gerechneten Instanzen

#Max / #Min = Anzahl an Instanzen, bei denen mit dieser Einstellung der größte GDB erzielt bzw. die geringste Anzahl an Generationen benötigt wurden

Signifikante Unterschiede sind fett hervorgehoben

Gewählte Parametereinstellung ist grau hinterlegt

4.1.5 Zusammenfassung

Auf die Darstellung der Tabellen für die übrigen Iterationen wird hier verzichtet und auf den Anhang verwiesen. Stattdessen zeigt Tabelle 4.9 die geprüften Parameter und Einstellungen sowie die festgelegten Standardeinstellungen.

Tabelle 4.9: Untersuchte und gewählte Parametereinstellungen nach Wahl der MaxW-Preisheuristik

Iteration	Parameter	Bez.	Einstellung	
I	Selektionsdruck	W1	1,6	
		W2	2,0	*
		W3	1,1	
		W4	1,0	
	Mutationsrate	W2	2,0/5,0/0,5	
		W5	4,0/10,0/1,0	
		W6	1,0/2,5/0,3	*
	Mixing Rate	W6	50%	
		W7	75%	
		W8	25%	*
II	Selektionsdruck	W8	2,0	
		W9	1,6	*
		W10	1,1	
		W11	1,0	
	Mutationsrate	W9	1,0/2,5/0,3	*
		W12	2,0/5,0/0,5	
		W13	0,5/1,3/0,1	
	Mixing Rate	W9	25%	*
		W14	50%	
		W15	12,5%	

*Gewählte Parametereinstellung ist grau hinterlegt

Die besten Ergebnisse wurden mit der *MaxW*-Preisheuristik, einem Selektionsdruck von 1,6, Mutationsraten von 1,0 / 2,5 / 0,3 und einer Mixing Rate von 25% erzielt. Da mit einer größeren Populationsgröße und einer höheren Anzahl an Nachkommen stets bessere Ergebnisse zu erzielen sind, ist die Überprüfung dieser Parameter mit der vorgestellten Methodik wenig zielführend. Stattdessen werden diese Parameter im nächsten Kapitel untersucht, um die Stärke der Performancesteigerung ggf. der Laufzeitverlängerung gegenüberstellen zu können. Als Standardeinstellungen werden zunächst eine Populationsgröße von 100 und eine sechsfache Anzahl an Nachkommen gewählt.

4.2 Eignung des Algorithmus zur simultanen Bündelung

4.2.1 Beurteilungsmaße

Wie in Kapitel 3.1.1 diskutiert, ist die Average Case Performance von Heuristiken in Bezug auf ihre Lösungsgüte und ihre Laufzeit durch eine umfangreiche experimentelle Studie zu evaluieren. Zum Vergleich der heuristisch ermittelten mit der optimalen Lösung wurden die Umweltfaktoren bewusst so gewählt, dass eine optimale Lösung mit Hilfe von Standardsoftware zur linearen Programmierung[215] bestimmbar ist. Unabhängig von den Instanzen zur Einstellung der Parameter wurde für jedes der 54 durch die Umweltfaktoren in Tabelle 4.1 gegebenen Settings 20 Instanzen erzeugt, sodass insgesamt 1080 Instanzen zur Verfügung stehen.

Die Lösungsgüte wird mit zwei Kriterien gemessen, zum einen mit dem *Erreichungsgrad EG*, zum anderen mit der *Hit Rate HR*. Der Erreichungsgrad für eine Instanz i ist der Quotient aus dem durch den Algorithmus ermitteltem und dem optimalen Gesamtdeckungsbeitrag. Er gibt an, wie viel Prozent des optimalen Zielfunktionswertes vom Algorithmus erreicht wurden. Formal berechnet er sich gemäß:

$$EG_i = \frac{GDB_i^{EA}}{GDB_i^{opt}} \cdot 100 \qquad (4.1)$$

Der Erreichungsgrad einer Instanz ist hingegen für die Algorithmusbewertung von untergeordneter Bedeutung, sodass als Performancekriterien der über alle Instanzen durchschnittliche Erreichungsgrad zusammen mit der Standardabweichung angegeben wird. Die Standardabweichung lässt einen Rückschluss auf die Performancestabilität zu. Je geringer die Standardabweichung, umso weniger schwanken die einzelnen Erreichungsgrade um den Mittelwert und umso zuverlässiger kann die Performance eingeschätzt werden. Es wird zwar keine formale Analyse der Worst-Case Performance durchgeführt, jedoch lässt der in allen Instanzen erreichte minimale Erreichungsgrad einen Rückschluss auf die Worst-Case Performance in realistischen Situationen zu.

Für den praktischen Einsatz eines Algorithmus ist es nicht nur wichtig, welchen durchschnittlichen Erreichungsgrad er erzielt, sondern auch wie wahrscheinlich er eine optimale Lösung ermittelt. Dies kann durch die sogenannte Hit Rate gemessen werden, welche die Anzahl optimal gelöster Instanzen angibt. Dadurch ist jedoch kein abschließender Eindruck möglich, ob die Heuristik in der Lage ist, nahoptimale Lösungen zu bestimmen. Dazu ist die Hit Rate zu verallgemeinern, indem alle Instanzen ge-

[215] Hier eingesetzt: FICO™ Xpress Optimization Suite der Fair Isaac Corporation mit Xpress-Ive in Version 1.19.01 und dem XPress-Optimizer in Version 19.00.00.

zählt werden, bei denen ein Erreichungsgrad oberhalb einer *Schwelle* α erzielt wurde. Formal berechnet sich die Hit Rate durch:

$$HR_\alpha = \sum_i h_i \qquad \text{mit } h_i = \begin{cases} 1 & \text{, falls } EG_i \geq \alpha \\ 0 & \text{, sonst} \end{cases} \tag{4.2}$$

Auch die Laufzeit wird mit zwei Kriterien gemessen, zum einen die benötigte Anzahl an Generationen, zum anderen die benötigte Rechenzeit in Sekunden. Da die Generationenanzahl rechnerunabhängig ist, ist sie in den Auswertungstabellen dargestellt, wohingegen die Rechenzeit aus Platzgründen nur im Text erwähnt wird. Für einen guten Überblick über das Laufzeitverhalten werden neben der durchschnittlich benötigten Anzahl an Generationen und der Standardabweichung auch die minimale und maximale Anzahl angegeben.

4.2.2 Ergebnis

Die Ergebnisse der Performanceanalyse sind in Tabelle 4.10 dargestellt. Mit der in Kapitel 4.1 ermittelten Parametereinstellung W9 wurde im Durchschnitt 95,9% der optimalen Lösung erreicht. Die Standardabweichung beträgt 4,9% und weist damit auf leicht schwankende Erreichungsgrade hin, sodass größere aber auch geringere Erreichungsgrade möglich sind. Der schlechteste Wert beträgt 70,9%. Von den 1080 Instanzen wurden jedoch knapp zwei Drittel, genauer 65,5% bzw. 707 Instanzen sehr gut (≥ 95%), 38,0% bzw. 410 Instanzen nahoptimal (≥ 99%) und 22,8% bzw. 246 Instanzen optimal gelöst. Dies zeigt, dass der Algorithmus in vielen Fällen zu sehr guten Ergebnissen führt. Im Durchschnitt wurden 53,19 Generationen benötigt, wobei die Standardabweichung mit 17,10 einen größeren Schwankungsbereich belegt. Minimal wurden 10, maximal 169 Generationen benötigt. Die Laufzeit betrug auf einem Desktop-PC[216] zwischen 44ms und 3,5min. Die kurze Laufzeit ermöglicht den Einsatz selbst in iterativen Planungsansätzen bei denen Produktprogramme basierend auf den Ergebnissen mit unterschiedlichen Vorgaben erneut optimiert werden.

Tabelle 4.10: Performance des Algorithmus mit Parametereinstellung W9

Parameter-einstellung	N	Erreichungsgrad		Hit Rate			Generationen	
		min	Durchschnitt (Std.Abw.)	100%	99%	95%	min/max	Durchschnitt (Std.Abw.)
W9	1080	0,709	0,959 (0,049)	246	410	707	10 / 169	53,19 (17,10)

N = Anzahl von gerechneten Instanzen
Erreichungsgrad = Zielfunktionswert EA ÷ Optimaler Zielfunktionswert
Hit Rate = Anzahl der Instanzen, deren heuristische Lösung einen Erreichungsgrad von mindestens 100%, 99%, 95% aufweist

[216] Die Berechnungen erfolgten auf einem PC mit Intel Core i7 Prozessor mit 2,67 GHz und 6 GB Arbeitsspeicher. Es wird jedoch nur einer der 8 Prozessorkerne für die Berechnungen verwendet.

4.2.3 Einfluss der Populationsgröße

Die Parameter Populationsgröße und Anzahl der Nachkommen sind in Kapitel 4.1 nicht untersucht worden, da steigende Werte stets zu besseren Ergebnissen führen. Die Höhe der Steigerung kann jedoch sinnvoll erst im Vergleich mit dem Optimum bewertet werden. Im Folgenden wird daher untersucht, wie sich der Erreichungsgrad bei Abweichung von den Standardeinstellungen (Populationsgröße = 100 und Anzahl der Nachkommen = 6-mal Populationsgröße) verändert. Tabelle 4.11 und Tabelle 4.12 zeigen die Performance des Algorithmus in Abhängigkeit von der Populationsgröße bzw. von der Anzahl der Nachkommen.

Tabelle 4.11: Performance des Algorithmus in Abhängigkeit von der Populationsgröße

Populations-größe	N	Erreichungsgrad		Hit Rate			Generationen	
		min	Durchschnitt (Std.Abw.)	100%	99%	95%	min/max	Durchschnitt (Std.Abw.)
25	1080	0,643	0,931 (0,063)	97	213	503	10 / 140	58,33 (21,74)
50	1080	0,712	0,946 (0,056)	175	304	611	10 / 121	54,18 (17,77)
100	1080	0,709	0,959 (0,049)	246	410	707	10 / 169	53,19 (17,10)
200	1080	0,717	0,965 (0,045)	314	470	751	10 / 123	53,61 (16,67)

Standardeinstellung ist grau hinterlegt
N = Anzahl von gerechneten Instanzen
Erreichungsgrad = Zielfunktionswert EA ÷ Optimaler Zielfunktionswert
Hit Rate = Anzahl der Instanzen, deren heuristische Lösung einen Erreichungsgrad von mindestens 100%, 99%, 95% aufweist

Sowohl eine Verdopplung der Populationsgröße auf 200 als auch eine Halbierung auf 50 verändert den durchschnittlichen Erreichungsgrad um weniger als einen Prozentpunkt. Die Verwendung einer größeren Population führt aber deutlich häufiger zum Auffinden des globalen Optimums. Von einer Reduzierung der Population auf 25 kann hingegen nur abgeraten werden, da der durchschnittliche Erreichungsgrad mit 93,1% ebenso wie die Hit Raten deutlich schlechter ausfallen. Auch wenn in allen Fällen die Anzahl benötigter Generationen nahezu konstant ist, halbiert sich bei einer Population von 50 die Laufzeit aufgrund weniger Berechnungen pro Generation, sodass im Schnitt nur noch zwischen 30ms und 1,5min benötigt werden. Überraschenderweise erhöht sich die Laufzeit bei einer Verdopplung nur unwesentlich und es ist mit einer Laufzeit von 1,5s bis 5min zu rechnen. Die Analyse zeigt, dass die Populationsgröße von 100 eine gute Balance zwischen Geschwindigkeit und Lösungsgüte darstellt. Je nach Zielsetzung kann jedoch auch nach oben bzw. unten abgewichen werden, wobei von einer Verkleinerung der Population um mehr als die Hälfte abzuraten ist.

4.2.4 Einfluss der Anzahl der Nachkommen

Wie aus Tabelle 4.12 ersichtlich, führt eine Verringerung der Anzahl an Nachkommen nicht nur zu deutlichen Einbußen im Erreichungsgrad und den Hit Raten, sondern gleichzeitig auch zu einem deutlichen Anstieg benötigter Generationen. Mit der neunfachen Anzahl an Nachkommen[217] lässt sich der Erreichungsgrad und die Hit Raten nochmal leicht steigern, während die durchschnittliche Anzahl an Generationen leicht sinkt. Die Laufzeit beträgt zwischen 1s und 3,5min. Die Ergebnisse zeigen, dass sich die Ergebnisse ohne Zeiteinbußen geringfügig steigern lassen, sofern die Anzahl an Nachkommen erhöht wird. Aus diesem Grund wird für die weiteren Berechnungen die Standardeinstellung auf die neunfache Anzahl an Nachkommen geändert.

Tabelle 4.12: Performance des Algorithmus in Abhängigkeit von der Anzahl an Nachkommen

Anzahl Nachkommen	N	Erreichungsgrad		Hit Rate			Generationen	
		min	Durchschnitt (Std.Abw.)	100%	99%	95%	min/max	Durchschnitt (Std.Abw.)
1x Population	1080	0,476	0,871 (0,123)	123	220	392	30 / 500	496,63 (36,19)
3x Population	1080	0,667	0,949 (0,055)	199	339	623	10 / 500	81,99 (90,71)
6x Population	1080	0,709	0,959 (0,049)	246	410	707	10 / 169	53,19 (17,10)
9x Population	1080	0,678	0,962 (0,048)	265	443	749	10 / 113	49,38 (15,38)

Standardeinstellung ist grau hinterlegt
N = Anzahl von gerechneten Instanzen
Erreichungsgrad = Zielfunktionswert EA ÷ Optimaler Zielfunktionswert
Hit Rate = Anzahl der Instanzen, deren heuristische Lösung einen Erreichungsgrad von mindestens 100%, 99%, 95% aufweist

4.2.5 Wiederholter Einsatz der Heuristik

Der entwickelte Algorithmus weist eine stabile Performance auf, sodass wiederholte Durchläufe mit identischen Parametereinstellungen im Durchschnitt zu ähnlichen Ergebnissen führen, wie Tabelle 4.13 belegt. Der durchschnittliche Erreichungsgrad liegt zwischen 96,2% und 96,4% und weist damit wie die Hit Raten nur sehr geringe Schwankungen auf. Gleiches gilt für die durchschnittlich benötigte Anzahl an Generationen und die Laufzeit.

[217] Anzahl der Nachkommen entspricht neun mal der Elternpopulation.

Tabelle 4.13: Performance des Algorithmus zur simultanen Bündelung mit identischen Einstellungen

Läufe	N	Erreichungsgrad		Hit Rate			Generationen	
		min	Durchschnitt (Std.Abw.)	100%	99%	95%	min/max	Durchschnitt (Std.Abw.)
Lauf W9.9x.1	1080	0,678	0,962 (0,048)	265	443	749	10 / 113	49,38 (15,38)
Lauf W9.9x.2	1080	0,614	0,963 (0,048)	274	428	757	10 / 108	49,65 (15,32)
Lauf W9.9x.3	1080	0,641	0,964 (0,046)	282	443	751	10 / 263	49,72 (16,41)
Lauf W9.9x.4	1080	0,712	0,963 (0,046)	263	426	757	10 / 117	49,73 (15,49)
Lauf W9.9x.5	1080	0,735	0,963 (0,047)	274	453	744	10 / 130	49,57 (15,57)
Lauf W9.9x.6	1080	0,717	0,962 (0,047)	269	428	737	10 / 111	49,51 (15,33)

N = Anzahl von gerechneten Instanzen
Erreichungsgrad = Zielfunktionswert EA ÷ Optimaler Zielfunktionswert
Hit Rate = Anzahl der Instanzen, deren heuristische Lösung einen Erreichungsgrad von mindestens 100%, 99%, 95% aufweist

Auch wenn der Algorithmus im Durchschnitt eine stabile Performance aufweist, werden einzelne Instanzen jedoch unterschiedlich gut gelöst. Gerade in der Unternehmenspraxis sind jedoch nur wenige Instanzen zu berechnen, wodurch die Möglichkeit besteht, diese mit dem Algorithmus mehrmals zu lösen und das beste Ergebnis zu verwenden.

Tabelle 4.14: Performance des Algorithmus zur simultanen Bündelung bei Mehrfachausführung

Maximum der Läufe	N	Erreichungsgrad		Hit Rate		
		min	Durchschnitt (Std.Abw.)	100%	99%	95%
W9.9x.1	1080	0,678	0,962 (0,048)	265	443	749
Best of 2 (W9.9x.1-W9.9x.2)	1080	0,717	0,965 (0,046)	292	450	787
Best of 3 (W9.9x.1-W9.9x.3)	1080	0,743	0,972 (0,040)	351	523	852
Best of 4 (W9.9x.1-W9.9x.4)	1080	0,743	0,975 (0,038)	380	569	883
Best of 5 (W9.9x.1-W9.9x.5)	1080	0,743	0,977 (0,037)	396	599	902
Best of 6 (W9.9x.1-W9.9x.6)	1080	0,743	0,978 (0,036)	405	612	910

N = Anzahl von gerechneten Instanzen
Erreichungsgrad = Zielfunktionswert EA ÷ Optimaler Zielfunktionswert
Hit Rate = Anzahl der Instanzen, deren heuristische Lösung einen Erreichungsgrad von mindestens 100%, 99%, 95% aufweist

Durch dieses Vorgehen lässt sich wie in Tabelle 4.14 dargestellt mit zwei- bis sechsfacher Wiederholung der durchschnittliche Erreichungsgrad um 0,3%- bis 1,6%-Punkte steigern, während gleichzeitig die Standardabweichung um bis zu 1,2%-Punkte sinkt. Gleichzeitig wurden mit sechsfacher Wiederholung 140 Instanzen zusätzlich optimal, 169 Instanzen zusätzlich nahoptimal und 161 Instanzen zusätzlich sehr gut gelöst.

Der Hit Raten Vergleich zwischen Mehrfach- und Einzelausführung zeigt, dass durch Mehrfachausführung sowohl bereits sehr gut gelöster Instanzen nun (nah-) optimal gelöst werden, als auch bisher schlecht gelöste Instanzen nun sehr gut gelöst werden. Es zeigt sich aber auch, dass über die dreimalige Wiederholung hinaus kaum Verbesserungen erzielbar sind, sodass im praktischen Einsatz eine dreimalige Wiederholung ein guter Kompromiss zwischen Laufzeit und Lösungsgüte darstellt. Aus diesem Grund wird diese Einstellung, mit der ein durchschnittlicher Erreichungsgrad von 97,2% erzielt wurde, auch zur Analyse des Einflusses externer Umweltfaktoren verwendet.

4.3 Performance in Abhängigkeit der Umweltfaktoren

Die bisherigen Untersuchungen zeigen, dass der Algorithmus in der Lage ist, gute Lösungen zu ermitteln. Dazu wurden bisher alle Instanzen gemeinsam betrachtet und auf eine Performanceanalyse für einzelne Settings verzichtet. Im Weiteren wird dies nachgeholt und untersucht, ob die Performance des Algorithmus von den Realisationen der in Tabelle 4.1 dargestellten Umweltfaktoren abhängt. So ist bspw. zu klären, ob steigende Produktkomplexität zu Performanceeinbußen führt. Die statistische Untersuchung des Einflusses erfolgt u.a. mit Hilfe der Varianzanalyse.

Tabelle 4.15 sind die Ergebnisse aufgeschlüsselt nach den Umweltfaktoren zu entnehmen. Die Varianzanalyse mit Haupteffekten und Interaktionseffekten 1. Ordnung ergibt, dass alle Umweltfaktoren mit Ausnahme der Anzahl der Produktlinien einen signifikanten Einfluss auf die Lösungsgüte ausüben. Das R^2 beträgt 0,508, sodass 50,8% der Varianz durch die untersuchten Faktoren erklärt werden kann. Dies ist ein akzeptabler Wert, zeigt aber auch, dass ein Teil der Performanceschwankungen nicht durch die Umweltfaktoren zu erklären ist und welcher daher in den konkreten Datenrealisationen begründet liegen muss. Die Anzahl benötigter Generationen wird durch die Umweltfaktoren deutlich besser erklärt, da hier das $R^2 = 0{,}780$ beträgt. Es zeigt sich, dass bei einer Irrtumswahrscheinlichkeit von $\alpha = 0{,}01$ alle Faktoren einen signifikanten Einfluss ausüben. Im Weiteren ist im Detail zu analysieren, welche Realisationen von Umweltfaktoren zu welcher Performance führen, wobei insbesondere die Lösungsgüte untersucht wird.

4.3.1 Einfluss der Linienanzahl

Die Anzahl der Linien übt keinen signifikanten Einfluss auf die Performance des Algorithmus auf, sodass nichts gegen die Annahme identischer durchschnittlicher Erreichungsgrade spricht. Hinsichtlich der Hit Raten zeigt sich, dass bei 2 Linien im Vergleich zu 4 Linien in allen drei Kategorien leicht bessere Hit Raten erreicht wurden. Dies lässt den Schluss zu, dass bei 2 Linien einige Instanzen deutlich schlechter gelöst worden sein müssen, um identische durchschnittliche Erreichungsgrade zu erzielen. Eine genauere Analyse zeigt, dass 9 von den 10 am schlechtesten gelösten Instanzen mit Erreichungsgraden zwischen 74,2% und 81,0% zwei Linien aufweisen, während nur eine dieser Instanzen vier Linien besitzt.

4.3.2 Einfluss der Segmentanzahl

Die Performanceentwicklung in Abhängigkeit von der Anzahl betrachteter Segmente entspricht den Erwartungen aufgrund theoretischer Überlegungen. Werden nur vier Segmente betrachtet, so beträgt der durchschnittliche Erreichungsgrad 99,0%, bei acht Segmenten 97,4% und bei zwölf Segmenten 95,1%. Die Standardabweichung erhöht sich im Gegenzug von 2,3% auf 3,6% bzw. 4,8%. Ein identisches Bild zeigen die Hit Raten. Auch wenn auf den ersten Blick die Performanceveränderung deutlich ausfällt, muss berücksichtigt werden, dass mit Erhöhung der Segmentzahl auch die Anzahl möglicher Produktprogramme stark ansteigt. So existieren bspw. bei drei Merkmalen und zwei Ausprägungen 8 verschiedene Produkte, sodass bei zwei Linien und bei Berücksichtigung der Nicht-Enthalten Alternative 81 Bündel existieren. Bei vier Segmenten sind davon 4, bei acht Segmenten 8 und bei zwölf Segmenten 12 auszuwählen. In Anbetracht aller Kombinationsmöglichkeiten existieren damit bei vier Segmenten ca. 1,7 Mio. bei acht Segmenten ca. 32 Mrd. und bei zwölf Segmenten $7 \bullet 10^{13}$ Produktprogramme (ohne Preise). Die Anzahl reduziert sich jedoch deutlich, wenn die Variantenanzahl pro Produktlinie wie in der Praxis üblich, unabhängig von der Anzahl der Segmente z.B. auf 4 festgelegt wird. Daher ist anzunehmen, dass bei Vermeidung von mit den Segmenten steigende Variantenzahl die Performance des Algorithmus weit weniger als hier dargestellt zurückgeht.

Tabelle 4.15: Performance des Algorithmus zur simultanen Bündelung in Abhängigkeit der Umweltfaktoren

Faktor	Ausprägung	N	Erreichungsgrad			Hit Rate			Generationen (Iterationen)		
			min	Durchschnitt (Std.Abw.)	p-Wert	100%	99%	95%	min/max	Durchschnitt (Std.Abw.)	p-Wert
Anzahl Linien					0,119						**<0,001**
	2	540	0,743	0,973 (0,042)		196	269	442	10 / 263	45,03 (14,83)	
	4	540	0,796	0,971 (0,038)		155	254	410	32 / 113	54,13 (15,24)	
Anzahl Segmente					**<0,001**						**<0,001**
	4	360	0,806	0,990 (0,023)		233	278	335	10 / 76	38,55 (8,38)	
	8	360	0,825	0,974 (0,036)		96	181	292	12 / 110	50,31 (12,74)	
	12	360	0,743	0,951 (0,048)		22	64	225	10 / 263	59,88 (16,72)	
Produktkomplexität					**<0,001**						**<0,001**
	einfach (3 Merk, 2 Ausp)	360	0,743	0,972 (0,048)		139	199	290	10 / 263	37,41 (10,94)	
	mittel (6 Merk, 3 Ausp)	360	0,891	0,987 (0,018)		133	213	335	34 / 108	51,26 (12,16)	
	komplex (9 Merk, 4 Ausp)	360	0,796	0,958 (0,042)		79	111	227	37 / 113	60,08 (14,55)	
Zahlungsbereitschaftstyp					**<0,001**						**0,002**
	Typ I	360	0,743	0,968 (0,048)		144	188	275	10 / 113	49,54 (15,43)	
	Typ II	360	0,783	0,981 (0,028)		133	196	310	11 / 263	50,16 (16,28)	
	Typ III	360	0,749	0,967 (0,040)		74	139	267	10 / 102	49,04 (15,38)	
Gesamt		1080	0,743	0,972 (0,040)		351	523	852	10 /263	49,58 (15,71)	
R^2					0,508						0,780

N = Anzahl von gerechneten Instanzen, Erreichungsgrad = Zielfunktionswert EA ÷ Optimaler Zielfunktionswert

Hit Rate = Anzahl der Instanzen, deren heuristische Lösung einen Erreichungsgrad von mindestens 100%, 99%, 95% aufweist

p-Wert = p-Werte der Haupteffekte ermittelt durch eine Varianzanalyse mit Haupteffekten und Interaktionseffekten 1. Ordnung

4.3.3 Einfluss der Produktkomplexität

Hinsichtlich der Produktkomplexität werden überraschenderweise Instanzen mit Produkten mittlerer Komplexität am besten gelöst werden. Hier wird ein Erreichungsgrad von 98,7% erzielt, während bei einfachen Produkten nur 97,2% und bei komplexen Produkten sogar nur 95,8% erreicht wird. Die Standardabweichung beträgt bei mittleren Produkten 1,8% und zeigt damit den geringeren Schwankungsbereich dieser Realisationen gegenüber einfachen und komplexen Produkten (Standardabweichung 4,8% bzw. 4,2%). Eine identische Reihenfolge ergibt sich bei Betrachtung der minimalen Erreichungsgrade. So werden bei einfachen Produkten minimal 74,3%, bei mittleren Produkten 89,1% und bei komplexen Produkten 79,6% erreicht. Die Hit Raten zeigen, dass mit zunehmender Komplexität wie zu erwarten weniger Instanzen optimal gelöst werden. Die Anzahl nahoptimaler und sehr guter Lösungen nimmt jedoch zunächst zu und nimmt dann wieder ab. Es scheint hier der Fall aufzutreten, dass bei einfachen Produkten einige Instanzen existieren, die vom Algorithmus nur schwierig gelöst werden können. Mit zunehmender Produktkomplexität verringert sich dieses Phänomen und der langfristige, negative Effekt aufgrund größerer Problemkomplexität kommt zum Tragen. Es ist daher mit weiter abnehmenden Erreichungsgraden zu rechnen, sofern die Produktkomplexität weiter erhöht wird.

4.3.4 Einfluss des Zahlungsbereitschaftstyps

Bezüglich der Zahlungsbereitschaft lässt sich feststellen, dass Instanzen mit Zahlungsbereitschaften vom Typ II besser gelöst werden als Instanzen der beiden anderen Typen. So wird bei Typ II im Durchschnitt ein Erreichungsgrad von 98,1% erzielt, während bei Typ I 96,8% und bei Typ III 96,7% erreicht werden. Diese Reihenfolge wird auch durch die Hit Raten bestätigt. Die Performance ist bei Typ II mit einer Standardabweichung von 2,8% sehr stabil, während der Erreichungsgrad bei Typ I und III mit einer Standardabweichung von 4,8% bzw. 4,0% deutlich stärker schwankt. Die Ursache dieser Unterschiede liegt in den Daten und ihrer Generierung begründet. Bei Typ I sind Zahlungsbereitschaften eines Kunden weitestgehend unabhängig von den Zahlungsbereitschaften anderer Kunden sowie den Kosten. Dadurch sind für den Algorithmus aber auch kaum Rückschlüsse möglich, welcher Bereich des Suchraums vorteilhaft ist. Selbst kleinere Änderungen können einen starken Zuwachs an Zahlungsbereitschaft bei einem Kunden und eine enorme Abnahme bei einem anderen bedeuten. Die Fitnesslandschaft, also das Gebirge der Gesamtdeckungsbeiträge in Abhängigkeit des Produktdesigns, wird daher stark zerklüftet sein und viele Sprünge aufweisen und daher schwierig zu optimieren sein.

Bei Zahlungsbereitschaften von Typ III verhält es sich entgegengesetzt. Die Zahlungsbereitschaft der Kunden basiert vornehmlich auf den Kosten, sodass Kunden für Merkmalsausprägungen mit höheren Kosten auch tendenziell höhere Zahlungsbereitschaften aufweisen. Es sollte daher einfacher sein, aussichtsreiche Bereiche des Suchraums zu identifizieren. Dies wird jedoch durch eine andere Tatsache erschwert. Durch die Beziehung der einzelnen Zahlungsbereitschaften zu den Kosten besitzen die Kunden auch untereinander sehr ähnliche Zahlungsbereitschaften. Kannibalisierungseffekte treten daher sehr viel häufiger auf und können weniger gut durch Anpassungen des Produktdesigns korrigiert werden. Hier kommt der Preissetzung eine Schlüsselrolle zu. Durch geeignete Preissetzung sind die Kunden zu identifizieren, durch deren Nichtbedienung die Preise der Produkte erhöht und somit der Gesamtdeckungsbeitrag insgesamt gesteigert werden kann. Die leichte Verschlechterung des Erreichungsgrades lässt darauf schließen, dass dies nicht im ausreichenden Maße erfolgt. Positiv ist jedoch festzustellen, dass der Algorithmus für die als realistisch eingeschätzten Zahlungsbereitschaften den höchsten Erreichungsgrad von 98,1% erzielte. Es bleibt daher zu hoffen, dass in der Realität erhobene Zahlungsbereitschaften ebenfalls nicht zu den Randfällen gemäß Typ I und III gehören und somit die Performance den hier präsentierten Gesamtdurchschnitt von 97,2% ähnlich stark übersteigt.

4.3.5 Analyse der identifizierten Auffälligkeiten

Die Analyse ergab, dass der Umweltfaktor *Produktkomplexität* einen auf Basis theoretischer Überlegungen überraschenden Einfluss auf die Performance des Algorithmus ausübt. Zudem wundert die Tatsache, dass die Instanz mit dem schlechtesten Erreichungsgrad ausgerechnet zwei Linien und einfach strukturierte Produkte aufweist. Darüber hinaus enthält sie zwölf Segmente und die Zahlungsbereitschaften sind gemäß Typ I generiert worden. Die Ursache ist daher durch genaue Analyse der Wirkung von Kombinationen von Umweltfaktoren zu identifizieren. Tabelle 4.16 stellt den durchschnittlichen Erreichungsgrad in Abhängigkeit von der Anzahl betrachteter Linien und der Produktkomplexität dar.

Tabelle 4.16: Durchschnittlicher Erreichungsgrad in Abhängigkeit von Linienanzahl und Produktkomplexität

Komplexität / Linien	einfach	mittel	komplex	Gesamt
2	95,1%	99,2%	97,7%	97,3%
4	99,3%	98,1%	93,8%	97,1%
Gesamt	97,2%	98,7%	95,8%	97,2%

Es zeigt sich, dass die identifizierte Ungereimtheit nur auftritt, sofern zwei Produktlinien betrachtet werden. Bei vier Produktlinien hingegen ergibt sich ein Verlauf gemäß theoretischer Überlegungen. Eine detaillierte Analyse dieses Effekts im 2-Linien-Fall in Abhängigkeit von Segmentanzahl und ZB-Typ ist in Tabelle 4.17 und Tabelle 4.18 dargestellt.

Tabelle 4.17: Durchschnittlicher Erreichungsgrad in Abhängigkeit von Segmentanzahl und Produktkomplexität bei zwei Produktlinien

Zwei Linien

Komplexität / Segmente	einfach	mittel	komplex	Gesamt
4	98,2%	99,8%	99,2%	99,1%
8	95,7%	99,5%	97,9%	97,7%
12	91,3%	98,3%	96,0%	95,2%
Gesamt	95,1%	99,2%	97,7%	97,3%

Überraschenderweise ist Tabelle 4.17 zu entnehmen, dass, obwohl die Erreichungsgrade bei mittelkomplexen Produkten nur leicht in Abhängigkeit der Segmentzahl schwankt, deutliche Unterschiede bei einfach strukturierten Produkten zu erkennen sind. Mit zunehmender Anzahl der betrachteten Segmente nimmt auch der ungewöhnliche Effekt zu.

Tabelle 4.18: Durchschnittlicher Erreichungsgrad in Abhängigkeit von Zahlungsbereitschaftstyp und Produktkomplexität bei zwei Produktlinien

Zwei Linien

Komplexität / ZB-Typ	einfach	mittel	komplex	Gesamt
Typ I	94,8%	99,9%	98,7%	97,8%
Typ II	96,2%	99,2%	98,3%	97,9%
Typ III	94,2%	98,5%	96,1%	96,3%
Gesamt	95,1%	99,2%	97,7%	97,3%

Bei zwei Produktlinien weisen alle Zahlungsbereitschaften den ungewöhnlichen Verlauf hinsichtlich der Produktkomplexität auf. Allerdings existieren leichte Unterschiede. Bei Zahlungsbereitschaften vom Typ II ist der Effekt am geringsten, gefolgt von Typ III und abschließend von Typ I. Es ist daher festzuhalten, dass anscheinend die Kombination von zwei Linien und einfachen Produkten, insbesondere in Kombination mit einer großen Anzahl von Segmenten und eher bei Zahlungsbereitschaften vom Typ I und III für den Algorithmus schwierig zu lösen ist.

Da gerade Kombinationen von Umweltfaktoren problematisch sind, die eher zu einer geringeren Anzahl möglicher Produktprogramme führen, ist der Schluss naheliegend, dass die Ursache weniger im evolutionären Teil als in der Preisheuristik begründet liegt. Dies ist nicht unwahrscheinlich, da im Rahmen der Parametereinstellung die Preisheuristik als die Haupteinflussgröße für die Performance identifiziert wurde und anscheinend unterschiedliche Instanzen bei Verwendung unterschiedlicher Heuristiken unterschiedlich gut gelöst werden. Die Ursache dürfte in der Tatsache begründet liegen, dass die Preisheuristiken unzureichend mit Meta-Bündeln und Surplussuperadditivität umgehen können. Es sei daran erinnert, dass die Preisheuristiken davon ausgehen, dass ein Kunde genau ein Bündel erwirbt. Auf Basis der Preissetzung und der Zahlungsbereitschaften der Kunden muss dies jedoch nicht immer der Fall sein. Zur Wahrung der Lösungsgültigkeit wird im Anschluss an die Preissetzung die Kundenwahl überprüft, ggf. angepasst und der korrekte Gesamtdeckungsbeitrag errechnet. Dies wird in den seltensten Fällen optimal sein. Vielmehr könnte durch eine andere Preissetzung vermutlich bessere Ergebnisse erzielt werden. Meta-Bündel sind immer dann problematisch, wenn sie die Surplussuperadditivität verletzen, was in nahezu allen realistischen Fällen nur dann geben ist, wenn die Meta-Bündel überschneidungsfrei sind. Ein Meta-Bündel, welches aus einer Linie mehrere Varianten enthält, ist hingegen in den meisten Fällen unproblematisch. Das Auftreten von problematischen Meta-

Bündeln ist jedoch stark von der Anzahl der Produktlinien abhängig. Bei vier Linien werden viele Kunden Bündel mit 3 oder 4 Produkten erwerben. Die resultierenden Meta-Bündel werden jedoch selten überschneidungsfrei sein. Anders bei zwei Produktlinien, hier werden einige Kunden auch nur ein Produkt erwerben. Die Kombination der Einzelprodukte stellt überschneidungsfreie Meta-Bündel dar, die somit die Surplussuperadditivität verletzen können. Ob überschneidungsfreie Bündel problematisch im Hinblick auf die Surplussuperadditivität werden, hängt von den Zahlungsbereitschaften der Kunden ab. Existieren nur wenige Merkmale und Ausprägungen, besteht nicht genügend Differenzierungspotenzial um zu vermeiden, dass ein Produkt für einen Kunden nicht auch einem anderen Kunden gefällt. Die Problematik verschärft sich einerseits, wenn aufgrund steigender Segmentzahl mehr Bündel angeboten werden und anderseits, wenn Kunden in ihren Zahlungsbereitschaften sehr ähnlich sind. Dies ist insbesondere bei Zahlungsbereitschaften vom Typ I und Typ III der Fall, da diese im Vergleich zu Typ II ähnlicher sind, wenn auch aus unterschiedlichem Grund. Zahlungsbereitschaften vom Typ III besitzen einen starken Kostenbezug, so dass alle Kunden für teurere Merkmalsausprägungen auch bereit sind mehr zu bezahlen. Bei Zahlungsbereitschaften vom Typ I hingegen weisen alle Kunden ein ähnliches Niveau von Zahlungsbereitschaft für die einzelnen Produkte auf.

Als Fazit kann daher festgehalten werden, dass die ungewöhnlichen Auswirkungen der Umweltfaktoren durch das gehäufte Auftreten von problematischen Meta-Bündeln erklärt werden kann. Es zeigt zudem, dass in erster Näherung mit der Adaption der Dobson Kalish Heuristiken gute Ergebnisse zu erzielen sind, diese jedoch verbessert werden können, wenn die Preisheuristik besser an die Besonderheiten der Preissetzung bei Bündeln angepasst werden.

5 Fazit und Ausblick

Entscheidungsträger im Produkt- und Preismanagement können zur optimalen Gestaltung von Produktprogrammen unter Berücksichtigung von Preisbündelung bisher nicht auf leistungsfähige Algorithmen zurückgreifen. Im Rahmen dieses Arbeitsberichts wird daher ein hybrider evolutionärer Algorithmus vorgestellt und umfassend evaluiert. Diese populationsbasierte Meta-Heuristik wurde aufgrund ihres zahl- und erfolgreichen Einsatzes zur optimalen Gestaltung von Produktlinien ausgewählt. Aufgrund der Komplexität war eine Problemdekomposition in ein *Design-Masterproblem* und ein *Pricing-Subproblem* notwendig. Der evolutionäre Algorithmus wurde daher um bereits erfolgreich eingesetzte Preisheuristiken ergänzt und so zu einem hybriden evolutionären Algorithmus, auch memetischer Algorithmus genannt, erweitert. In jeder Generation wird zu einem vom evolutionären Algorithmus vorgeschlagenen Produktprogramm mittels der Preisheuristik ein passendes Preissystem bestimmt bzw. „erlernt“, bevor der Gesamtdeckungsbeitrag berechnet wird.

Die Besonderheit des hier eingesetzten evolutionären Algorithmus liegt in seiner Kodierung. Anstelle eines einfachen Strings wird ein komplexer String verwendet, der als semantische Hierarchie interpretiert werden kann. Danach stellt ein Individuum ein vollständiges Produktprogramm dar, welches aus mehreren Bündeln besteht. Die Bündel besitzen einen Preis und enthalten von jedem Produkttyp maximal eine Variante. Die Varianten wiederum werden durch Merkmale und Ausprägungen beschrieben und sind ihrerseits Teil einer u.U. beschränkten Produktlinie. Als Rekombinationsoperator wird eine die semantische Hierarchie berücksichtigende Variante des Uniform Crossovers verwendet. Die Mutationsoperatoren *Change-Product-Specification*, *Replace-Product-In-Bundle* sowie *Replace-Product-In-Line* sorgen für die genetische Vielfalt und verhindern eine vorzeitige Konvergenz. Die Fitness eines Individuums ergibt sich gemäß dem *Linear Ranking* in Abhängigkeit des Rangs den das Individuum innerhalb der Population aufgrund des erzielbaren Gesamtdeckungsbeitrags inne hat.

Vor Berechnung des Gesamtdeckungsbeitrags wird mit den von Dobson und Kalish entwickelten Heuristiken *Reassignment* und *PriceGreedy* ein zum Produktprogramm passendes Preissystem bestimmt. Erstere macht sich den Umstand zu Nutze, dass die Bestimmung optimaler Preise bei gegebener Zuordnung von Kunden zu Bündeln dem Finden kürzester Wege innerhalb eines Graphens entspricht. Kürzeste Wege sind jedoch aufgrund der Algorithmen von Dijkstra bzw. Bellman und Ford sehr schnell exakt zu bestimmen. Zur Identifikation einer guten Zuordnung versucht die *Reassign-*

ment-Heuristik, eine gegebene Zuordnung sukzessive durch Neuzuordnung (Reassignment) zu verbessern. Anfänglich können die Kunden dem Bündel zugeordnet werden, für welches sie den größten Reservationspreis besitzen (*MaxR*-Heuristik) oder dem Bündel, bei dem die mögliche Wohlfahrt als Differenz von Reservationspreis und Kosten maximal ist (*MaxW*-Heuristik). Im Gegensatz dazu wird bei der *PriceGreedy* Heuristik in jedem Schritt gierig (greedy) das Bündel mit dem größten Potenzial dem Produktprogramm hinzugefügt. Anschließend wird zunächst der Preis des neu hinzugefügten Bündels festgelegt, bevor der Reihe nach die Preise der bereits im Produktprogramm enthaltenden Bündel ggf. angepasst werden. Die Preissetzung bzw. -anpassung erfolgt ebenfalls gierig im Hinblick auf den erzielbaren Gesamtdeckungsbeitrag. Da beide Heuristiken zur Preisfindung im Rahmen der Produktliniengestaltung entwickelt wurden, können sie nicht die Einhaltung der Surplussuperadditivitätsbedingung sicherstellen. Daher wird die resultierende Zuordnung mit dem *MetaCheck* überprüft. Sollte ein Kunde zu einem aus den Bündeln gebildeten Meta-Bündel wechseln, wird der Gesamtdeckungsbeitrag entsprechend korrigiert.

Vor Evaluation des Algorithmus wurden mit Hilfe von 540 Testinstanzen die Parameter eingestellt. Es zeigt sich, dass für die Qualität der gefundenen Lösung die Preisheuristik der entscheidende Parameter ist. Die besten Ergebnisse wurden mit Verwendung der *MaxW + Reassignment*-Preisheuristik erzielt. Im Gegensatz dazu schneidet die *PriceGreedy* Heuristik und insbesondere die *MaxR + Reassignment*-Heuristik sehr schlecht ab. Die besten Ergebnisse sind mit der *MaxW + Reassignment*-Preisheuristik, einem hohen Selektionsdruck von 1,6 gepaart mit einer niedrigen Mutationsrate von 1,0% / 2,5% / 0,3% für die drei Mutationsoperatoren *Change-Product-Specification*, *Replace-Product-In-Bundle* und *Replace-Product-In-Line* erreicht worden. Die Mixing Rate zur Einstellung des Rekombinationsoperators kann nahezu frei gewählt werden, da signifikante Unterschiede kaum festgestellt werden konnten, obwohl eine Mixing Rate von 25% leicht vorteilhaft ist. Die Parameter Populationsgröße und Anzahl der Nachkommen sind im Gegensatz zu den anderen Parametern nicht auf Basis der Testinstanzen, sondern im Rahmen der experimentellen Evaluation überprüft worden. Dies war notwendig, um dem mit steigenden Werten ansteigenden durchschnittlichen Erreichungsgrad die zu erwartende längere Laufzeit gegenüberzustellen. Hinsichtlich der Population kann die Verwendung von 50 bis 100 Individuen empfohlen werden. Darunter ist mit deutlichen Qualitätseinbußen, darüber nur mit geringfügigen Verbesserungen bei deutlicher Laufzeitverlängerung zu rechnen, sodass für die weiteren Berechnungen der Ausgangswert von 100 Individuen beibehalten wurde. Hinsichtlich der Anzahl der Nachkommen führte die Erzeugung der neunfachen Population zu den besten Ergebnissen und gleichzeitig zu den wenigsten Generationen. Da sich die Laufzeit

gemessen in Sekunden trotz einer größeren Anzahl an Fitnessauswertungen pro Generation nur leicht erhöhte, wird für die weiteren Berechnungen diese Einstellung der Ausgangseinstellung (6-fach) vorgezogen.

Die Auswertung der experimentellen Studie mit 1080 Instanzen zeigt, dass der entwickelte Algorithmus in der Lage ist, simultan Produktdesign, Bündelkonfigurationen und Preise zu optimieren. Im Durchschnitt wird mit der *MaxW*-Preisheuristik und den besten Einstellungen ein Erreichungsgrad von 96,2% erzielt. Auch durch mehrmalige Wiederholung der Berechnung mit identischen Einstellungen verändert sich dieses Ergebnis nicht, was die stabile Performance des Verfahrens belegt. Dies gilt jedoch nur bei aggregierter Betrachtung, wie eine genaue Analyse der in einzelnen Instanzen erzielten Erreichungsgrade zeigt. Die kurze Laufzeit des Verfahrens von einigen Millisekunden bis zu wenigen Minuten pro Instanz erlaubt in der Praxis die mehrmalige Ausführung des Algorithmus und die Verwendung des Laufs mit dem besten Ergebnis. So kann mit identischen Einstellungen und sechsfacher Ausführung der Erreichungsgrad um 1,6% gesteigert werden. Die im Weiteren durchgeführte Analyse der Umweltfaktoren zeigt zudem, dass die Performance u.a. vom Zahlungsbereitschaftstyp abhängt. Bei Zahlungsbereitschaften von Typ II, die aufgrund ihrer Erzeugung am ehesten Zahlungsbereitschaften in der Realität entsprechen, wird bei dreifacher Wiederholung im Durchschnitt 98,1% des Optimums erzielt, wodurch abschließend das Verfahren als praxistauglich angesehen werden kann.

Nicht alle Umweltfaktoren weisen den vermuteten Einfluss auf die Performance auf. Während die Performance mit zunehmender Segmentzahl aufgrund steigender Komplexität wie erwartet leicht zurückgeht, scheint eine geringere Anzahl von Linien sowie eine geringere Produktkomplexität schwieriger lösbar zu sein. Die Ergebnisse weisen auf ein Problem in den Preisheuristiken hin. Bei wenigen Linien treten häufiger überschneidungsfreie Meta-Bündel auf, durch die die Surplussuperadditivitätsbedingung verletzt werden kann. Die Verletzung geschieht umso öfter, je weniger Möglichkeiten für andere Produktdesigns bestehen (bei einfachen Produkten am wenigsten), je mehr Varianten angeboten werden (bei 12 Segmenten am meisten), je mehr Segmente existieren und je ähnlicher sich die Zahlungsbereitschaften sind (tendenziell ähnlicher bei Typ I und Typ III). Zur Behebung dieser Problematik sind die existierenden Preisheuristiken anzupassen oder neue Heuristiken zu entwickeln, die mit dieser Situation besser umgehen können. Es ist anzunehmen, dass dadurch die Erreichungsgrade nochmal gesteigert werden können.

Abschließend ist festzuhalten, dass die Kombination aus evolutionärem Algorithmus und den Dobson-Kalish Preisheuristiken zu einem Algorithmus führt, mit dem das

Problem der simultanen Bündelung stabil, schnell und gut gelöst werden kann. Weiterentwicklungen sollten sich vor allem auf die Preisheuristik und dort speziell auf die Berücksichtigung von Meta-Bündeln konzentrieren. Die Beachtung der Surplussuperadditivitätsbedingung im Rahmen der Preissetzung anstelle einer nachträglichen Korrektur dürfte zu einer weiteren Steigerung der Lösungsgüte führen. Jedoch bereits in der vorliegenden Version erzielt der Algorithmus bei Mehrfachausführung auf realitätsnahen Daten im Durchschnitt 98,1% der optimalen Lösung. Neben der hohen Lösungsgüte erlauben vor allem die kurzen Laufzeiten von unter 4 Minuten den Einsatz in der Unternehmenspraxis. Entscheidungsträgern steht damit eine leistungsfähige Heuristik zur Verfügung, um gute Produktprogramme unter Berücksichtigung von Preisbündelung zu ermitteln.

Literaturverzeichnis

Alba, E. (Hrsg.) (2005): Parallel Metaheuristics: A New Class of Algorithms, Hoboken, NJ 2005.

Almeida, F.; Blesa A., Maria J.; Blum, C., et al. (Hrsg.) (2006): Hybrid Metaheuristics, Proceedings of the 3rd international workshop, HM 2006, Gran Canaria, Spain, October 13 - 14, 2006, Berlin u.a. 2006.

Bäck, T.; Fogel, D. B.; Michalewicz, Z. (2000a): Evolutionary Computation 2: Advanced Algorithms and Operators, Bristol u.a. 2000.

Bäck, T.; Fogel, David B.; Michalewicz, Z. (2000b): Evolutionary Computation 1: Basic Algorithms and Operators, Bristol u.a. 2000.

Bäck, T.; Fogel, David B.; Michalewicz, Zbigniew (Hrsg.) (1997): Handbook of Evolutionary Computation, Bristol u.a. 1997.

Baker, J. E. (1985): Adaptive Selection Methods for Genetic Algorithms, in: Grefenstette, J. J. (Hrsg.): Genetic Algorithms and Their Applications: Proceedings of the 1st International Conference on Genetic Algorithms, ICGA 1985, Pittsburgh, USA, July 24- 26, 1985, Hillsdale, NJ 1985, S. 101–111.

Baker, J. E. (1987): Reducing Bias and Inefficiency in the Selection Algorithm, in: Grefenstette, J. J. (Hrsg.): Genetic Algorithms and Their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms, ICGA 1987, Cambridge, USA, July 28- 31, 1987, Hillsdale, NJ 1987, S. 14–21.

Balakrishnan, P. V.; Jacob, V. S. (1996): Genetic algorithms for product design, in: Management Science, Vol. 42, No. 8, 1996, S. 1105–1117.

Baldwin, J. M. (1896): A New Factor in Evolution, in: The American Naturalist, Vol. 30, No. 354, 1896, S. 441–451.

Bamberg, G.; Baur, F.; Krapp, M. (2008): Statistik, 14., korrigierte Aufl., München 2008.

Barr, R.; Golden, B.; Kelly, J.; Resende, M.; Stewart, W. (1995): Designing and Reporting on Computational Experiments with Heuristic Methods, in: Journal of Heuristics, Vol. 1, No. 1, 1995, S. 9–32.

Bartz-Beielstein, T.; Blesa Aquilera, M. J.; Blum, C., et al. (Hrsg.) (2007): Hybrid Metaheuristics, Proceedings of the 4th international workshop, HM 2007, Dortmund, Germany, October 8 - 9, 2007, Berlin u.a. 2007.

Bellman, R. E. (1958): On a Routing Problem, in: Quarterly Journal of Applied Mathematics, Vol. 16, No. 1, 1958, S. 87–90.

Berens, W. (1992): Beurteilung von Heuristiken, zugl. Habilitationsschrift Universität Münster 1991, Wiesbaden 1992.

Beyer, H.-G. (2001): The Theory of Evolution Strategies, Berlin u.a. 2001.

Beyer, H.-G.; Bruchseifer, E.; Jakob, W.; Pohlheim, H.; Sendhoff, B.; To, T. B. (2001): Evolutionäre Algorithmen: Begriffe und Definitionen, Arbeitsbericht, Reihe CI, Nr. 115/01 des Sonderforschungsbereichs SFB 531, Technische Universität Dortmund, Dortmund, 2001, online verfügbar unter http://www2.staff.fh-vorarlberg.ac.at/~hgb/ea-terminologie/richtlinien.html, abgerufen am 12.06.2009.

Blesa Aquilera, M. J.; Blum, C.; Cotta, C., et al. (Hrsg.) (2008): Hybrid Metaheuristics, Proceedings of the 5th international workshop, HM 2008, Malaga, Spain, October 8-9, 2008, Berlin u.a. 2008.

Blesa Aguilera, M. J.; Blum, C.; Roli, A., et al. (Hrsg.) (2005): Hybrid Metaheuristics, Proceedings of the 2nd international workshop, HM 2005, Barcelona, Spain, August 29-30, 2005, Berlin u.a. 2005.

Blickle, T. (1997): Theory of Evolutionary Algorithms and Application to System Synthesis, zugl. Diss. Eidgenössische Technische Hochschule Zürich 1996, Zürich 1997.

Blum, C.; Roli, A. (2008): Hybrid Metaheuristics: An Introduction, in: Blum, C.; Blesa Aguilera, M. J.; Roli, A., et al. (Hrsg.): Hybrid Metaheuristics: An Emerging Approach to Optimization, Bd. Vol. 114, Berlin u.a. 2008, S. 1–30.

Blum, C.; Roli, A.; Alba, E. (2005): An Introduction to Metaheuristic Techniques, in: Alba, E. (Hrsg.): Parallel Metaheuristics: A New Class of Algorithms, Hoboken, NJ 2005, S. 3–42.

Blum, C.; Roli, A.; Sampels, M. (Hrsg.) (2004): Hybrid Metaheuristics: Online Proceedings of the 1st International Workshop, HM 2004, Valencia, Spain, August 22-23, 2004.

Bollhöfer, M.; Mehrmann, V. (2004): Numerische Mathematik: Eine projektorientierte Einführung für Ingenieure, Mathematiker und Naturwissenschaftler, Wiesbaden 2004.

Bosworth, J.; Foo, N.; Zeigler, B. P. (1972): Comparison of Genetic Algorithms with Conjugate Gradient Methods.

Cotta, C.; Talbi, E.-G.; Alba, E. (2005): Parallel Hybrid Metaheuristics, in: Alba, E. (Hrsg.): Parallel Metaheuristics: A New Class of Algorithms, Hoboken, NJ 2005, S. 347–370.

Crainic, T. G.; Toulouse, M. (2003): Parallel Strategies for Meta-Heuristics, in: Glover, F.; Kochenberger, G. A. (Hrsg.): Handbook of Metaheuristics, Boston, Mass. 2003, S. 475–514.

Culberson, J. C. (1998): On the Futility of Blind Search: An Algorithmic View of "No free Lunch", in: Evolutionary Computation, Vol. 6, No. 2, 1998, S. 109–127.

Darwin, C. (1859): On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life, London 1859.

Davis, L. (1991): Handbook of Genetic Algorithms, New York 1991.

Dawkins, R. (1976): The Selfish Gene, New York 1976.

Dijkstra, E. W. (1959): A Note on Two Problems in Connexion With Graphs, in: Numerische Mathematik, Vol. 1, 1959, S. 267–271.

Dobson, G.; Kalish, S. (1988): Positioning and Pricing a Product Line, in: Marketing Science, Vol. 7, No. 2, 1988, S. 107–125.

Dobson, G.; Kalish, S. (1993): Heuristics for Pricing and Positioning a Product-Line Using Conjoint and Cost Data, in: Management Science, Vol. 39, No. 2, 1993, S. 160–175.

Domschke, W.; Drexl, A. (2005): Einführung in Operations Research, 6., überarb. und erw. Aufl., Berlin u.a. 2005.

Dorigo, M.; Stützle, T. (2004): Ant Colony Optimization, Cambridge, Mass. 2004.

Dréo, J.; Siarry, P.; Petrowski, A.; Taillard, E. (2006): Metaheuristics for Hard Optimization: Methods and Case Studies, Berlin u.a. 2006.

Eberhart, R. C.; Kennedy, J.; Shi, Y. (2001): Swarm Intelligence, San Francisco, u.a. 2001.

Eiben, A. E; Smith, J. E (2003): Introduction to Evolutionary Computing, Berlin u.a. 2003.

Eiben, A. E.; Hinterding, R.; Michalewicz, Z. (1999): Parameter Control in Evolutionary Algorithms, in: IEEE Transactions on Evolutionary Computation, Vol. 3, No. 2, 1999, S. 124-141.

Ernst, H. (2008): Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis, 4., vollständig überarb. Aufl., Wiesbaden 2008.

Fink, A.; Voß, S. (2003): Anwendung von Metaheuristiken zur Lösung betrieblicher Planungsprobleme: Potenziale und Grenzen einer softwaretechnischen Unterstützung, in: Wirtschaftsinformatik, 45. Jg., Nr. 4, 2003, S. 395–407.

Fisher, M. L. (1980): Worst-Case Analysis of Heuristic Algorithms, in: Management Science, Vol. 26, No. 1, 1980, S. 1–17.

Fogel, D. B. (1992): Evolving Artificial Intelligence, Ph.D. thesis, University of California, San Diego, San Diego, CA 1992.

Fogel, L. J.; Owens, A. J.; Walsh, M. J. (1966): Artifical Intelligence Through Simulated Evolution, New York 1966.

Ford, L. R. (1956): Network Flow Theory.

Forrest, J. J. H.; Tomlin, J. A. (2007): Branch and bound, integer, and non-integer programming, in: Annals of Operations Research, Vol. 149, No. 1, 2007, S. 81–87.

Garey, M. R.; Johnson, D. S. (1979): Computers and Intractability: A Guide to the Theory of NP-Completeness, New York, NY 1979.

Garus, S. (2000): Evolutionäre Algorithmen in der simulationsunterstützten Produktionsprozessplanung: Einsatz in der chemischen Industrie, zugl. Diss. Ruhr-Universität Bochum 1999, Wiesbaden 2000.

Gerdes, I.; Klawonn, F.; Kruse, R. (2004): Evolutionäre Algorithmen: Genetische Algorithmen - Strategien und Optimierungsverfahren - Beispielanwendungen, Wiesbaden 2004.

Glover, F.; Kochenberger, G. A. (Hrsg.) (2003): Handbook of Metaheuristics, Boston, Mass. 2003.

Glover, F.; Laguna, M. (1997): Tabu Search, Boston u.a. 1997.

Goldberg, D. E. (1989): Genetic Algorithms in Search, Optimization, and Machine Learning, Boston 1989.

Grosan, C.; Abraham, A. (2007): Hybrid Evolutionary Algorithms: Methodologies, Architectures and Reviews, in: Grosan, C.; Abraham, A.; Ishibuchi, H. (Hrsg.): Hybrid Evolutionary Algorithms, Bd. Vol. 75, Berlin u.a. 2007, S. 1–17.

Großmann, C.; Terno, J. (1997): Numerik der Optimierung, 2., durchges. Aufl., Stuttgart 1997.

Hagelschuer, P. B. (1971): Theorie der linearen Dekomposition, Berlin u.a. 1971.

Hamming, R. W. (1950): Error Detection and Error Correction Codes, in: The Bell System Technical Journal, Vol. 26, No. 2, 1950, S. 147–160.

Hanke-Bourgeois, M. (2009): Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens, 3., aktualisierte Aufl.., Wiesbaden 2009.

Hart, W. E; Krasnogor, N; Smith, J. E.(2005): Memetic Evolutionary Algorithms, in: Hart, W. E. Krasnogor, N; Smith, J.E.(Hrsg.): Recent Advances in Memetic Algorithms, Bd. 166, Berlin u.a. 2005, S. 3–30.

Hart, W. E; Krasnogor, N.; Smith, J. E. (Hrsg). (2004): Special Issue on Memetic Algorithms, in: Evolutionary Computation, Vol. 12, No. 3, 2004.

Hertz, A.; Kobler, D. (2000): A Framework for the Description of Evolutionary Algorithms, in: European Journal of Operational Research, Vol. 126, No. 1, 2000, S. 1–12.

Hillier, F. S.; Lieberman, G. J. (2005): Introduction to Operations Research, 8. Aufl., New York 2005.

Holland, J. H. (1975): Adaption in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, Ann Arbor 1975.

Hooker, J. (1995): Testing Heuristics: We Have It All Wrong, in: Journal of Heuristics, Vol. 1, No. 1, 1995, S. 33–42.

Janning, W.; Knust, E. (2004): Genetik: Allgemeine Genetik, molekulare Genetik, Entwicklungsgenetik, Stuttgart 2004.

Jensen, P. A.; Bard, J. F. (2003): Operations Research: Models and Methods, Hoboken NJ 2003.

Jiao, J. R.; Zhang Y.; Wang Y. (2007): A Heuristic Genetic Algorithm for Product Portfolio Planning, in: Computers & Operations Research, Vol. 34, No. 6, 2007, S. 1777–1799.

Jong, K. A. de (2006): Evolutionary Computation: A Unified Approach, Cambridge, Mass. 2006.

Jong, K. A. de (2007): Parameter Setting in EAs: a 30 Year Perspective, in: Lobo, Fernando G.; Lima, Cláudio F.; Michalewicz, Zbigniew (Hrsg.): Parameter Setting in Evolutionary Algorithms, Bd. 54, Berlin, Heidelberg 2007, S. 1–18.

Jong, K. A. de; Spears, W. M. (1992): A Formal Analysis of the Role of Multi-Point Crossover in Genetic Algorithms, in: Annals of Mathematics and Artificial Intelligence, Vol. 5, No. 1, 1992, S. 1–26.

Jourdan, L.; Basseur, M.; Talbi, E.-G. (2009): Hybridizing Exact Methods and Metaheuristics: A Taxonomy, in: European Journal of Operational Research, Vol. 199, No. 3, 2009, S. 620–629.

Kistner, K.-P. (2003): Optimierungsmethoden: Einführung in die Unternehmensforschung für Wirtschaftswissenschaftler, Heidelberg 2003.

Kistner, K.-P.; Steven, M. (2001): Produktionsplanung, 3., vollst. überarb. Aufl., Heidelberg 2001.

Koza, J. R. (1992): Genetic Programming: On the Programming of Computers by Means of Natural Selection, Cambridge, Mass. 1992.

Koza, J. R. (1994): Genetic Programming II: Automatic Discovery of Reusable Programs, Cambridge, Mass. 1994.

Koza, J. R.; Bennett III, F. H.; Andre, D.; Keane, M. A. (1999): Genetic Programming III: Darwian Invention and Problem Solving, San Francisco, CA 1999.

Krasnogor, N.; Smith, J. (2005): A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues, in: IEEE Transactions on Evolutionary Computation, Vol. 9, No. 5, 2005, S. 474–488.

Krumke, S. O.; Noltemeier, H. (2005): Graphentheoretische Konzepte und Algorithmen, Wiesbaden 2005.

Laarhoven, P. J. M. van; Aarts, E. H. L. (1987): Simulated Annealing: Theory and Applications, Dordrecht 1987.

Lamarck, J.-B. de (1809): Philosophie Zoologique, Paris 1809.

Man, K.-F.; Tang, K.-S.; Kwong, S. (1999): Genetic Algorithms: Concepts and Designs, London u.a. 1999.

Meister, A. (2008): Numerik linearer Gleichungssysteme: Eine Einführung in moderne Verfahren, 3., überarb. Aufl., Wiesbaden 2008.

Michalewicz, Z. (1996): Genetic Algorithms + Data Structures = Evolution Programs, 3. überarb. und erw. Aufl., Berlin u.a. 1996.

Michalewicz, Z.; Fogel, D. B. (2004): How to Solve It: Modern Heuristics, 2., überarb. und erweiterte Aufl., Berlin u.a. 2004.

Moscato, P. (1999): Memetic Algorithms: A Short Introduction, in: Corne, D.; Dorigo, M.; Glover, F. (Hrsg.): New Ideas in Optimization, London u.a. 1999, S. 219–235.

Moscato, P. (2002): Memetic Algorithms' Bibliography, online verfügbar unter http://www.densis.fee.unicamp.br/~moscato/memetic_home.html, zuletzt aktualisiert am 23.1.2002, abgerufen am 18.06.2008.

Moscato, P.; Cotta, C. (2003): A Gentle Introduction to Memetic Algorithms, in: Glover, F.; Kochenberger, G. A. (Hrsg.): Handbook of Metaheuristics, Boston, Mass. 2003, S. 105–144.

Müller-Merbach, H. (1981): Heuristics and Their Design: A Survey, in: European Journal of Operational Research, Vol. 8, No. 1, 1981, S. 1–23.

Neumann, K.; Morlock, M. (2002): Operations Research, 2. Aufl., München 2002.

Nissen, V. (1997): Einführung in evolutionäre Algorithmen: Optimierung nach dem Vorbild der Evolution, Braunschweig 1997.

Ong, Y.-S.; Krasnogor, N.; Ishibuchi, H. (Hrsg). (2007): Special Issue on Memetic Algorithms, in: IEEE Transactions on Systems, Man & Cybernetics: Part B, Vol. 37, No. 1, 2007.

Ong, Y.-S.; Lim, M.-H.; Neri, F.; Ishibuchi H. (Hrsg.) (2009): Special Issue on Emerging Trends in Soft Computing - Memetic Algorithms, in: Soft Computing, Vol. 13, No. 8-9, 2009.

Passarge, E.; Kohlhase, J. (2006): Kapitel 1: Genetik, in: Siegenthaler, W.; Blum, H. E. (Hrsg.): Klinische Pathophysiologie, 9., völlig neu bearbeitete Aufl., Stuttgart 2006, S. 4–25.

Pidd, M. (1999): Just Modeling Through: A Rough Guide to Modeling, in: Interfaces, Vol. 29, No. 2, 1999, S. 118–132.

Pohlheim, H. (2000): Evolutionäre Algorithmen: Verfahren, Operatoren und Hinweise für die Praxis, Berlin u.a. 2000.

Polasek, W. (1997): Schließende Statistik: Einführung in die Schätz- und Testtheorie für Wirtschaftswissenschaftler, Berlin u.a. 1997.

Prins, C.; Sorensen, K.; Sevaux, M.; Marti, R. (2010): Metaheuristics: A Comprehensive Guide to the Design and Implementation of Effective Optimisation Strategies, London u.a. 2010.

Puchinger, J.; Raidl, G. R. (2005): Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification, in: Mira, J.; Álvarez, J. R. (Hrsg.): Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach: Proceedings of the 1st International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2005, Las Palmas, Canary Islands, Spain, June 15-18, 2005, Bd. 3562, Berlin u.a. 2005, S. 41–53.

Raidl, G. R. (2006): A Unified View on Hybrid Metaheuristics, in: Almeida, F.; Blesa Aguilera, M. J.; Blum, C., et al. (Hrsg.): Hybrid Metaheuristics, Proceedings of the 3rd international workshop, HM 2006, Gran Canaria, Spain, October 13 - 14, 2006, Berlin u.a. 2006, S. 1–12.

Rardin, R. L.; Uzsoy, R. (2001): Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial, in: Journal of Heuristics, Vol. 7, No. 3, 2001, S. 261–304.

Ravindran, A.; Phillips, D. T.; Solberg, James J. (1987): Operations Research: Principles and Practice, New York 1987.

Rechenberg, I. (1973): Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Stuttgart 1973.

Reeves, C. R. (1993): Evaluation of Heuristic Performance, in: Reeves, Colin R. (Hrsg.): Modern Heuristic Techniques for Combinatorial Problems, New York u.a. 1993, S. 304–315.

Reeves, C. R. (Hrsg.) (1993): Modern Heuristic Techniques for Combinatorial Problems, New York u.a. 1993.

Reeves, C. R.; Beasley, J. E. (1993): Introduction, in: Reeves, C. R. (Hrsg.): Modern Heuristic Techniques for Combinatorial Problems, New York u.a. 1993, S. 1–19.

Rothlauf, F. (2006): Representations for Genetic and Evolutionary Algorithms, 2. Aufl., Berlin u.a. 2006.

Scholl, A. (2001): Robuste Planung und Optimierung: Grundlagen - Konzepte und Methoden - experimentelle Untersuchungen, zugl. Habilitationsschrift Technische Universität Darmstadt 2000, Heidelberg 2001.

Schwefel, H.-P. (1977): Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie: Mit einer vergleichenden Einführung in die Hill-Climbing- und Zufallsstrategie, Basel, Stuttgart 1977.

Shioda, R.; Tunçel, L.; Myklebust, T. G. J. (2009): Maximum Utility Product Pricing Models and Algorithms Based on Reservation Prices: wird erscheinen, in: Computational Optimization and Applications, 2009.

Silver, E. A. (2004): An Overview of Heuristic Solution Methods, in: Journal of the Operational Research Society, Vol. 55, No. 9, 2004, S. 936–956.

Stauß, B. (2006): Optimale Gestaltung von Auswahlmenüs und deren Verwendung im Variantenmanagement, zugl. Diss. Universität Karlsruhe 2005, Frankfurt am Main u. a. 2006.

Syswerda, G. (1989): Uniform Crossover in Genetic Algorithms, in: Schaffer, James David (Hrsg.): Genetic Algorithms: Proceedings of the 3rd International Conference on Genetic Algorithms, ICGA 1989, Fairfax, USA, June 4-7, 1989, San Francisco, Calif. 1989, S. 2–9.

Taha, H. A. (2006): Operations Research: An Introduction, 8. Aufl., Upper Saddle River, NJ 2006.

Talbi, E.-G. (2002): A Taxonomy of Hybrid Metaheuristics, in: Journal of Heuristics, Vol. 8, No. 5, 2002, S. 541–562.

Toutenburg, H.; Heumann, C. (2008): Induktive Statistik: Eine Einführung mit R und SPSS, 4., überarb. und erw. Aufl., Berlin u.a. 2008.

VDI/VDE-Richtlinie 3550, Blatt 3, (2003): Computational Intelligence: Evolutionäre Algorithmen: Begriffe und Definitionen, auch enthalten in: VDI/VDE Handbuch Mess- und Automatisierungstechnik, Band 3.

Wegener, I. (2003): Komplexitätstheorie: Grenzen der Effizienz von Algorithmen, Berlin u.a. 2003.

Weicker, K. (2007): Evolutionäre Algorithmen, 2., überarb. und erw. Aufl., Wiesbaden 2007.

Werners, B. (2008): Grundlagen des Operations Research: Mit Aufgaben und Lösungen, 2., überarb. Aufl., Berlin u.a. 2008.

Werners, B.; Becker, N.; Pietschmann, U. (2009): Preis- und Produktlinienoptimierung im Maschinen- und Anlagenbau, in: Müller, S.; Schuh, G.; Werners, B. (Hrsg.): Systematische Entwicklung und Preisbildung für Sach- und Dienstleistungsbündel, Frankfurt am Main 2009, S. 58–84.

Whitley, D.; Gordon, V. S.; Mathias, K. (1994): Lamarckian Evolution, The Baldwin Effect, and Function Optimization, in: Davidor, Y.; Schwefel, H.-P.;

Männer, R. (Hrsg.): Parallel Problem Solving from Nature (PPSN III): Proceedings of the 3rd International Conference on Parallel Problem Solving from Nature, Jerusalem, Israel, October 9 - 14, 1994 proceedings, Bd. 866, Berlin u.a. 1994, S. 6–15.

Williams, H. P. (1999): Model Building in Mathematical Programming, 4. Aufl., Chichester u.a. 1999.

Winston, W. L. (2004): Operations Research: Applications and Algorithms, 4. Aufl., Belmont, Calif. 2004.

Wolpert, D. H.; Macready, W. G. (1997): No Free Lunch Theorems for Optimization, in: IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, 1997.

Zimmermann, H.-J. (2008): Operations Research: Methoden und Modelle. Für Wirtschaftsingenieure, Betriebswirte, Informatiker, 2., akt. Aufl., Wiesbaden 2008.

Anhang

Tabelle A.1: Anpassung des Selektionsdrucks nach Wahl der MaxW-Preisheuristik (Iteration 2)

Selektions-druck		N	Gesamtdeckungsbeitrag #Max	Ø (Std.Abw.)	Vorzeichentest (p-Werte)				Generationen (Iterationen) #Min	Ø (Std.Abw.)	Vorzeichentest (p-Werte)				
W8	2,0	540	247	3.391.058 (3.779.452)	X \ Y	W8	W9	W10	W11	350	47 (25,40)	X \ Y	W8	W9	W10	W11
					W8	-	0,997	0,063	**0,002**			W8	-	**<0,001**	**<0,001**	**<0,001**
W9	1,6	540	278	3.386.967 (3.759.688)	W9	**0,009**	-	**<0,001**	**<0,001**	193	49 (26,44)	W9	>0,999	-	**<0,001**	**<0,001**
W10	1,1	540	220	3.376.124 (3.760.321)	W10	0,967	>0,999	-	>0,999	63	55 (43,93)	W10	>0,999	>0,999	-	**<0,001**
W11	1,0	540	219	3.370.500 (3.740.262)	W11	>0,999	>0,999	0,541	-	56	56 (43,23)	W11	>0,999	>0,999	>0,999	-

N = Anzahl von gerechneten Instanzen

#Max / #Min = Anzahl an Instanzen, bei denen mit dieser Einstellung der größte GDB erzielt bzw. die geringste Anzahl an Generationen benötigt wurden

Signifikante Unterschiede sind fett hervorgehoben

Gewählte Parametereinstellung ist grau hinterlegt

Tabelle A.2: Anpassung der Mutationsrate nach Wahl der MaxW-Preisheuristik (Iteration 2)

Mutationsrate		N	Gesamtdeckungsbeitrag #Max	Ø (Std.Abw.)	Vorzeichentest (p-Werte)				Generationen (Iterationen) #Min	Ø (Std.Abw.)	Vorzeichentest (p-Werte)			
W9	1,0/2,5/0,3	540	320	3.386.967 (3.759.688)	X \ Y	W9	W12	W13	202	49 (26,44)	X \ Y	W9	W12	W13
					W9	-	**<0,001**	**0,009**			W9	-	**<0,001**	>0,999
W12	2,0/5,0/0,5	540	233	3.348.331 (3.710.037)	W12	>0,999	-	>0,999	107	78 (109,99)	W12	>0,999	-	>0,999
W13	0,5/1,3/0,1	540	277	3.380.676 (3.762.948)	W13	0,997	**0,002**	-	353	46 (16,15)	W13	**<0,001**	**<0,001**	-

N = Anzahl von gerechneten Instanzen

#Max / #Min = Anzahl an Instanzen, bei denen mit dieser Einstellung der größte GDB erzielt bzw. die geringste Anzahl an Generationen benötigt wurden

Signifikante Unterschiede sind fett hervorgehoben

Gewählte Parametereinstellung ist grau hinterlegt

Tabelle A.3: Anpassung der Mixing Rate nach Wahl der MaxW-Preisheuristik (Iteration 2)

Mixing Rate		N	Gesamtdeckungsbeitrag #Max	Ø (Std.Abw.)	Vorzeichentest (p-Werte)				Generationen (Iterationen) #Min	Ø (Std.Abw.)	Vorzeichentest (p-Werte)			
W9	25%	540	288	3.386.967 (3.759.688)	X \ Y	W9	W14	W15	210	49 (26,44)	X \ Y	W9	W14	W15
					W9	-	0,103	0,158			W9	-	0,593	>0,999
W14	50%	540	268	3.381.291 (3.764.032)	W14	0,942	-	0,674	206	49 (26,56)	W14	0,519	-	>0,999
W15	12,50%	540	268	3.383.827 (3.763.767)	W15	0,905	0,441	-	275	57 (17,75)	W15	**<0,001**	**<0,001**	-

N = Anzahl von gerechneten Instanzen

#Max / #Min = Anzahl an Instanzen, bei denen mit dieser Einstellung der größte GDB erzielt bzw. die geringste Anzahl an Generationen benötigt wurden

Signifikante Unterschiede sind fett hervorgehoben

Gewählte Parametereinstellung ist grau hinterlegt

Tabelle A.4: Gegebenes Design für 3 Bündel

Produkt	Merkmal		Bündel A	Bündel B	Bündel C
Kippsattel-auflieger	M1	Ladenlänge	9400mm	8200mm	8200mm
	M2	Mulde	2Metall-Mulde	Alu-Kasten	Alu-Kasten
	M3	Rückwand	hydraulisch	hydraulisch	hydraulisch
	M4	Achsen	3	2	2
Telematik	M5	Kühlung	○	○	○
	M6	Sicherheit	●	●	●
	M7	Just-In-Time	●	○	●
	M8	Trailerinfo	○	○	○
Service-Paket	M9	Fahrleistung	<100.000km	<100.000km	<150.000km
	M10	Vertragsbindung	60 Monate	36 Monate	48 Monate
	M11	Umfang	Wartung & Reparatur	Wartung	Wartung & Reparatur
	M12	Reifendienst	○	●	○
Finanzierung	M13	Art		Mietkauf	Leasing
	M14	Laufzeit		48 Monate	60 Monate
	M15	Schutzpaket		gr. Paket	Ohne

● enthalten / ○ nicht enthalten

Tabelle A.5: Kosten und Zahlungsbereitschaften für Merkmalsausprägungen des Kippsattelaufliegers

Kippsattelauflieger		Kosten	ZB von Kundensegment			
			S_1	S_2	S_3	S_4
M_1: Ladelänge	A_1: 8200 mm	3.000 €	6.000 €	6.000 €	6.000 €	4.000 €
	A_2: 9400 mm	4.000 €	8.000 €	7.000 €	4.000 €	10.000 €
	A_3: 10450 mm	7.000 €	10.000 €	8.000 €	2.000 €	4.000 €
M_2: Mulde	A_1: Alu-Kastenmulde	2.000 €	4.000 €	3.000 €	5.000 €	3.000 €
	A_2: 2Metall-Mulde	4.000 €	8.000 €	6.000 €	2.000 €	3.000 €
	A_3: Stahl-Rundmulde	2.500 €	4.000 €	3.000 €	5.000 €	3.000 €
M_3: Rückwand	A_1: Pendelklappe	2.000 €	6.000 €	4.000 €	1.000 €	10.000 €
	A_2: Kombitür	4.000 €	8.000 €	6.000 €	8.000 €	1.000 €
	A_3: Hydraulisch	5.000 €	10.000 €	8.000 €	6.000 €	4.000 €
M_4: Achsen	A_1: Zwei Achsen	2.000 €	3.000 €	3.000 €	6.000 €	5.000 €
	A_2: Drei Achsen	3.000	5.000 €	4.000 €	3.000 €	4.000 €

Tabelle A.6: Kosten und Zahlungsbereitschaften für Merkmalsausprägungen der Telematik

Telematik		Kosten	ZB von Kundensegment			
			S_1	S_2	S_3	S_4
M_5: Kühl-überwachung	A_1: enthalten	1.500 €	0 €	0 €	0 €	0 €
	A_2: nicht enthalten	0 €	0 €	0 €	0 €	0 €
M_6: Sicherheits-überwachung	A_1: enthalten	2.500 €	3.000 €	0 €	5.000 €	1.000 €
	A_2: nicht enthalten	0 €	0 €	0 €	0 €	0 €
M_7: Just in Time-Paket	A_1: enthalten	1.500 €	2.000 €	2.000 €	1.000 €	0 €
	A_2: nicht enthalten	0 €	0 €	0 €	0 €	0 €
M_8: Trailer-information	A_1: enthalten	2.000 €	3.000 €	0 €	2.000 €	4.000 €
	A_2: nicht enthalten	0 €	0 €	0 €	0 €	0 €

Tabelle A.7: Kosten und Zahlungsbereitschaften für Merkmalsausprägungen des Service-Pakets

Service-Paket		Kosten	ZB von Kundensegment			
			S_1	S_2	S_3	S_4
M_9: Fahrleistung	A_1: ≤ 100.000 km	400 €	2.000 €	0 €	500 €	1.000 €
	A_2: ≤ 150.000 km	600 €	1.000 €	1.000 €	1.000 €	2.000 €
	A_3: ≤ 200.000 km	1.600 €	500 €	500 €	2.000 €	1.000 €
M_{10}: Vertrags-bindung	A_1: 36 Monate	550 €	500 €	500 €	1.500 €	1.000 €
	A_2: 48 Monate	800 €	750 €	750 €	500 €	1.500 €
	A_3: 60 Monate	1.200 €	1.500 €	500 €	500 €	1.000 €
M_{11}: Umfang	A_1: Wartung	1.000 €	1.000 €	1.500 €	0 €	1.500 €
	A_2: W & Reparatur	2.000 €	2.000 €	2.000 €	1.000 €	3.000 €
	A_3: W, R & Ersatzteile	3.000 €	4.000 €	0 €	2.000 €	4.500 €
M_{12}: Reifendienst	A_1: Enthalten	1.100 €	2.000 €	500 €	0 €	1.000 €
	A_2: Nicht enthalten	0 €	0 €	0 €	0 €	0 €

Tabelle A.8: Kosten und Zahlungsbereitschaften für Merkmalsausprägungen der Finanzierung

Finanzierung		Kosten	ZB von Kundensegment			
			S_1	S_2	S_3	S_4
M_{13}: Art	A_1: Mietkauf	750 €	0 €	500 €	1.500 €	1.000 €
	A_2: Leasing	750 €	0 €	1.500 €	1.000 €	1.000 €
	A_3: Kredit	750 €	0 €	0 €	1.000 €	1.000 €
M_{14}: Laufzeit	A_1: 48 Monate	400 €	0 €	500 €	2.000 €	500 €
	A_2: 60 Monate	500 €	0 €	750 €	1.000 €	1.500 €
	A_3: 72 Monate	600 €	0 €	1.500 €	1.000 €	1.000 €
M_{15}: Schutzpaket	A_1: ohne	0 €	0 €	0 €	0 €	0 €
	A_2: kl. Schutzpaket	1.800 €	2.000 €	1.000 €	1.000 €	500 €
	A_3: gr. Schutzpaket	3.000 €	4.000 €	1.000 €	2.000 €	4.000 €