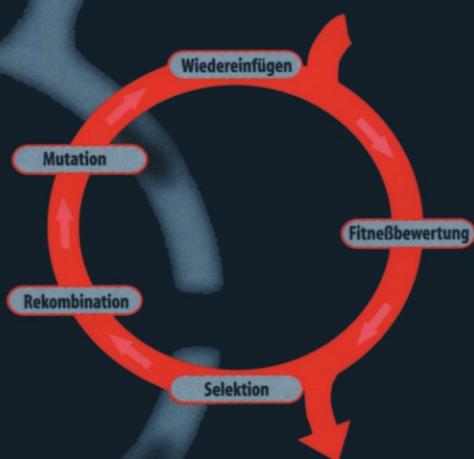


Hartmut Pohlheim

# Evolutionäre Algorithmen

Verfahren, Operatoren und Hinweise  
für die Praxis



Springer



VDE

# Evolutionäre Algorithmen

Springer-Verlag Berlin Heidelberg GmbH

Hartmut Pohlheim

# Evolutionäre Algorithmen

Verfahren, Operatoren und Hinweise für die Praxis

Mit 105 Abbildungen



Springer

**Dr.-Ing. Hartmut Pohlheim**  
**Brodberg 38**  
**D - 14532 Kleinmachnow**

*e-mail:[hartmut@pohlheim.com](mailto:hartmut@pohlheim.com)*  
*<http://www.pohlheim.com>*  
*<http://www.geatbx.com>*

Die Deutsche Bibliothek - CIP- Einheitsaufnahme  
Evolutionäre Algorithmen [Medienkombination] : Verfahren, Operatoren und Hinweise  
für die Praxis / Hartmut Pohlheim. - Berlin ; Heidelberg ; New York ; Barcelona ; Hongkong ; London ;  
Mailand ; Paris ; Singapur ; Tokio : Springer 2000  
(VDI-Buch)

ISBN 978-3-642-63052-1  
DOI 10.1007/978-3-642-57137-4

ISBN 978-3-642-57137-4 (eBook)

Buch. 1999      Gb.  
CD-ROM. Matlab-Toolbox. - 1999

**Additional material to this book can be downloaded from <http://extras.springer.com>.**

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zu widerhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

© Springer-Verlag Berlin Heidelberg 2000  
Softcover reprint of the hardcover 1st edition 2000

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, daß solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Sollte in diesem Werk direkt oder indirekt auf Gesetze, Vorschriften oder Richtlinien (z.B. DIN, VDI, VDE) Bezug genommen oder aus ihnen zitiert worden sein, so kann der Verlag keine Gewähr für die Richtigkeit, Vollständigkeit oder Aktualität übernehmen. Es empfiehlt sich, gegebenenfalls für die eigenen Arbeiten die vollständigen Vorschriften oder Richtlinien in der jeweils gültigen Fassung hinzuzuziehen.

Satz: computer to plate von Autorendaten

Herstellung: ProduServ GmbH Verlagsservice, Berlin

Umschlaggestaltung: Struve&Partner, Heidelberg

Gedruckt auf säurefreiem Papier

SPIN:10697809

68/3020SC - 5 4 3 2 1 0

# Vorwort

In den letzten Jahren kommen Evolutionäre Algorithmen zunehmend als Optimierungsverfahren bei industriellen Lösungen zur Anwendung. Dieser Einsatz wird sich auf Grund der vielfältigen Anwendungsmöglichkeiten in den kommenden Jahren weiter verstärken.

Mit diesem Buch wird – unter Nutzung der bekanntgewordenen Literatur und der Ergebnisse aus eigenen Praxislösungen – der Versuch unternommen, einen umfassenden Überblick über Evolutionäre Algorithmen zu geben. Dies geschieht in einer einfachen und verständlichen Form und ist immer an der Praxis orientiert.

Ziel ist dabei, durch die ausführliche und praxisverbundene Darstellung das Gebiet der Evolutionären Algorithmen für die ingenieurtechnische Anwendung aufzuarbeiten.

Dieses Buch ist für Ingenieure geschrieben, die Evolutionäre Algorithmen zur Lösung technischer Aufgaben einsetzen möchten. Außerdem profitieren von diesem Buch diejenigen, welche sich einen schnellen Überblick über Funktion und Einsatzmöglichkeiten Evolutionärer Algorithmen machen wollen.

Der Leser erhält alle Informationen, die für einen erfolgreichen Einsatz Evolutionärer Algorithmen notwendig sind. Dies beginnt mit einer Einführung in die Struktur Evolutionärer Algorithmen. Darauf aufbauend werden alle grundlegenden Bestandteile, Verfahren und Operatoren beschrieben. Eine Reihe von Erweiterungen Evolutionärer Algorithmen erleichtert die Lösung großer und komplexer Probleme. Besonderes Augenmerk wird auf Hinweise und Empfehlungen für die praktische Anwendung gelegt.

Den Abschluß bildet die ausführliche Darstellung mehrerer ausgewählter Beispiele aus der Praxis. Durch die detaillierte Beschreibung aller Schritte und die Erläuterung möglicher Schwierigkeiten sowie deren Lösung wird ein schneller Start bei der Lösung eigener Probleme ermöglicht.

Ein ausführliches Literaturverzeichnis, Glossar und Sachverzeichnis runden das Buch ab und bieten einen schnellen Zugriff auf die enthaltenen Informationen.

Dem Buch liegt eine CD mit der GENETIC AND EVOLUTIONARY ALGORITHM TOOLBOX für MATLAB (GEATbx) bei. Viele der im Buch beschriebenen Verfahren, Operatoren und Beispiele sind in der Toolbox enthalten. Damit erhält der Leser ein fortgeschrittenes Werkzeug für den Einsatz Evolutionärer Algorithmen, das sofort für die Lösung eigener Praxisprobleme eingesetzt werden kann. Die Toolbox hat sich in den letzten Jahren bei über hundert Anwendern weltweit bewährt.

Mit dem vorliegenden Buch und der Toolbox als einsatzfähiges Werkzeug liegt den in der Praxis stehenden Fachkollegen ein umfassendes Nachschlagewerk vor. Es gibt Hinweise und Hilfen für zu lösende Aufgaben.

Auf der Webseite zum Buch – <http://www.pohlheim.com/eavoh/> – sind weitergehende und neue Informationen zu finden. Unter <http://www.geatbx.com/> können Informationen zu Erweiterungen der Toolbox abgerufen werden.

Möge dieses Buch sein Ziel erfüllen und den Benutzern ein hilfreicher Partner beim Einsatz Evolutionärer Algorithmen als Optimierungsverfahren sein.

Kleinmachnow, August 1999

Hartmut Pohlheim

# Inhaltsverzeichnis

<b>Abkürzungsverzeichnis .....</b>	<b>XIII</b>
<b>1 Einleitung.....</b>	<b>1</b>
<b>2 Struktur und Aufbau Evolutionärer Algorithmen.....</b>	<b>7</b>
<b>3 Grundlegende Verfahren und Operatoren .....</b>	<b>13</b>
3.1 Fitneßzuweisung .....	16
3.1.1 Proportionale Fitneßzuweisung .....	17
3.1.2 Reihenfolgebasierte Fitneßzuweisung (Ranking).....	18
3.1.3 Mehrkriterielle Fitneßzuweisung.....	20
3.2 Selektion.....	24
3.2.1 Roulette selektion .....	25
3.2.2 Stochastic universal sampling .....	25
3.2.3 Turnierselektion.....	26
3.2.4 Truncation-Selektion .....	27
3.2.5 Analyse der Selektionsverfahren.....	28
3.2.6 Vergleich der Selektionsverfahren.....	31
3.3 Rekombination.....	34
3.3.1 Diskrete Rekombination .....	35
3.3.2 Rekombination reeller und ganzzahliger Variablen.....	35
3.3.3 Rekombination binärer Variablen (crossover).....	39
3.3.4 Rekombination bei kombinatorischen Problemen .....	42
3.4 Mutation .....	46
3.4.1 Mutation reeller und ganzzahliger Variablen .....	47
3.4.2 Mutation reeller Variablen mit Adaption der Schrittweiten .....	51
3.4.3 Mutation binärer Variablen .....	53
3.4.4 Mutation bei kombinatorischen Problemen .....	54
3.5 Wiedereinfügen (Reinsertion).....	57
3.6 Initialisierung der Individuen .....	60
3.6.1 Zufällige Initialisierung .....	61

3.6.2	Erweiterte Initialisierung (nicht-zufällig) .....	62
3.7	Abbruchkriterien.....	63
3.7.1	Direkte Abbruchkriterien.....	64
3.7.2	Abgeleitete Abbruchkriterien .....	65
3.7.3	Einsatz der Abbruchkriterien.....	70
3.8	Zusammenfassung.....	71
<b>4</b>	<b>Populationen, verschiedene Strategien und Konkurrenz .....</b>	<b>73</b>
4.1	Klassifikation von Populationsmodellen .....	74
4.2	Globales Modell .....	75
4.3	Lokales Modell .....	77
4.3.1	Topologie von Nachbarschaften .....	78
4.3.2	Allgemeine Beschreibung von Nachbarschaften .....	79
4.3.3	Selektion in Nachbarschaft.....	83
4.3.4	Wiedereinfügen in Nachbarschaft .....	83
4.3.5	Analyse des lokalen Modells.....	84
4.4	Regionales Modell.....	87
4.4.1	Anzahl und Größe der Unterpopulationen.....	89
4.4.2	Migrationstopologien .....	90
4.4.3	Migrationsintervall und Migrationsrate.....	92
4.4.4	Auswahl der Migranten.....	94
4.4.5	Ablauf der Migration.....	94
4.4.6	Analyse des regionalen Modells.....	95
4.5	Anwendung verschiedener Strategien .....	96
4.5.1	Berechnung der Ordnung der Unterpopulationen.....	97
4.5.2	Beispiel der Anwendung verschiedener Strategien .....	99
4.5.3	Analyse der Anwendung verschiedener Strategien .....	101
4.6	Konkurrierende Unterpopulationen.....	102
4.6.1	Erfolg der Unterpopulationen .....	103
4.6.2	Aufteilung der Ressourcen .....	103
4.6.3	Umsetzung der Verteilung der Ressourcen .....	105
4.6.4	Beispiel des Einsatzes konkurrierender Unterpopulationen .....	108
4.6.5	Analyse der konkurrierenden Unterpopulationen .....	111
4.7	Zusammenfassung.....	113
<b>5</b>	<b>Visualisierung und Optimierung .....</b>	<b>115</b>
5.1	Systematisierung der Visualisierung von EA .....	116
5.1.1	Kriterien der Systematisierung .....	116
5.1.2	Daten zur Visualisierung.....	117
5.1.3	Schema der Systematisierung .....	118

5.2 Globale Eigenschaften einer Population .....	119
5.2.1 Zielfunktionswert des besten Individuums der Population (Konvergenzdiagramm) .....	119
5.2.2 Variablen des jeweils besten Individuums einer Population .....	121
5.2.3 Zielfunktionswerte (Güte) aller Individuen über mehrere Generationen .....	123
5.2.4 Position und Größe der Unterpopulationen .....	127
5.3 Lokale Eigenschaften einer Population .....	128
5.3.1 Variablen aller Individuen einer Generation .....	128
5.3.2 Zielfunktionswerte aller Individuen einer Generation .....	131
5.3.3 Distanzverteilung und Distanzkarten der Individuen einer Generation .....	134
5.4 Hochdimensionale Visualisierung .....	138
5.4.1 Verfahren .....	138
5.4.2 Anwendung der mehrdimensionalen Visualisierung .....	140
5.4.3 Analyse der mehrdimensionalen Visualisierung .....	143
5.5 Eigenschaften der Zielfunktion .....	144
5.5.1 Direkte Darstellung der Zielfunktion .....	145
5.5.2 Fitneß-Distanz-Verteilung .....	148
5.5.3 Visualisierung problemspezifischer Daten und Ergebnisse .....	150
5.6 Protokollierung von Daten und Ergebnissen .....	152
5.7 Zusammenfassung und Ausblick .....	153
 <b>6 Genetic and Evolutionary Algorithm Toolbox for Matlab .....</b>	<b>157</b>
6.1 Aufbau und Struktur der GEATbx .....	158
6.2 Anwenderschnittstelle – Scriptfunktionen .....	161
6.3 Vordefinierte Algorithmen – Toolboxfunktionen .....	161
6.4 Zentralfunktion .....	162
6.4.1 Initialisierung .....	163
6.4.2 Fitneßzuweisung und Selektion .....	163
6.4.3 Rekombination .....	164
6.4.4 Mutation .....	165
6.4.5 Wiedereinfügen .....	165
6.4.6 Migration und Konkurrenz zwischen Unterpopulationen .....	166
6.4.7 Terminierung .....	166
6.4.8 Visualisierung .....	166
6.4.9 Protokollierung .....	167
6.5 Zielfunktionen und Beispiele .....	168
6.6 Einbeziehung problemspezifischen Wissens .....	169
6.7 Dokumentation .....	169

6.8 Zusammenfassung und Ausblick.....	170
<b>7 Kombination von Operatoren zu Evolutionären Algorithmen.....</b>	<b>171</b>
7.1 Allgemein einstellbare Verfahren und Operatoren.....	172
7.1.1 Fitneßzuweisung und Selektion.....	172
7.1.2 Anwendung verschiedener Strategien und Konkurrenz zwischen Unterpopulationen .....	173
7.1.3 Regionales Populationsmodell (Migration zwischen Unterpopulationen).....	174
7.1.4 Zusammenfassung allgemein einstellbare Verfahren.....	174
7.2 Global orientierte Parameteroptimierung.....	175
7.3 Lokal orientierte Parameteroptimierung.....	176
7.4 Parameteroptimierung binärer Variablen.....	178
7.5 Kombinatorische Optimierung.....	179
7.6 Parameteroptimierung von Variablen verschiedener Repräsentation .....	181
7.7 Zusammenfassung.....	184
<b>8 Anwendung Evolutionärer Algorithmen auf Praxisprobleme .....</b>	<b>185</b>
8.1 Vorgehen bei der Lösung von Optimierungsaufgaben .....	186
8.1.1 Einordnung des Problems und Festlegung der Zielfunktion.....	187
8.1.2 Untersuchungen des Systemverhaltens.....	188
8.1.3 Festlegung des Optimierungsverfahrens .....	191
8.1.4 Durchführung und Auswertung von Optimierungen.....	192
8.2 Optimierung mehrdimensionaler Funktionen .....	194
8.2.1 Verwendete Optimierungsverfahren.....	195
8.2.2 RASTRIGIN's Funktion.....	196
8.2.3 ROSENBROCK's Funktion.....	198
8.2.4 <i>Moved axis parallel hyper-ellipsoid</i> Funktion.....	201
8.2.5 Zusammenfassung mehrdimensionale Funktionen .....	204
8.3 Parameteridentifikation eines Dieselmotormodells .....	204
8.3.1 Beschreibung des Systems .....	205
8.3.2 Untersuchungen des Systemverhaltens.....	207
8.3.3 Zielfunktion für die Optimierung .....	211
8.3.4 Verwendetes Optimierungsverfahren .....	213
8.3.5 Problem spezifische Visualisierung.....	214
8.3.6 Ergebnisse der Optimierungen .....	215
8.3.7 Auswertung der Ergebnisse.....	222
8.3.8 Zusammenfassung Parameteridentifikation Dieselmotor .....	224
8.4 Optimierung der Parameter eines Reglers (Fahrzeuglenkung) .....	226

---

8.4.1 Modell der Querlenkung.....	226
8.4.2 Anforderungen an den Regler.....	227
8.4.3 Simulation des Gesamtsystems .....	228
8.4.4 Aufstellung der Zielfunktion .....	229
8.4.5 Durchführung der Optimierung .....	232
8.4.6 Ergebnisse der Optimierung .....	234
8.4.7 Zusammenfassung Fahrzeuglenkung .....	238
<b>8.5 Steuerung eines komplexen Systems (Gewächshausklima).....</b>	<b>239</b>
8.5.1 Einleitung .....	239
8.5.2 Kurzbeschreibung des Gewächshausklimamodells .....	240
8.5.3 Repräsentation der Individuen und Zielfunktion .....	242
8.5.4 Einbeziehung problemspezifischen Wissens .....	243
8.5.5 Optimale Steuerung für durchschnittliche Tage .....	244
8.5.6 Optimale Steuerung bei veränderten Preisen .....	246
8.5.7 Optimale Steuerung unter Verwendung realer Wetterdaten .....	247
8.5.8 Zusammenfassung Gewächshausklima.....	252
<b>8.6 Zusammenfassung.....</b>	<b>253</b>
<b>9 Schlußbetrachtungen .....</b>	<b>257</b>
9.1 Zusammenfassung.....	257
9.2 Ausblick.....	261
<b>Anhang.....</b>	<b>267</b>
A.1 Historische Entwicklung Evolutionärer Algorithmen.....	267
A.1.1 Erste Arbeiten zu Evolutionären Algorithmen.....	267
A.1.2 Evolutionäre Programmierung.....	271
A.1.3 Evolutionsstrategien .....	272
A.1.4 Genetische Algorithmen .....	273
A.1.5 Evolutionäre Algorithmen heute.....	275
A.2 Gewächshaus- und Pflanzenmodell.....	276
A.2.1 Zustandsgleichungen des Gewächshauses .....	276
A.2.2 Zustandsgleichungen des Pflanzenmodells (Paprika).....	278
A.2.3 Biomasse und Gewinn .....	279
A.2.4 Beschränkungen .....	280
<b>Literaturverzeichnis .....</b>	<b>281</b>
Evolutionäre Algorithmen und Optimierung .....	281
Kombinatorische Optimierung (TSP, Scheduling) .....	291
Behandlung von Populationen – Parallele Modelle.....	292
Visualisierung .....	294

Polyplloidie bei Evolutionären Algorithmen .....	295
Biologie, Genetik und Populationsgenetik .....	295
Pflanzenwachstum und Gewächshaus.....	296
 <b>Glossar .....</b>	 <b>297</b>
 <b>Sachverzeichnis.....</b>	 <b>307</b>

# Abkürzungsverzeichnis

Abb.	Abbildung
Abschn.	Abschnitt
<i>AnzIndUnterPop</i>	Anzahl der Individuen pro Unterpopulation
<i>AnzKnoten</i>	Anzahl der Knoten
<i>AnzUnterPop</i>	Anzahl der Unterpopulationen
biol.	biologisch
<i>Dim</i>	Dimension
<i>Dist</i>	Distanz (Abstand)
EA	Evolutionäre Algorithmen (engl.: <i>Evolutionary Algorithms</i> )
engl.	englisch
EP	Evolutionäre Programmierung (engl.: <i>Evolutionary Programming</i> )
ES	Evolutionsstrategien (engl.: <i>Evolution Strategies</i> )
<i>Fit</i>	Fitneß
GA	Genetische Algorithmen (engl.: <i>Genetic Algorithms</i> )
GEA/GEATbx	<i>Genetic and Evolutionary Algorithms</i> (benutzt für <i>GEA Toolbox for Matlab</i> )
<i>Gen</i>	Generation, bezeichnet Index einer Generation innerhalb eines Optimierungslaufes (in Gleichungen verwendet)
Gl.	Gleichung
GP	Genetische Programmierung (engl.: <i>Genetic Programming</i> )
gr.	griechisch
HTML	<i>hypertext markup language</i> (Formatanweisungen für Text im World Wide Web)
<i>Ind</i>	Individuum, bezeichnet Index eines Individuums (in Gleichungen verwendet)

<i>IxDimKnoten</i>	absoluter Index eines Knotens, mehrdimensional (Beschreibung von Nachbarschaften)
<i>IxKnoten</i>	absoluter Index eines Knotens, eindimensional
<i>IxRelDimKnoten</i>	relativer Index eines Knotens, mehrdimensional
JSS	<i>job shop scheduling</i>
lat.	lateinisch
<i>LenDim</i>	Länge pro Dimension (engl.: <i>length of dimension</i> ), Anzahl der Knoten pro Dimension
LinRank	lineares Ranking
LossDiv	engl.: <i>loss of diversity</i> (Verlust an Vielfalt)
min	Minimum
<i>n / Nvar</i>	Anzahl der Variablen eines Individuums bzw. eines Problems
<i>Nind</i>	Anzahl der Individuen der Population, der Unterpopulation oder des Selektionspools (engl.: <i>number of individuals</i> )
<i>Pos</i>	Position oder Rang eines Individuums oder einer Unterpopulation (in einer nach einem Kriterium sortierten Liste bzw. Rangfolge)
<i>RadiusAnord</i>	Radius einer Anordnung
<i>RadiusNach</i>	Radius einer Nachbarschaft
SelInt	Selektionsintensität (engl.: <i>selection intensity</i> )
SelVar	Selektionsvarianz (engl.: <i>selection variance</i> )
<i>SP</i>	engl.: <i>selective pressure</i> (Selektionsdruck)
SubPop	engl.: <i>subpopulation</i> (Unterpopulation)
<i>Tour</i>	engl.: <i>tournament size</i> (Turniergröße)
<i>Trunc</i>	engl.: <i>truncation threshold</i> (Abschneideschwelle) oder Truncation-Selektion
TSP	<i>travelling sales person</i> (Handelsreisender)
Turnier	Turnierselektion
Var	Variable (eines Individuums)
<i>VP</i>	Verteilungsdruck (in Analogie zu Selektionsdruck <i>SP</i> )
ZFW	Zielfunktionswert (engl.: <i>objective value</i> )

# 1 Einleitung

Überall in unserer Umgebung und bei der Arbeit begegnen wir Problemen bzw. Systemen, die von einer Anzahl von Entscheidungen bzw. variablen Größen abhängen. Diese Entscheidungen oder Variablen können mehrere Zustände annehmen bzw. eine Variable kann in einem Bereich jeden Wert annehmen. Je nach Wahl und Einstellung der Variablen zeigt das System ein unterschiedliches Verhalten. Der Nutzer oder Betreiber des Systems möchte, daß das System ein bestimmtes Verhalten zeigt. Dazu muß er diejenige Einstellung der Variablen suchen, die dieses oder ein naheliegendes Verhalten hervorruft. Eine weitere Möglichkeit ist, daß nach dem besten Verhalten des Systems gesucht werden soll, wobei die inhaltliche Fragestellung – was ist das beste Verhalten dieses Systems – sehr unterschiedlich aussehen kann. Das System soll besonders gut an eine bestimmte Aufgabenstellung angepaßt sein. Stets muß nach einer Einstellung bzw. Kombination der Variablen gesucht werden, die das gewünschte Verhalten erbringen. Dieser Prozeß der Suche nach der gewünschten Einstellung der Variablen wird als Optimierung bezeichnet.

Das Verhalten eines Systems wird durch die Zielfunktion beschrieben. Für jede Einstellung der Variablen läßt sich durch die Zielfunktion ein Zielfunktionswert, oft auch Gütwert genannt, berechnen. Ist die Zielfunktion so definiert, daß ein kleinerer Zielfunktionswert besser ist, dann wird von einer Minimierung gesprochen. Das bedeutet, daß die Variablen gesucht werden, die den minimalen Zielfunktionswert ergeben. Die umgekehrte Suche entspricht einer Maximierung. Jede zu maximierende Zielfunktion läßt sich durch Multiplikation mit -1 in eine zu minimierende Zielfunktion umwandeln. Deshalb wird ohne Verlust an Allgemeingültigkeit im folgenden immer von einer Minimierung ausgegangen.

Von den Variablen wird durch den für jede Variable vorgegebenen Definitionsbereich ein Suchraum aufgespannt, der von den Optimierungsverfahren auf die unterschiedlichste Art und Weise durchmustert wird. An einem Ende der Skala steht die zufällige Suche (globale Suche), bei der zufällige Kombinationen von Variablenwerten probiert werden und die für jedes Problem anwendbar ist. Auf der anderen Seite stehen mathematische Verfahren, die über die Auswertung mathematischer Eigenschaften (Ableitungen usw.) günstige Variablenwerte ergeben, dafür aber spezielle Eigenschaften der Zielfunktion erfordern und dadurch in ihrer Anwendbarkeit eingeschränkt sind (lokale Suche). Für alle Verfahren stellen sich die Fragen:

- Wo im Suchraum wird mit der Suche begonnen (Initialisierung)?
- Wird mit mehreren Punkten gleichzeitig gearbeitet?
- Wie wird aus den bisherigen Punkten der folgende Punkt berechnet?
- Wann ist die Suche beendet (Abbruchkriterium)?

Je nachdem, wie jede dieser Fragen beantwortet wird, können die verschiedensten Verfahren unterschieden werden. Über viele Jahre hinweg wurde eine riesige Anzahl von Verfahren zur Optimierung entwickelt.

Bisher wurde die Optimierung als ein technisches Verfahren betrachtet. Bei einem Blick in unsere Umgebung ist sehr schnell festzustellen, daß wir von einer Vielzahl lebender Systeme umgeben sind. Eine genauere Betrachtung zeigt, wie gut die vielen unterschiedlichen Lebewesen an ihren jeweiligen Lebensraum angepaßt sind. Jedes Individuum stellt eine Einstellung von Variablen dar, die Zielfunktion ist der Lebensraum eines Individuums und der Zielfunktionswert die Überlebenschance eines Individuums. Je besser ein Individuum an seine Umgebung angepaßt ist, um so bessere Überlebens- und Fortpflanzungschancen hat es. Dabei sind der Lebensraum und die Überlebenschance bei natürlichen Individuen sehr komplexe Größen. Diese Anpassung ist ein Optimierungsprozeß, der seit der Entstehung des Lebens abläuft und immer weiter geht – die Evolution.

Von CHARLES DARWIN wurde 1848 die Bedeutung der Evolution in seinem Buch „*On the origin of species by natural selection*“ [Dar1859] zum ersten Mal dargestellt. Darwin zeigte auch, welche Mechanismen hinter der Evolution stecken. Dies sind zum ersten die **Selektion** und zum zweiten die **Variation**. Die Prozesse der Selektion (Auswahl) entscheiden unter anderem, welche Individuen Nachkommen erzeugen oder welche Individuen überleben. Durch die Variation wird bestimmt, wie die Eltern Nachkommen produzieren.

Ist das alles? Im Prinzip ja! Die Evolution ist ein Prozeß, der auf sehr einfachen Prinzipien basiert. Trotzdem (oder vielleicht gerade deshalb) ist die Evolution ein sehr leistungsfähiges Verfahren, wie ein genauerer Blick in unsere Umgebung deutlich zeigt.

Es stellt sich die Frage, inwieweit es möglich ist, die in der Natur vorhandenen Prinzipien und Mechanismen auf technische Systeme zu übertragen. Können die Vorgänge der natürlichen Evolution zur Adaption bzw. Optimierung technischer Systeme verwendet werden? Die Erkenntnisse aus dem Gebiet der biologischen Evolution, der wissenschaftlichen Genetik und der Populationsgenetik bieten dazu eine große Fülle an Informationen. Die Zielsetzung liegt darin, aus dieser Menge an komplexen Erkenntnissen die Prinzipien und Methoden zu extrahieren, welche die grundlegende Struktur des Evolutionsprozesses charakterisieren. Darauf aufbauend können nachfolgend weitere Teilebereiche betrachtet werden. Ziel kann dabei aber nicht ein reines Kopieren der natürlichen Prozesse und Vorgänge sein. Vielmehr sollen die erkannten Prinzipien auf technische Systeme übertragen werden. Dies kann zu einer starken Analogie zu natürlichen Prozessen führen, aber auch zu einer hohen Abstraktion von diesen. Mit dieser Übertragung von Prinzipien der natürlichen Evolution auf technische Systeme in den verschiedensten Abstufungen beschäftigt sich das Gebiet der Evolutionären Algorithmen.

In Kap. 2 werden die **Struktur und der Aufbau Evolutionärer Algorithmen** übersichtsartig vorgestellt. Die Grundprinzipien ihrer Funktionsweise werden vorgestellt und die Operatoren und Methoden in ihrem Ablauf umrissen. Die Eigenschaften Evolutionärer Algorithmen werden genannt und auf deren Auswirkungen für den Einsatz bzw. Nichteinsatz Evolutionärer Algorithmen hingewiesen. Damit soll dem Leser ein schneller Einstieg in das Gebiet der Evolutionären Algorithmen ermöglicht und ein Überblick gegeben werden, ob die Anwendung

Evolutionärer Algorithmen bei seinen zu lösenden Optimierungsproblemen sinnvoll ist. Für weitergehende Fragen wird gezielt auf die entsprechenden Kapitel und Abschnitte im folgenden Text verwiesen.

Auf diesen einführenden Erläuterungen aufbauend werden in Kap 3 die grundlegenden **Operatoren Evolutionärer Algorithmen** ausführlich vorgestellt und erläutert. Die Beschreibung der Bestandteile Evolutionärer Algorithmen geht von einer vereinfachenden Variante aus. Darauf fußend werden Erweiterungen des Aufbau, der Struktur und der einzelnen Operatoren und Algorithmen vorgestellt. Dieses Prinzip zieht sich durch alle Ebenen bis zur Erläuterung verschiedener Varianten von Operatoren und Methoden. Es wird immer versucht, die Gemeinsamkeiten aufzuzeigen. Dies erleichtert es dem Leser deutlich, die Auswirkungen der vorhandenen Unterschiede zu erkennen und besser abzuschätzen. Scheinbar unterschiedliche Algorithmen lassen sich dadurch oftmals als Spezialfälle anderer Algorithmen erkennen.

In Kap. 4 werden **Populationsmodelle** vorgestellt, die bei Evolutionären Algorithmen zum Einsatz kommen. Jedes der Modelle, globales, regionales und lokales Modell, setzt auf einer unterschiedlichen Abstraktionsebene an. Durch die Anwendung der Modelle wird eine verbesserte und differenzierte Modellierung der Wechselwirkungen zwischen Individuen in einer Population erreicht. Eine gleichzeitige Anwendung der Modelle in einer hierarchischen Struktur ist direkt möglich. Dies führt zu einer weiter verbesserten und robusteren Suche bei großen und komplexen Problemen.

Auf der Grundlage des regionalen Modells werden zwei Erweiterungen dieses Modells vorgestellt und ausführlich erläutert:

- die Anwendung verschiedener Strategien und
- die Konkurrenz zwischen Unterpopulationen.

Beide Erweiterungen führen zu einer verbesserten Modellierung der Prozesse in einer Population. Außerdem orientieren sich die Erweiterungen an Prozessen, die ähnlich in der Natur zu beobachten sind. Diese Erweiterungen führen zu einer Leistungssteigerung Evolutionärer Algorithmen und zu einer weiter verbesserten robusten Suche.

Speziell in der praktischen Anwendung können damit neue Prinzipien genutzt werden:

- gleichzeitige Verwendung verschiedener Strategien und
- effektive Verteilung der zur Verfügung stehenden Ressourcen zugunsten der erfolgreichen Strategien zwischen den Unterpopulationen.

Diese Prinzipien konnten bisher nur durch äußere Mechanismen zur Anwendung gebracht werden und waren damit nicht integraler Bestandteil Evolutionärer Algorithmen.

Bei der Arbeit mit Evolutionären Algorithmen werden große Mengen von Daten und Zwischenergebnissen produziert. Diese Daten sind relativ einfach strukturiert. Durch die auftretende Größe der Datenmenge ist es aber unmöglich, diese Zahlen direkt auszuwerten. In Kap. 5 werden **Methoden und Verfahren der Visualisierung** vorgestellt, die für eine Auswertung dieser Datenmengen gut geeignet sind. Durch eine Zusammenfassung der Daten und eine Abstraktion der vielen Einzelschritte innerhalb eines Evolutionären Algorithmus werden visuelle Verfahren entwickelt, die ein Verständnis des Verhaltens und der Arbeitsweise

Evolutionärer Algorithmen ermöglichen. Weiterhin können dadurch der Vergleich verschiedener Evolutionärer Algorithmen und die komfortable Interpretation der Ergebnisse durchgeführt werden.

Die dargestellten Visualisierungsverfahren ermöglichen eine Bewertung des Verlaufs eines Evolutionären Algorithmus unter verschiedenen Aspekten:

- auf verschiedenen zeitlichen Abstraktionsebenen (eine Generation oder mehrere Generationen),
- für unterschiedliche Mengen an Daten (einzelne Individuen, Teilpopulationen, ganze Populationen) sowie
- für direkte und abgeleitete Größen.

Außerdem werden Verfahren zur Visualisierung von Eigenschaften der Zielfunktion und zur hochdimensionalen Visualisierung vorgestellt. Insgesamt betrachtet stehen mit diesen Visualisierungsmethoden fortgeschrittene Verfahren zur Untersuchung Evolutionärer Algorithmen zur Verfügung, die besonders bei der Lösung praktischer Probleme von hohem Nutzen sind.

In Kap. 6 wird eine Implementierung der in den Kapiteln 2-5 dargestellten Verfahren und Methoden vorgestellt, die *Genetic and Evolutionary Algorithm Toolbox for use with Matlab* [GEATbx]. In diesem Programm Paket sind alle besprochenen (und weitere) Verfahren, Populationsmodelle und Visualisierungsmethoden enthalten. Sie stehen damit unter MATLAB [MW94] direkt zur Verfügung und können für die Lösung verschiedenster Probleme eingesetzt werden. Durch die Implementierung für MATLAB und die Erstellung der Dokumentation in HTML wurde eine vollständige Plattformunabhängigkeit der GEATbx erreicht. Die dem Buch beiliegende CD enthält Version 1.95 der GEATbx. Der Leser kann damit die besprochenen Verfahren und Methoden direkt auf eigene Probleme anwenden.

Nach den theoretischen Darstellungen und der programmtechnischen Umsetzung in den bisherigen Kapiteln wird in Kap. 7 gezeigt, wie sich aus den Evolutionären Operatoren und Methoden **leistungsfähige Evolutionäre Algorithmen zusammenstellen** lassen. Für einige in der Praxis häufig vorliegende Problemklassen (Parameteroptimierung, kombinatorische Optimierung) werden angepaßte Evolutionäre Algorithmen zusammengestellt und in ihrer Parametrisierung begründet. Mit diesen fertigen Evolutionären Algorithmen lassen sich viele Fragen zur Parametrisierung Evolutionärer Algorithmen beantworten. Die Kombination der einzelnen Operatoren und deren Parameter wird verständlich. Der Anwender kann diese angepaßten Algorithmen direkt auf seine eigenen Anwendungen und Problemstellungen übertragen bzw. anpassen.

In Kap. 8 wird die **Optimierung ausgewählter Anwendungen** detailliert vorgestellt. Den Anfang bildet eine allgemeine Darstellung des prinzipiellen Vorgehens bei der Bearbeitung eines neuen Optimierungsproblems, Abschn. 8.1. Dieser Abschnitt stellt eine einführende Anleitung dar. Es werden die Schritte erläutert, die bei der ingenieurtechnischen Lösung von Optimierungsproblemen zu beachten sind bzw. bei der Problemlösung helfen. Außerdem werden verschiedene Varianten zur Voruntersuchung der zu lösenden Optimierungsprobleme gezeigt, die auch bei der Verwendung anderer Optimierungsverfahren angewendet werden können bzw. sollten.

Die Anwendungsbeispiele in den folgenden Abschnitten von Kap. 8 wurden aus mehreren Problemklassen als jeweils repräsentative Beispiele ausgewählt. Die Probleme werden vorgestellt und die vorgenommenen Schritte zur Lösung der bestehenden Optimierungsaufgabe ausführlich gezeigt sowie die verwendeten Evolutionären Algorithmen beschrieben. Dadurch werden dem Leser Beispiele gegeben, die direkt auf die eigenen zu lösenden Aufgaben übertragen werden können.

Die Darstellung des Lösungsweges erfolgt je nach der Komplexität der behandelten Aufgabe in verschiedenen Richtungen. Einfache Probleme dienen der Demonstration des prinzipiellen Herangehens an die Problemlösung. Bei den komplexeren Problemen wird u.a. gezeigt, wie durch Voruntersuchungen des Systemverhaltens und die Einbeziehung von problemspezifischem Wissen die Problemgröße verringert bzw. die Lösung des Problems vereinfacht werden kann. Außerdem wird die Anwendung der Visualisierungsmethoden gezeigt.

Einige der vorgestellten Anwendungen stellen komplexe Probleme dar, die mit anderen Optimierungsverfahren nicht oder nur schwer zu lösen gewesen wären. Insbesondere die einfache Art und Weise, mit der bei Evolutionären Algorithmen problemspezifisches Wissen in den Optimierungsprozeß eingebunden werden kann, stellt einen Vorteil für die Lösung dieser Probleme dar. So konnten für einige der Anwendungen neue Ergebnisse gefunden bzw. Aufgaben in einer Größenordnung gelöst werden, die bisher nicht bearbeitbar waren.

In Kap. 9 wird eine Zusammenfassung des Buches gegeben. Der folgende Ausblick beleuchtet einige Aspekte Evolutionärer Algorithmen, die in den kommenden Jahren größere Bedeutung erlangen könnten, wenn sehr große und komplexe Systeme gelöst werden müssen.

Im Anhang wird ausführlich auf die **historische Entwicklung Evolutionärer Algorithmen** eingegangen. Erste Ansätze zur Entwicklung und Verwendung Evolutionärer Algorithmen wurden in den 50er Jahren unternommen. In den 80er Jahren wurden die Evolutionären Algorithmen populär. Aber erst in den 90er Jahren verzeichnete dieses Gebiet eine rasante Entwicklung. Abschnitt A.1 stellt frühe Arbeiten Evolutionärer Algorithmen vor und gibt einen kurzen Überblick über die aktuellen Entwicklungen. Der Rest des Anhangs enthält erweiterte Informationen zu den Anwendungen aus Kap. 8, die für ein tieferes Verständnis der verwendeten Modelle oder Simulationen notwendig sind, bei der Erläuterung der Lösung der Probleme aber unverhältnismäßig viel Platz eingenommen hätten.

Das **Literaturverzeichnis** ab S.281 stellt die in dieser Arbeit verwendete Literatur zusammen. Die einzelnen Einträge wurden dabei thematischen Gebieten zugeordnet, die sich weitgehend an der Gliederung dieser Arbeit orientieren. Dadurch wird neben der einfachen Aufführung der Arbeiten gleichzeitig ein besserer Überblick über die Arbeiten eines thematischen Bereiches gegeben. Im folgenden **Glossar** werden in einer kompakten Form viele Begriffe aus dem Gebiet der Evolutionären Algorithmen sowie aus angrenzenden Bereichen erläutert. Das ausführliche **Sachverzeichnis** bietet einen schnellen Zugang zu allen Bereichen des Buches über die entsprechenden Stichworte.

## 2 Struktur und Aufbau Evolutionärer Algorithmen

Dieses Kapitel beschreibt Struktur und Aufbau Evolutionärer Algorithmen übersichtsartig. Ausgehend von der Definition Evolutionärer Algorithmen und einer kurzen Beschreibung der Arbeitsweise erfolgt die Entwicklung einer einfachen Struktur Evolutionärer Algorithmen. Diese wird nachfolgend durch spezielle Verfahren erweitert. Dabei bleibt die grundlegende Struktur aber weiterhin erhalten.

Die vorgestellte Struktur Evolutionärer Algorithmen stellt die Grundlage für eine einheitliche Beschreibung der Bestandteile Evolutionärer Algorithmen dar. Diese Struktur bezieht viele bisher getrennt betrachtete Varianten Evolutionärer Algorithmen ein. Die bisher mit einer voneinander abweichenden Struktur betrachteten Verfahren der Evolutionsstrategien, der Evolutionären Programmierung und der Genetischen Algorithmen lassen sich damit auf eine gemeinsame Struktur zurückführen. Weitere Varianten können als Spezialfälle eingeordnet werden.

Anschließend wird kurz auf die grundlegenden Bestandteile Evolutionärer Algorithmen und wichtiger Erweiterungen eingegangen sowie deren Funktionsweise umrissen. Diese Charakterisierung dient einem schnellen Erfassen der Funktion und Aufgabe der einzelnen Operatoren und Methoden.

Zum Abschluß werden die wichtigsten Eigenschaften Evolutionärer Algorithmen genannt und auf deren Auswirkungen für den Einsatz bzw. Nichteinsatz Evolutionärer Algorithmen hingewiesen.

Mit diesem Kapitel soll dem Leser ein schneller Einstieg in das Gebiet der Evolutionären Algorithmen ermöglicht werden. Außerdem wird dem Leser ein Überblick gegeben, ob die Anwendung Evolutionärer Algorithmen bei seinen zu lösenden Optimierungsproblemen sinnvoll ist. Für weitergehende Fragen wird gezielt auf die folgenden Kapitel verwiesen.

### **Was sind Evolutionäre Algorithmen?**

Evolutionäre Algorithmen sind stochastische Suchverfahren, die an die Prinzipien der natürlichen biologischen Evolution angelehnt sind. Evolutionäre Algorithmen arbeiten gleichzeitig auf einer Anzahl von potentiellen Lösungen, der Population von Individuen. Auf diese Individuen (Lösungen) wird das Prinzip „Der Stärkere überlebt!“ angewendet, um im Sinne einer Zielfunktion immer bessere Individuen zu erzeugen, die am Ende zu einer guten Lösung führen.

### **Wie arbeiten Evolutionäre Algorithmen?**

Nach einer Initialisierungsphase zu Beginn wird in einem Prozeß über mehrere Generationen eine Suche durchgeführt. In jeder Generation wird eine Anzahl neuer Lösungen erstellt. Dies erfolgt durch die Auswahl von Individuen entsprechend ihrer Fitneß, aus denen nachfolgend durch die Anwendung evolutionärer Reproduktionsoperatoren neue Individuen erzeugt werden. Diese Individuen werden in die Population eingefügt, wodurch eine neue Population entsteht. Die neue Population dient als Ausgangspunkt für die Erstellung neuer Individuen in der nächsten Generation. Dieser Prozeß führt zur Evolution (Entwicklung) von Populationen von Individuen, die immer besser an die jeweilige Zielstellung angepaßt sind, als die Individuen, von denen sie erzeugt wurden.<sup>1</sup> Dies entspricht der natürlichen Anpassung. Da Evolutionäre Algorithmen mit Populationen von Individuen arbeiten<sup>2</sup>, wird die Suche im Probletraum in einer parallelen Art und Weise durchgeführt.

Evolutionäre Algorithmen modellieren verschiedene natürliche Prozesse, z.B. Selektion, Reproduktion, Rekombination, Mutation, Migration, Lokalität, Nachbarschaft, Parallelität und Konkurrenz. Jeder dieser Prozesse kann in einer Vielzahl von Varianten auftreten.

### **Wie sind Evolutionäre Algorithmen strukturiert?**

Abbildung 2-1 zeigt die Struktur eines einfachen Evolutionären Algorithmus. Diese Struktur ist die Minimalvariante eines Evolutionären Algorithmus, die als Ausgangspunkt für die folgenden Erläuterungen dient. Erweiterungen lassen sich in diese Struktur später gut eingliedern, wobei die grundlegende Struktur erhalten bleibt.

Zu Beginn der Arbeit eines Evolutionären Algorithmus erfolgt die Initialisierung. Diese beinhaltet neben der Parametrisierung vor allem die Erstellung der Anfangspopulation. Normalerweise werden die Individuen der Anfangspopulation zufällig im Definitionsbereich der Variablen initialisiert. Allerdings können für diese Initialisierung auch problemspezifische Verfahren verwendet werden. Die erstellte Anfangspopulation wird durch die Zielfunktion bewertet. Damit ist die Population der ersten Generation produziert.

Wenn das definierte Abbruchkriterium<sup>3</sup> noch nicht erreicht ist, beginnt die Erstellung einer neuen Generation und damit der evolutionäre Kreislauf. Jedem Individuum wird entsprechend seines Zielfunktionswertes und im Vergleich zu allen anderen Individuen der Population eine Fitneß zugewiesen (Fitneßzuweisung). Entsprechend dieser Fitneß werden die Individuen (Eltern) für die Produktion von neuen Individuen (Nachkommen) ausgewählt (Selektion).

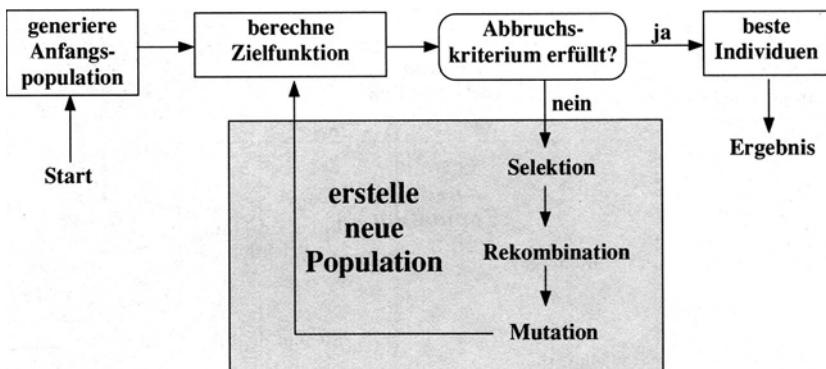
---

<sup>1</sup> Voraussetzung ist die Wahl von evolutionären Operatoren, die aus guten Individuen mit einer Wahrscheinlichkeit größer als Null bessere Nachkommen produzieren.

<sup>2</sup> Dies steht im Gegensatz zu vielen klassischen Optimierungsverfahren, bei denen eine einzelne Lösung verwendet wird.

<sup>3</sup> Abbruchkriterium kann das Erreichen eines bestimmten Zielfunktionswertes, einer Anzahl von Generationen, das Verstreichen einer festgelegten Zeitspanne oder einer Anzahl von Generationen ohne weiteren Fortschritt sein.

Die Eltern produzieren unter Anwendung evolutionärer Operatoren (Rekombination, Mutation) die Nachkommen. Durch Rekombination werden die Nachkommen erstellt und diese werden dann durch Mutation mit einer gewissen Wahrscheinlichkeit verändert. Die neuen Individuen werden durch die Zielfunktion bewertet und in die Population eingefügt, wobei Individuen der Population durch die Nachkommen ersetzt werden. Damit ist eine neue Population erstellt. Wenn das Abbruchkriterium noch nicht erfüllt ist, beginnt der Prozeß der Erstellung einer neuen Population von vorn.



**Abb. 2-1.** Struktur eines einfachen Evolutionären Algorithmus

Ein einfacher Evolutionärer Algorithmus, wie er gerade beschrieben wurde, ist leistungsstark und arbeitet auf einer breiten Klasse von Problemen gut.

Die oben genannten grundlegenden Bestandteile Evolutionärer Algorithmen werden ausführlich in Kap. 3, ab S.13 beschrieben.

### Wie kann ein einfacher Evolutionärer Algorithmus erweitert werden?

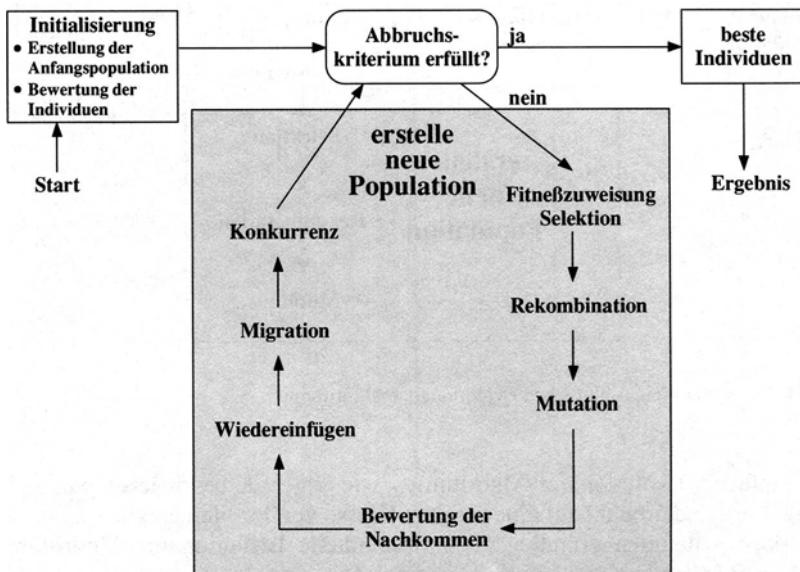
Es gibt eine Anzahl von Erweiterungen für Evolutionäre Algorithmen, durch die bessere Ergebnisse erzielt werden können. Dies beinhaltet vor allem spezielle Operatoren und Verfahren sowie die erweiterte Behandlung der Population.

Ein Beispiel sind erweiterte Methoden zum Einfügen der Nachkommen in die Population und zur Ersetzung der Eltern (Wiedereinfügen). Dadurch können die Leistungsfähigkeit Evolutionärer Algorithmen erhöht sowie spezielle Verhaltensweisen erreicht werden. Außerdem lassen sich durch diese Erweiterung mehrere bisher getrennt betrachtete Verfahren einheitlich und innerhalb der hier vorgestellten Struktur betrachten. Methoden des Wiedereinfügens werden in Abschn. 3.5 beschrieben.

Eine erweiterte Behandlung der Population erfolgt durch die Anwendung von Populationsmodellen und deren Erweiterung. Beispiele dafür sind die Unterteilung der Population in Unterpopulationen, die Anwendung verschiedener Strate-

gien für jede Unterpopulation und miteinander konkurrierende Unterpopulationen. In diesen Erweiterungen verhält sich jede Unterpopulation wie die Gesamtpopulation in einem einfachen Evolutionären Algorithmus, es werden nur zusätzliche Elemente eingeführt. Populationsmodelle und ihre Erweiterungen werden in Kap. 4 ausführlich vorgestellt.

In Abb. 2-2 ist die Struktur eines erweiterten Evolutionären Algorithmus dargestellt. Die Erweiterungen lassen sich ohne Probleme in die in Abb. 2-1 vorgestellte Struktur einfügen, die Grundstruktur der Evolutionären Algorithmen bleibt erhalten.



**Abb. 2-2.** Struktur eines erweiterten Evolutionären Algorithmus

Die erweiterten Evolutionären Algorithmen sind leistungsfähiger als einfache Evolutionäre Algorithmen. Außerdem modellieren sie die Evolution in einer Art, die der in der Natur vorhandenen ähnlicher ist, als dies bei einfachen Evolutionären Algorithmen der Fall ist.

Die Auswertung der während der Arbeit des Evolutionären Algorithmus entstandenen Datenmengen ist ein wichtiger Aspekt, der bei der ingenieurtechnischen Anwendung oftmals über den Erfolg entscheidet. Nur durch eine „Erweiterung“ der Evolutionären Algorithmen um leistungsfähige Visualisierungsmethoden ist eine erfolgreiche Anwendung der verschiedenen Operatoren und ihrer Kombinationen möglich. Die Entwicklung bzw. Zusammenstellung spezieller Evolutionärer Algorithmen, die an bestimmte Aufgabenstellungen oder Problemklassen angepaßt sind, ist ohne diese Visualisierungsmethoden nur

schwer möglich. Deshalb wird eine ausführliche Behandlung von Visualisierungsmethoden für Evolutionäre Algorithmen in Kap. 5 vorgenommen.

### **Was unterscheidet Evolutionäre Algorithmen von traditionellen Optimierungsverfahren?**

Evolutionäre Algorithmen sind stochastische Suchverfahren, die an die Prinzipien der natürlichen biologischen Evolution angelehnt sind und mit einer Population von Individuen arbeiten. Dadurch ergeben sich einige grundlegende Unterschiede zu traditionellen deterministischen wie auch anderen stochastischen Suchverfahren.

Die wichtigsten Unterschiede sind:

- Evolutionäre Algorithmen suchen in einer Population von Punkten parallel, nicht nur von einem einzelnen Punkt aus.
- Evolutionäre Algorithmen benötigen keine Ableitungen der Zielfunktionen oder andere Hilfsinformationen. Nur der Zielfunktionswert wird als Grundlage für die Suche verwendet.
- Evolutionäre Algorithmen verwenden Wahrscheinlichkeitsregeln, keine deterministischen Regeln.
- Evolutionäre Algorithmen können eine Anzahl von möglichen Lösungen für ein Problem anbieten. Die abschließende Auswahl kann dem Benutzer überlassen werden.<sup>4</sup>
- Evolutionäre Algorithmen können auf Probleme der verschiedensten Repräsentationen (kontinuierliche und diskrete Variablen) angewendet werden.
- Evolutionäre Algorithmen arbeiten sowohl im kontinuierlichen als auch im diskontinuierlichen und multimodalen Funktionsraum.
- Evolutionäre Algorithmen sind im allgemeinen einfach und flexibel anzuwenden, da es für die Festlegung der Zielfunktion keine Einschränkungen gibt.

Evolutionäre Algorithmen stellen vielseitige, robuste und leistungsfähige Such- und Optimierungsverfahren dar.

### **Wann sollten Evolutionäre Algorithmen eingesetzt werden?**

Durch die oben genannten Unterschiede zu anderen Optimierungsverfahren bietet sich der Einsatz Evolutionärer Algorithmen insbesondere bei den folgenden Fällen an:

- bei stark nichtlinearen Zielfunktionen,
- bei diskontinuierlichen Zielfunktionen,

---

<sup>4</sup> Dadurch ist der Evolutionäre Algorithmus besonders dann von Vorteil, wenn das spezielle Problem nicht eine eindeutige Lösung, sondern eine Anzahl von PARETO-optimalen Lösungen hat. Dies kann bei der mehrkriteriellen Optimierung bzw. bei Scheduling-Problemen auftreten und hilft bei der Identifizierung mehrerer alternativer Lösungen gleichzeitig.

- bei multimodalen Zielfunktionen und
- bei Problemen mit Variablen unterschiedlicher Repräsentation (binär, ganzzahlig, reell).

Es gibt eine Vielzahl unterschiedlicher Evolutionärer Algorithmen. Durch den Einsatz verschiedener Operatoren und die Anpassung von deren Optionen kann eine breite Palette von Algorithmen für die Optimierung der unterschiedlichsten Probleme erstellt werden. Es gibt nicht DEN Evolutionären Algorithmus, sondern die Gesamtheit der eingesetzten Operatoren und Optionen definiert einen Evolutionären Algorithmus.

In Kap. 7 werden leistungsfähige Evolutionäre Algorithmen in ihrer Zusammenstellung und Parametrisierung für die Optimierung ausgewählter Problemklassen beschrieben. Aus dem dort beschriebenen Spektrum wird ersichtlich, daß mit Evolutionären Algorithmen eine Vielzahl von Optimierungsverfahren definiert werden kann, deren Spannweite von stark global orientierten bis hin zu lokal ausgerichteten Suchverfahren reicht. Je mehr über die Charakteristik eines Problems bekannt ist und je besser der eingesetzte Evolutionäre Algorithmus darauf abgestimmt werden kann, um so erfolgreicher wird dessen Einsatz sein.

### ***Wann sollten Evolutionäre Algorithmen NICHT eingesetzt werden?***

Evolutionäre Algorithmen können im Prinzip auf (fast) alle Probleme angewendet werden. Allerdings sollten sie auf gar keinen Fall als Ersatz für klassische Optimierungsmethoden angesehen werden.

Insbesondere sollten Evolutionäre Algorithmen nicht eingesetzt werden, wenn:

- ein spezielles Lösungsverfahren für das Problem bekannt ist und
- die Berechnung der Zielfunktion sehr aufwendig ist.

Ein spezielles Lösungsverfahren wird immer besser als ein allgemeiner Evolutionärer Algorithmus sein. In einem speziellen Verfahren steckt viel problemspezifisches Wissen, ohne dessen Ausnutzung der Evolutionäre Algorithmus langsamer sein muß.

Evolutionäre Algorithmen führen eine größere Anzahl von Zielfunktionsberechnungen durch. Deshalb ist der Einsatz Evolutionärer Algorithmen nicht praktikabel, wenn diese Berechnungen sehr lange dauern. Allerdings ist die Grenze fließend und verschiebt sich durch die zunehmende Leistungsfähigkeit der eingesetzten Rechentechnik von Jahr zu Jahr. Probleme, die vor wenigen Jahren als zu groß eingestuft wurden, bereiten heute kaum noch Schwierigkeiten. Dieser Trend wird sich fortsetzen und kann durch eine vernünftige Parallelisierung der Berechnungen weiter verstärkt werden [PPW99].

### 3 Grundlegende Verfahren und Operatoren

Dieses Kapitel beschreibt die Funktionsweise von grundlegenden Verfahren und Operatoren Evolutionärer Algorithmen. Die Erläuterungen bauen auf der einführenden Beschreibung in Kap. 2 auf.

Bei der Darstellung der Funktionsweise der Operatoren wird eine einheitliche Beschreibung verwendet, welche die Gemeinsamkeiten betont. Viele Operatoren lassen sich auf ein Verfahren zurückführen und unterscheiden sich nur in der Wahl eines oder mehrerer Parameter. Durch diese Hervorhebung der Gemeinsamkeiten der Operatoren lassen sich die vorhandenen Unterschiede leichter erkennen und das unterschiedliche Verhalten der Operatoren direkter beschreiben und einschätzen. Dies ermöglicht ein besseres Verständnis des Zusammenwirkens der Verfahren und der Zusammenstellung entsprechender Algorithmen. Außerdem können durch diese einheitliche Darstellung neue Operatoren und Verfahren in den Kontext vorhandener Verfahren eingeordnet und damit leicht auf ihre „Neuheit“ hin eingeschätzt werden.

Zu Beginn des Kapitels wird eine Übersicht zu den evolutionären Operatoren und deren Verfahren gegeben. In den folgenden Abschnitten werden die einzelnen Operatoren und Verfahren ausführlich dargestellt. In Abschn. 3.1 wird die Fitneßzuweisung und in Abschn. 3.2 die Selektion erläutert. Abschnitt 3.3 behandelt die verschiedenen Rekombinationsverfahren. In Abschn. 3.4 wird die Mutation dargestellt und das Wiedereinfügen (*reinsertion*) in Abschn. 3.5. Den Abschluß bilden Hinweise zur Initialisierung der Individuen in Abschn. 3.6 und zu Abbruchkriterien für die Optimierung in Abschn. 3.7.

Die meisten der folgenden Erläuterungen können auf (fast) alle Problemklassen angewendet werden. Dies betrifft insbesondere den Bereich der Fitneßzuweisung, der Selektion, des Wiedereinfügens und der Abbruchkriterien. Ausführlichere Erläuterungen dazu mit besonderem Schwerpunkt auf der Anwendung der einzelnen Operatoren werden in Kap. 7 gegeben.

Bei der Erläuterung der problemspezifischen Operatoren für Rekombination, Mutation und Initialisierung der Individuen liegt das Hauptaugenmerk auf Verfahren für die Parameteroptimierung. Für kombinatorische Probleme werden jeweils einige bekannte Operatoren und Verfahren vorgestellt.

#### **Fitneßzuweisung und Selektion**

In der Fitneßzuweisung wird festgelegt, wieviele Nachkommen jedes Individuum produziert. Die Selektion entscheidet, welche Individuen für die Fortpflanzung ausgewählt werden.

Die Fitneßzuweisung erfolgt durch eines der folgenden Verfahren:

- proportionale Fitneßzuweisung (*proportional fitness assignment*) [Gol89],
- reihenfolgebasierte Fitneßzuweisung (*rank-based fitness assignment*) [Bak85] oder
- mehrkriterielle Fitneßzuweisung (*multi-objective ranking*) [ZDT99].

Die direkte Auswahl (Selektion) der Individuen wird im nächsten Schritt durchgeführt. Eltern werden entsprechend ihrer Fitneß mittels eines der folgenden Verfahren ausgewählt:

- Rouletteselektion (*roulette-wheel selection*) [Bak87],
- *stochastic universal sampling* [Bak87],
- Truncation-Selektion (Abschneide Selektion) [MSV95] oder
- Turnierselektion (*tournament selection*) [BT95].

## ***Rekombination***

Durch die Rekombination werden aus den Eltern die Nachkommen erstellt. Dazu wird die Information der Eltern miteinander kombiniert (die Eltern sind die Paarungspopulation). In Abhängigkeit der Repräsentation der Variablen in den Individuen (reell, ganzzahlig, binär) und des Einsatzzweckes (Art der Zielfunktion: Parameteroptimierung, kombinatorische Optimierung) können bzw. müssen verschiedene Verfahren angewendet werden.

Rekombination für die Parameteroptimierung:

- alle Repräsentationen (reell, ganzzahlig, binär):
  - diskrete Rekombination (*discrete recombination*) [MSV93a] (bekannt für die Rekombination reeller Variablen), entspricht *uniform crossover* [Sys89] (bekannt für die Rekombination binärer Variablen).
- Rekombination reeller Variablen:
  - intermediäre Rekombination (*intermediate recombination*) [MSV93a],
  - Linien-Rekombination (*line recombination*) [MSV93a],
  - erweiterte Linien-Rekombination (*extended line recombination*) [Müh94].
- Rekombination binärer Variablen:
  - *single-point/double-point/multi-point crossover*<sup>1</sup>,
  - *shuffle crossover* [CES89],
  - *crossover with reduced surrogate* [Boo87].

Rekombination für die kombinatorische Optimierung:

- *edge recombination* [WSF89] und [WSS91],
- *maximal preservative crossover* [Müh91],
- *order bzw. position crossover* [Sys91].

---

<sup>1</sup> Für die Rekombination binärer Variablen hat sich der Begriff *crossover* (Kreuzung) eingebürgert. Dies hat vor allem historische Ursachen: Bei Genetischen Algorithmen wurden anfangs fast nur binäre Variablen verwendet und für die Rekombination immer der Begriff *crossover* benutzt. Beide Begriffe sind aber gleichwertig. Im weiteren wird, außer bei der Benennung spezieller Verfahren, immer der Begriff Rekombination verwendet.

## **Mutation**

Nach der Rekombination werden die Nachkommen einer Mutation unterzogen. Dabei werden die Variablen der Nachkommen durch kleine Störungen (Mutationsschritt) verändert. Diese Veränderungen geschehen mit einer geringen Wahrscheinlichkeit. In Abhängigkeit von der Repräsentation der Variablen und der Art der Zielfunktion können u.a. die folgenden Verfahren verwendet werden:

- Mutation für reelle Variablen:
  - kontinuierliche und verteilte Mutation nach [VMC95] und [SVM96],
  - Mutation einer Evolutionsstrategie nach [OGH94] und [HOG95],
- Mutation für ganzzahlige (integer) Variablen:
  - in Analogie zur reellen Mutation (für Parameteroptimierung),
- Mutation für binäre Variablen:
  - Veränderung des Variablenwertes,
- Mutation für Reihenfolgeoptimierung:
  - Vertauschung von Variablen (*swap/insert/reverse/scramble mutation*).

## **Wiedereinfügen (Reinsertion)**

Nachdem die Nachkommen produziert wurden, müssen sie in die Population eingefügt werden. Dies ist besonders wichtig, wenn die Nachkommen nicht einfach die Elternpopulation ersetzen, sondern z.B. weniger Individuen produziert wurden, als in der Population enthalten sind oder wenn nicht alle produzierten Nachkommen in die Population eingefügt werden sollen. Durch das Wiedereinfügeverfahren wird entschieden, welche Nachkommen in die Population eingefügt werden und welche Individuen der Population ersetzt werden („sterben“).

## **Initialisierung der Individuen**

Zu Beginn eines Evolutionären Algorithmus müssen die Individuen der Anfangspopulation erstellt werden. Meistens wird eine zufälligen Initialisierung der Individuen innerhalb des vorgegebenen Suchraumes vorgenommen.

Oftmals liegt aber Vorwissen über ein zu lösendes Problem in Form einiger bekannter guter Lösungen (Individuen) vor. Diese bekannten Lösungen können und sollten in die Optimierung einbezogen werden. Eine Form der Einbeziehung dieses Wissens ist die nicht-zufällige Initialisierung der Anfangspopulation. Diese Methode wird als erweiterte Initialisierung bezeichnet.

## **Abbruchkriterien**

Ein evolutionärer Algorithmus läuft so lange, bis ein festgelegtes Abbruchkriterium erfüllt ist. Im allgemeinen wird ein Lauf auf eine maximale Anzahl von Generationen begrenzt. Daneben existieren eine Reihe weiterer Abbruchkriterien, die sich in direkte und abgeleitete Abbruchkriterien unterteilen lassen.

Direkte Abbruchkriterien:

- maximale Anzahl von Generationen bzw. Zielfunktionsberechnungen,
- maximale Rechenzeit,
- Erreichen eines „globales“ Optimum.

Abgeleitete Abbruchkriterien:

- Standardabweichung der Zielfunktionswerte der aktuellen Generation,
- *laufender Mittelwert*,
- *GuterSchlechtester*,
- *Phi* (Mittelwert der Zielfunktionswerte),
- *Kappa* (Verteilung der Individuen im Suchraum).

### 3.1 Fitneßzuweisung

Durch die Fitneßzuweisung wird jedem Individuum der Population eine Fitneß zugewiesen. Die Fitneß (Fortpflanzungswahrscheinlichkeit bzw. Selektionswahrscheinlichkeit) wird aus dem Zielfunktionswert des Individuums und den Zielfunktionswerten aller anderen Individuen des Selektionspools<sup>2</sup> unter Verwendung einer Fitneßfunktion gebildet. Die Fitneßfunktion führt eine Transformation der Zielfunktionswerte in nichtnegative Werte durch. Dabei erhalten bessere Individuen (bessere Güte) eine höhere Fitneß als schlechte Individuen. Die Fitneß eines Individuums ist immer im Vergleich zu allen anderen Individuen des Selektionspools zu betrachten (im Gegensatz zum Zielfunktionswert, der nur von den Variablen des Individuums abhängt).

Die Fitneß wird im folgenden Schritt, der aktuellen Auswahl der Individuen (Selektion, Abschn. 3.2), verwendet. Außerdem kann die Fitneß an anderen Stellen innerhalb des Evolutionären Algorithmus zum Vergleich der Güte der Individuen untereinander verwendet werden (z.B. Migration, Abschn. 4.4 und Konkurrenz, Abschn. 4.6).

Für die Fitneßzuweisung lassen sich die folgenden Verfahren unterscheiden:

- proportionale Fitneßzuweisung (*proportional fitness assignment* ), Unterabschn. 3.1.1,
- reihenfolgebasierte Fitneßzuweisung (*rank-based fitness assignment*), Unterabschn. 3.1.2.

Bisher wurde davon ausgegangen, daß die Zielfunktion nur einen Zielfunktionswert liefert. Bei vielen Problemen liegen aber mehrere Werte vor, aus denen eine Bewertung der Qualität einer Lösung erfolgen muß. Die Zielfunktion liefert mehrere Zielfunktionswerte bzw. Kriterien.

---

<sup>2</sup> Der Selektionspool enthält alle Individuen, die zusammen zur Auswahl der Eltern in Betracht gezogen werden. Bei einer Gesamtpopulation (globales Populationsmodell) ist dies die Population. Wird dagegen das regionalen Populationsmodell verwendet, stellt jede Unterpopulation einen Selektionspool dar (s. Abb. 4-1, S.75).

In diesem Fall kommt die folgende Methode zum Einsatz:

- mehrkriterielle Fitneßzuweisung (z.B. *multi-objective ranking*), Unterabschn. 3.1.3.

Auf jedes dieser Verfahren wird im folgenden eingegangen. Dabei wird die proportionale Fitneßzuweisung nur aus Gründen der Vollständigkeit behandelt, da ihre Anwendung in der Praxis viele Probleme (s. Abb. 3-1) bereiten kann. Die reihenfolgebasierte (*ranking*) und die mehrkriterielle Fitneßzuweisung (*multi-objective ranking*) werden dagegen ausführlich behandelt. Beide Methoden haben eine hohe Bedeutung in der Praxis und sind in ihren Anwendungsbereichen die Methoden der Wahl.

### 3.1.1 Proportionale Fitneßzuweisung

Bei der proportionalen Fitneßzuweisung ([Gol89]) wird jedem Individuum ein Fitneßwert zugewiesen, der proportional zu seinem Zielfunktionswert ist. Für die Fitneßfunktion wurden verschiedene Verfahren (u.a. [GB89]) verwendet. Die wichtigsten Skalierungen sind in Gl. 3-1 dargestellt.

$$\begin{aligned}
 \text{lineare Skalierung:} & \quad \text{Fitneß}(Ind) = a \cdot ZFW(Ind) + b \\
 \text{linear dynamische Skalierung:} & \quad \text{Fitneß}(Ind) = a \cdot ZFW(Ind) + b(Generation) \\
 \text{logarithmische Skalierung:} & \quad \text{Fitneß}(Ind) = b - \log(ZFW(Ind)) \\
 \text{exponentielle Skalierung:} & \quad \text{Fitneß}(Ind) = (a \cdot ZFW(Ind) + b)^k
 \end{aligned} \tag{3-1}$$

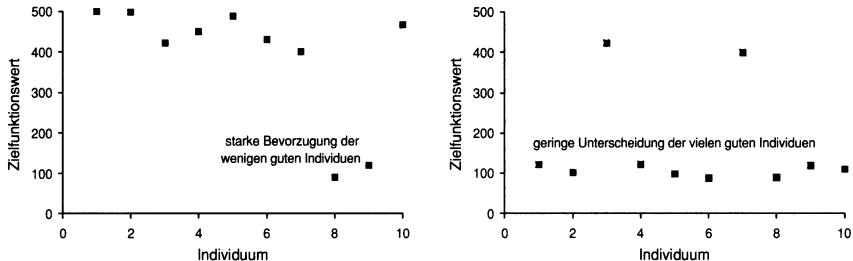
ZFW: Zielfunktionswert; Ind: Individuum  
 $a, b, k$ : Skalierungsgrößen (problemspezifisch)

Bei der Verwendung dieser Fitneßfunktionen treten jedoch Probleme auf. Abbildung 3-1 dient zur Veranschaulichung zweier Beispiele, an denen sich diese Probleme deutlich zeigen.

Im linken Diagramm von Abb. 3-1 sind eine Anzahl von Zielfunktionswerten dargestellt, wobei zwei Individuen einen sehr niedrigen Zielfunktionswert haben, alle anderen Individuen einen wesentlich höheren und in etwa gleichen Wert. Bei einer proportionalen Fitneßzuweisung würden die beiden guten Individuen eine sehr hohe Fitneß im Vergleich zu den schlechten Individuen zugewiesen bekommen. Dies hätte zur Folge, daß die wenigen guten Individuen sehr viele Nachkommen produzieren würden, von den schlechten Individuen würden nur wenige überhaupt eine Chance zur Produktion von Nachkommen bekommen. Die in der Population vorhandene Mannigfaltigkeit würde dadurch sehr schnell verloren gehen. Dieser Vorgang wird als vorzeitige Konvergenz (*premature convergence*) bezeichnet und hat seine Ursache im zu hohen Selektionsdruck, der durch die proportionale Fitneßzuweisung erzeugt wurde.

Im rechten Diagramm von Abb. 3-1 sind Zielfunktionswerte dargestellt, bei denen viele ähnlich gute Individuen auftreten und nur wenige schlechte Individuen. In diesem Fall würde es bei der proportionalen Fitneßzuweisung zu einer sehr geringen Unterscheidung in den Fitneßwerten der guten Individuen kommen. Die Auswahl zwischen den guten Individuen würde eher einer zufälligen Auswahl gleichen. Dies wird als Steckenbleiben (*stagnation*) bezeichnet und durch den zu

geringen Selektionsdruck hervorgerufen, der durch die proportionale Fitneßzuweisung erzeugt wurde.



**Abb. 3-1.** Probleme bei der Anwendung der proportionalen Fitneßzuweisung; links: die wenigen guten Individuen (Minimierung: ein kleinerer Zielfunktionswert ist besser) werden zu stark bevorzugt, rechts: die vielen guten Individuen unterscheiden sich untereinander kaum in ihrer Fitneß

Zusammenfassend kann festgestellt werden, daß durch die Fitneßzuweisung eine Verteilung der Fitneßwerte in der Art zu erfolgen hat, daß wenige gute Individuen keine übermäßige Anzahl von Nachkommen produzieren und bei vielen guten Individuen trotzdem eine signifikante Unterscheidung der Fitneßwerte der guten Individuen stattfindet.

Diese Aufgabe einer effektiven Kontrolle des Selektionsdruckes kann durch eine proportionale Fitneßzuweisung nicht allgemein gelöst werden. Für spezielle Anwendungen dagegen, bei denen die Verteilung der Zielfunktionswerte in etwa bekannt ist, kann die proportionale Fitneßzuweisung ausreichend sein.

Eine robuste Lösung bringt erst die reihenfolgebasierte Fitneßzuweisung. Die proportionale Fitneßzuweisung sollte heute, außer in Spezialfällen, nicht mehr verwendet werden<sup>3</sup> und wird deshalb hier auch nicht weiter erläutert.

### 3.1.2 Reihenfolgebasierte Fitneßzuweisung (Ranking)

Bei der reihenfolgebasierten Fitneßzuweisung ([Bak85]) wird der Selektionspool entsprechend der Zielfunktionswerte sortiert. Die einem Individuum zugewiesene Fitneß hängt hier nur von der Position/dem Rang des Individuums in dieser sortierten Liste ab (und damit nicht vom Zielfunktionswert). Damit werden die oben besprochenen Probleme der proportionalen Fitneßzuweisung vermieden. Durch die reihenfolgebasierte Fitneßzuweisung wird eine gleichmäßige Skalierung über den Selektionspool eingeführt, die zu einem einfachen und effektiven Weg der Kontrolle der Verteilung der Fitneßwerte führt.

<sup>3</sup> Daß die proportionale Fitneßzuweisung eine ungeeignete Fitneßzuweisung zur Optimierung ist, wird auch in [Bli97], S.65 bei der Analyse verschiedener Selektionsverfahren festgestellt.

Für die Durchführung der Fitneßzuweisung werden folgende Variablen verwendet:  $Nind$  ist die Anzahl der Individuen im Selektionspool,  $Pos$  die Position/der Rang eines Individuum im sortierten Selektionspool (das schlechteste Individuum hat die Position  $Pos = 1$ , das beste Individuum die Position  $Pos = Nind$ ),  $SP$  ist der Selektionsdruck.

Zwei Fitneßfunktionen werden vorgestellt: lineares und nichtlineares Ranking. Damit berechnet sich der Fitneßwert eines Individuum durch:

Lineares Ranking:

$$Fitneß(Pos) = 2 - SP + 2 \cdot (SP - 1) \cdot \frac{(Pos - 1)}{(Nind - 1)} \quad (3-2)$$

Lineares Ranking erlaubt einen Selektionsdruck im Bereich [1,0 2,0].

In [Poh95] wurde eine Methode des Ranking unter Verwendung einer nichtlinearen Verteilung eingeführt, welche die direkte Angabe des gewünschten Selektionsdruckes erlaubt. Die Benutzung des nichtlinearen Ranking erlaubt die Verwendung eines höheren Selektionsdruckes, als dies beim linearen Ranking möglich ist.

Nichtlineares Ranking:

$$Fitneß(Pos) = \frac{Nind \cdot X^{Pos-1}}{\sum_{i=1}^{Nind} X^{i-1}} \quad (3-3)$$

$X$  kann bei Angabe des Selektionsdruckes  $SP$  und der Individuenanzahl  $Nind$  direkt als Lösung des folgenden Polynoms berechnet werden:

$$0 = (SP - 1) \cdot X^{Nind-1} + SP \cdot X^{Nind-2} + \dots + SP \cdot X + SP \quad (3-4)$$

Nichtlineares Ranking erlaubt einen Selektionsdruck im Bereich [1,  $Nind-2$ ].

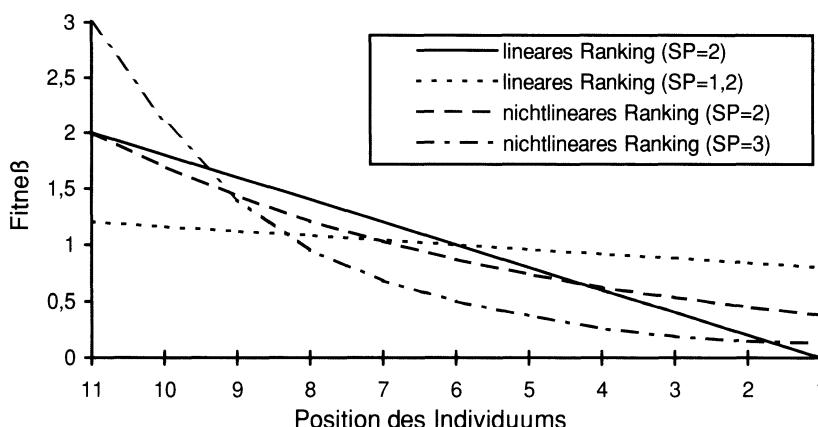


Abb. 3-2. Vergleich der Fitneßfunktionen für lineares und nichtlineares Ranking

In Abb. 3-2 werden die Auswirkungen verschiedener Werte des Selektionsdruckes auf die Fitneßwerte bei linearem und nichtlinearem Ranking grafisch miteinander verglichen.

Beim linearen Ranking mit einem Selektionsdruck von 2 ( $SP=2$ ) erhält das beste Individuum eine Fitneß von 2, für die weiteren Individuen nimmt die Fitneß linear ab. Es ist zu beachten, daß das schlechteste Individuum eine Fitneß von 0 erhält. Damit hat das schlechteste Individuum keine Chance, im nächsten Selektionsschritt ausgewählt zu werden. Diese Fitneß gleich 0 für das schlechteste Individuum tritt nur bei linearem Ranking und einem Selektionsdruck von 2 auf. Bei jedem niedrigeren Selektionsdruck (oder bei nichtlinearem Ranking) hat das schlechteste Individuum immer eine Fitneß größer 0. Dies ist sehr gut beim zweiten Beispiel, lineares Ranking mit einem Selektionsdruck von 1,2 ( $SP=1,2$ ) zu erkennen. Das beste Individuum hat eine Fitneß von 1,2, bei allen weiteren Individuen verringert sich die Fitneß linear. Das schlechteste Individuum hat aber immer noch eine Fitneß von 0,8 und damit eine gute Chance, im nächsten Selektionsschritt ausgewählt zu werden.

Beim Vergleich zwischen linearem und nichtlinearem Ranking unter Verwendung eines Selektionsdruckes von jeweils 2 ist zu erkennen (s. Abb. 3-2), daß die Fitneßwerte für die guten Individuen beim nichtlinearen Ranking etwas niedriger als beim linearen Ranking sind<sup>4</sup>. Für die schlechteren Individuen kehrt sich dies um. Beim nichtlinearen Ranking erhalten damit die schlechteren Individuen eine etwas höhere Fitneß als beim linearen Ranking mit gleichem Selektionsdruck.

Der eigentliche Anwendungsbereich des nichtlinearen Ranking liegt allerdings in der Verwendung eines Selektionsdruckes größer als 2 (beim linearen Ranking kann kein Selektionsdruck größer als 2 verwendet werden). Ein Beispiel dafür ist in Abb. 3-2 für einen Selektionsdruck von 3 gegeben. Die Fitneß für die besten Individuen ist wesentlich größer als bei den nicht so guten Individuen<sup>5</sup>. Allerdings haben die schlechtesten Individuen immer eine Fitneß größer als 0.

Das robuste Verhalten der reihenfolgebasierten Fitneßzuweisung wurde in einer Reihe von Arbeiten (u.a. [BH91], [Why89]) gezeigt und ist die Methode der Wahl für die Fitneßzuweisung.

### 3.1.3 Mehrkriterielle Fitneßzuweisung

Bei der proportionalen und der reihenfolgebasierten Fitneßzuweisung wurde davon ausgegangen, daß die Individuen jeweils nur einen Zielfunktionswert aufweisen. In vielen praktischen Problemen gibt es aber mehrere Kriterien, die zur Beurachtung der Güte eines Individuums betrachtet werden müssen. Erst aus dem Vergleich dieser mehreren Kriterien (daher mehrkriteriell) kann entschieden werden, welches Individuum besser als ein anderes ist. Aus diesen wechselseiti-

<sup>4</sup> Für die jeweils besten Individuen müssen die Fitneßwerte bei gleichem Selektionsdruck identisch sein (ergibt sich aus der Definition des Selektionsdruckes).

<sup>5</sup> In Abb. 3-2 ist z.B. die Fitneß des mittleren Individuums (Individuum 6) bei nichtlinearem Ranking mit  $SP=3$  nur noch 1/6 der des besten Individuums. Bei linearem Ranking mit  $SP=2$  dagegen wäre die Fitneß nur 1/2 der des besten Individuums. Damit wird eine sehr starke Bevorzugung der sehr guten Individuen erreicht.

gen Vergleichen lässt sich dann wie bei einkriteriellen Problemen eine Reihenfolge der Individuen in der Population bilden - mehrkriterielles Ranking. Nachdem diese Reihenfolge erstellt wurde, können die Methoden des einkriteriellen Ranking aus Unterabschn. 3.1.2 verwendet werden, um die Reihenfolge der Individuen in korrespondierende Fitneßwerte umzurechnen.

Bei der mehrkriteriellen Fitneßzuweisung (und damit der mehrkriteriellen Optimierung - *multiobjective optimization*) geht es um die gleichzeitige Minimierung von  $NObj$  Kriterien  $f_r$ , mit  $r = 1, \dots, NObj$ . Die Werte  $f_r$  werden von der Zielfunktion ermittelt, die wiederum von den Variablen der Individuen abhängt (die Entscheidungsvariablen).

Ein einfaches Beispiel soll zur Motivation der folgenden Überlegungen dienen. Bei der Produktion von Gegenständen sollen die Produktionskosten niedrig gehalten und die Gegenstände in kurzer Zeit hergestellt werden. In der Produktionsplanung können verschiedene Lösungen ausgearbeitet werden, die sich in Anzahl und Art der eingesetzten Maschinen sowie der Anzahl der Arbeitskräfte unterscheiden. Zur Bewertung jeder Lösung dienen die Kriterien Produktionskosten  $f_1$  und Herstellungszeit  $f_2$ , die durch die Zielfunktion berechnet werden.

Durch den Vergleich von 2 Lösungen kann entschieden werden, ob eine der beiden Lösungen besser als die andere ist. Der Vergleich von 2 Lösungen kann nach dem folgenden Schema allgemein und für eine beliebige Anzahl von Kriterien durchgeführt werden (Pareto-Ranking bzw. Pareto-Dominanz):

$$\begin{aligned} \forall i \in \{1, \dots, NObj\}, \quad f_i^{Lös_1} &\leq f_i^{Lös_2} \quad \text{und} \quad \exists i \in \{1, \dots, NObj\}, \quad f_i^{Lös_1} < f_i^{Lös_2} \\ \Rightarrow \text{Lösung}_1 &\text{ p} < \text{Lösung}_2 \quad (\text{p} < : \text{ partially less than (plt) - teilweise kleiner als}) \end{aligned} \quad (3-5)$$

Wenn  $\text{Lösung}_1$   $p <$  (teilweise kleiner als)  $\text{Lösung}_2$  ist, dann dominiert  $\text{Lösung}_1$  über  $\text{Lösung}_2$ . In dem hier verwendeten Beispiel heißt dies: wenn für  $\text{Lösung}_1$  Kosten und Zeit kleiner sind als bei  $\text{Lösung}_2$ , so ist  $\text{Lösung}_1$  besser als  $\text{Lösung}_2$ . Es wäre sogar ausreichend, wenn einer der beiden Werte für beide Lösungen gleich ist (gleiche Kosten) und nur der andere kleiner ist (kürzere Zeit).

Wenn allerdings keine der Lösungen über die andere dominiert, dann sind beide Lösungen in Bezug auf die Pareto-Ordnung als gleichwertig zu betrachten. Sich untereinander nicht dominierende Individuen erhalten denselben Rang zugewiesen.

Der Rang eines Individuums in der Population ( $Rang_i$ ) ist davon abhängig, wieviele Individuen ( $NumInd_{dominiert}$ ) dieses Individuum dominieren [Fon95]:

$$Rang_i = 1 + NumInd_{dominiert} \quad (3-6)$$

Alle Lösungen, die bei der Optimierung gefunden werden und von keiner anderen Lösung dominiert werden, stellen die Pareto-optimalen Lösungen (*Pareto-optimal set*) dieses Problems dar (Pareto-Optimalität). Diese Lösungen erhalten alle einen Rang von 1 zugewiesen. Bei jeder Pareto-optimalen Lösung ist es nicht möglich, eines der Kriterien zu verbessern, ohne das ein oder mehrere andere Kriterien schlechter werden.

Bei der Verwendung des einfachen Pareto-Ranking in Gl. 3-5 sind alle Pareto-optimalen Lösungen untereinander gleichwertig. Bei vielen praktischen Problemen ist es aber möglich, eine weitergehende Unterscheidung vorzunehmen. Zur Illustration soll wieder das bereits oben verwendete Beispiel der Produktion von

Gegenständen herangezogen werden. Im Extremfall könnte bei einer Lösung gar nichts produziert werden. Damit sind die Kosten gleich Null und die Herstellungszeit unendlich. Keine andere Lösung könnte mit niedrigeren Kosten die Gegenstände herstellen. Damit würde bei alleiniger Verwendung der Pareto-Dominanz diese Lösung zu den Pareto-optimalen gehören, da sie nicht dominiert werden kann. In einem zweiten Extremfall wird mit höchstem Aufwand in sehr kurzer Zeit produziert, wodurch die Kosten sehr hoch werden. Es dürfte klar sein, daß beide Extremfälle nicht erwünscht sind, obwohl sie zu den nichtdominierten Lösungen gehören.

Um Lösungen vom Pareto-optimalen Set auszuschließen, die außerhalb der vom Anwender gewünschten Ergebnisse liegen, können Ziele (*goals*) für die einzelnen Kriterien eingeführt werden. Erst beim Erreichen dieser Ziele für die einzelnen Kriterien ist eine Lösung akzeptabel. Dieses Vorgehen wird auch als Methode der Ungleichungen (*method of inequalities*, MOI) oder *goal programming* bezeichnet. Die einzelnen Ziele werden als Ungleichungen vorgegeben.

Im hier verwendeten Beispiel der Produktion könnte dies heißen, daß eine obere Grenze für die Kosten und die Herstellungszeit festgelegt wird. Erst wenn beide Ziele gleichzeitig durch eine Lösung erreicht oder unterschritten sind, ist das Ergebnis akzeptabel.

Bei der Einbeziehung der Ziele (*goals*) in das mehrkriterielle Ranking sieht der Vergleich von zwei Lösungen etwas komplexer aus. Die folgenden Annahmen werden getroffen:

$$\begin{aligned} F^{Lös_1} &= \left[ f_1^{Lös_1}, f_2^{Lös_1}, \dots, f_{NObj}^{Lös_1} \right], & F^{Lös_2} &= \left[ f_1^{Lös_2}, f_2^{Lös_2}, \dots, f_{NObj}^{Lös_2} \right] \\ Goals &= \left[ Goal_1, Goal_2, \dots, Goal_{NObj} \right] \end{aligned} \quad (3-7)$$

In den Definitionen wird jeweils der Operator *partially less*  $p <$  aus Gl. 3-5 verwendet. 3 verschiedene Fälle können unterschieden werden.

1. Lösung<sub>1</sub> erfüllt keines der Ziele.

$$\begin{aligned} F^{Lös_1} &> Goals \quad \wedge \quad \text{Lösung}_1 p < \text{Lösung}_2 \\ \Rightarrow \quad &\text{Lösung}_1 \text{ bevorzugt} \end{aligned} \quad (3-8)$$

2. Lösung<sub>1</sub> erfüllt alle Ziele.

$$\begin{aligned} F^{Lös_1} &\leq Goals \quad \wedge \quad (\text{Lösung}_1 p < \text{Lösung}_2 \quad \vee \quad \sim (F^{Lös_2} \leq Goals)) \\ \Rightarrow \quad &\text{Lösung}_1 \text{ bevorzugt} \end{aligned} \quad (3-9)$$

3. Lösung<sub>1</sub> erfüllt einige der Ziele.

$$\begin{aligned} \text{Festlegung: } F_M &= \left[ f_1 f_2 \dots f_m \right], \quad F_K = \left[ f_{m+1} f_{m+2} \dots f_{NObj} \right] \\ \text{Annahme: } F_M^{Lös_1} &> Goals_M \quad \wedge \quad F_K^{Lös_1} \leq Goals_M \\ \left( F_M^{Lös_1} p < F_M^{Lös_2} \quad \vee \quad F_M^{Lös_1} = F_M^{Lös_2} \right) &\wedge \left( F_K^{Lös_1} p < F_K^{Lös_2} \quad \vee \quad \left( F_K^{Lös_2} \leq Goals_K \right) \right) \\ \Rightarrow \quad &\text{Lösung}_1 \text{ bevorzugt} \end{aligned} \quad (3-10)$$

Pareto-Ranking unter Verwendung eines Vektors von Zielen für die einzelnen Kriterien ermöglicht eine verbesserte Bildung der Reihenfolge einer Anzahl von Lösungen im Vergleich zum einfachen Pareto-Ranking.

Durch die mehrkriterielle Optimierung soll eine Anzahl von nichtdominierten Lösungen gefunden werden, das Pareto-optimale Set. Ein normaler Evolutionärer Algorithmus konvergiert allerdings zu einer einzelnen Lösung. Dieser Prozeß wird als genetische Drift bezeichnet. Daher müssen Methoden eingebaut werden, die eine Erhaltung bzw. Vergrößerung der Verschiedenartigkeit der Population erreichen (Verhinderung von *premature convergence*).

Durch die Anwendung einer Fitneßaufteilung (*fitness sharing*) kann einerseits der genetischen Drift entgegen gewirkt werden. Andererseits wirkt der Algorithmus darauf, daß ein größerer Teil der Pareto-optimalen Lösungen ermittelt wird. Die Grundidee ist, daß Individuen in einer bestimmten Nische die verfügbaren Ressourcen untereinander aufteilen müssen. Damit wird die Fitneß eines Individuums um so niedriger, je mehr Individuen in seiner Nähe sind.

In der Anwendung muß die Größe der Nischen festgelegt werden und die Aufteilung der Ressourcen innerhalb einer Nische. Methoden für die Fitneßaufteilung werden u.a. in [HN93], [SD94] und [Fon95] vorgeschlagen.

In diesem Unterabschnitt konnte nur eine kurze Einführung in die mehrkriterielle Fitneßzuweisung im Rahmen von Evolutionären Algorithmen gegeben werden. Für weitergehende Fragen zu einzelnen Verfahren sei auf die entsprechenden Arbeiten verwiesen. Im Rahmen Evolutionärer Algorithmen sind insbesondere die folgenden Arbeiten von Interesse: [Hor97], [Fon95], [ZT98], [HN93], [SD94], [Vel99]. Ein Verzeichnis vieler Arbeiten zur mehrkriteriellen Optimierung mit Evolutionären Algorithmen ist [Coe99] abrufbar. Die Literatur zur mehrkriteriellen Optimierung außerhalb Evolutionärer Algorithmen ist kaum zu überschauen. Als ein Startpunkt kann [Mie99] empfohlen werden.

Bei der Betrachtung der verschiedenen Methoden und Bestandteile der mehrkriteriellen Optimierung sollten klassische Methoden zur Zusammenführung mehrerer Kriterien (*scalarization method*) nicht vergessen werden.

Die bekannteste Methode ist die gewichtete Summe (*weighted sum*). Dabei wird jedem Kriterium ein Gewichtungswert zugeordnet. Durch die lineare Kombination aller gewichteten Kriterien ergibt sich der zusammengefaßte Zielfunktionswert  $F_{ws}$ :

$$F_{ws} = \sum_{r=1}^{NObj} W_r \cdot f_r \quad (3-11)$$

Die gewichtete Summe wird insbesondere eingesetzt, wenn die unterschiedliche Wertigkeit der einzelnen Kriterien bekannt ist bzw. eingeschätzt werden kann. Da dies bei der praktischen Anwendung öfters der Fall ist, findet die gewichtete Summe häufig Anwendung. Zwei Beispiele für den Einsatz der gewichteten Summe werden in Unterabschn. 8.4.4, ab S.229, und Unterabschn. 8.5.3, ab S.242 gegeben.

Wenn eine mehrkriterielles Problem durch einkriterielle Optimierung gelöst wird, erhält man nur eine Punktlösung. Der Vorteil, mehrere gleichwertige Lösungen in Bezug auf einen Zielvektor zu erhalten, geht verloren. An dieser Stelle muß der Anwender entscheiden, ob die einfache Handhabung der gewichteten

Summe oder die Approximation der Pareto-optimalen Lösungen wichtiger für die Lösung des Problems ist.

## 3.2 Selektion

Durch die Selektion werden Individuen entsprechend ihrer Fitneß ausgewählt. Diese Individuen dienen als Elter für die Erzeugung von Nachkommen<sup>6</sup>. Die Selektionswahrscheinlichkeit eines Individuum, ergibt sich aus dem Fitneßwert des Individuum, normalisiert mit der gesamten Fitneß des Selektionspools (Normalisierung der Fitneßwerte).

Die ersten Unterabschnitte gehen auf die 4 gebräuchlichen Selektionsverfahren, Rouletteselektion, *stochastic universal sampling*, Turnierselektion und Truncation-Selektion ein und erläutern sie in ihrer Funktionsweise. In Unterabschn. 3.2.5 wird eine Analyse der einzelnen Selektionsverfahren vorgenommen, auf deren Grundlage in Unterabschn. 3.2.6 ein Vergleich der Verfahren untereinander vorgenommen wird. Aus diesem Vergleich ergeben sich viele Antworten in Bezug auf den Einsatz und die Anwendung der einzelnen Verfahren in der Praxis.

Innerhalb dieses Abschnitts zur Selektion werden einige Begriffe zur Bewertung bzw. zum Vergleich der einzelnen Verfahren benutzt. Eine Erläuterung dieser Begriffe, die sich an [Bak87] und [BT95] orientiert, wird hier vorangestellt:

### Selektionsdruck (selective pressure)

- Wahrscheinlichkeit der Auswahl des besten Individuum, verglichen mit der durchschnittlichen Selektionswahrscheinlichkeit aller Individuen des Selektionspools,

### bias

- absolute Differenz zwischen der normalisierten Fitneß eines Individuum und seiner erwarteten Reproduktionswahrscheinlichkeit,

### spread

- Bereich der möglichen Werte für die Anzahl der Nachkommen eines Individuum,

### Loss of diversity (Verlust an Vielfalt/Mannigfaltigkeit/Diversität)

- Anteil der Individuen des Selektionspools, die während einer Selektionsphase nicht ausgewählt werden (die Individuen/Information, die bei einer vollständigen Ersetzung der Elternpopulation durch die Nachkommenpopulation verloren gehen würden),

---

<sup>6</sup> Eine Selektion von Individuen findet auch an anderen Stellen innerhalb eines Evolutionären Algorithmus statt, z.B.: zur Auswahl von Individuen, die durch Nachkommen ersetzt werden, zur Auswahl unter den Nachkommen, wenn nicht alle Nachkommen in die Population eingefügt werden, zur Auswahl von Individuen, die in andere Unterpopulationen migrieren oder zur Auswahl der Individuen, die bei Konkurrenz an andere Unterpopulationen abgegeben werden.

### Selektionsintensität (selection intensity)

- erwarteter durchschnittlicher Fitneßwert der selektierten Individuen nach einer durchgeführten Selektion (Voraussetzung: normalverteilte Fitneßwerte des Selektionspools),

### Selektionsvarianz (selection variance)

- erwartete Varianz der Fitneßwerte der selektierten Individuen nach einer durchgeführten Selektion (Voraussetzung: normalverteilte Fitneßwerte des Selektionspools).

#### 3.2.1 Rouletteselektion

Das bekannteste Verfahren zur Auswahl von Individuen ist die Rouletteselektion. Die Individuen werden entsprechend ihres absoluten Fitneßwertes (fitneßproportional) aus dem Selektionspool ausgewählt. Die Rouletteselektion wird auch als *stochastic sampling with replacement* bezeichnet [Bak87].

Die Rouletteselektion ist ein stochastisches Verfahren, das nach folgendem Prinzip arbeitet: Die Individuen des Selektionspools werden einzelnen Abschnitten einer Linie zugeordnet. Die Größe eines jeden Abschnittes entspricht der Fitneß des Individuum. Jetzt wird eine Zufallszahl (gleichverteilt im Bereich zwischen null und Länge der Linie) generiert. Das Individuum, in dessen Abschnitt diese Zufallszahl zeigt, wird ausgewählt. Dieser Vorgang wird so oft wiederholt, wie Individuen ausgewählt werden sollen. Dies entspricht einem Roulette, nur daß hier die Größe der Abschnitte der Fitneß der jeweiligen Individuen entspricht.

Die Rouletteselektion hat einen *bias* von null. Es kann aber kein minimaler *spread* garantiert werden<sup>7</sup>.

Die Rouletteselektion ist das bekannteste Verfahren der fitneßproportionalen Selektion. Das optimale Verfahren dagegen ist *stochastic universal sampling*.

#### 3.2.2 Stochastic universal sampling

Beim *stochastic universal sampling* [Bak87] werden die Individuen entsprechend ihres absoluten Fitneßwertes aus dem Selektionspool ausgewählt (fitneßproportionale Selektion). Der Ablauf des Verfahrens ist ähnlich der Rouletteselektion.

Die Individuen des Selektionspools werden einzelnen Abschnitten einer Linie zugeordnet, genau wie bei der Rouletteselektion. Anders ist dagegen, daß hier eine Anzahl von Zeigern gleichmäßig über die Linie verteilt werden, s. Abb. 3-3. Die Anzahl der Zeiger entspricht der Anzahl von Individuen  $N_{Select}$ , die auszuwählen sind. Der Abstand zwischen jeweils benachbarten Zeigern beträgt

---

<sup>7</sup> Im Extremfall kann es passieren, daß in allen Versuchen das schlechteste Individuum bzw. nur schlechte Individuen ausgewählt werden. Die Wahrscheinlichkeit ist zwar gering, dieser Fall wird vom Verfahren der Rouletteselektion aber nicht ausgeschlossen.

$1/NSelect$ . Diese Zeiger werden durch eine Zufallszahl im Bereich  $[0, 1/NSelect]$  vom Nullpunkt der Linie aus zusammenhängend verschoben. Der erste Zeiger befindet sich damit an der Stelle der einen gezogenen Zufallszahl, alle anderen Zeiger in jeweils gleichem Abstand  $1/NSelect$  voneinander hinter dem ersten Zeiger (der 2. Zeiger damit an der Stelle Zufallszahl+ $1/NSelect$  usw.).

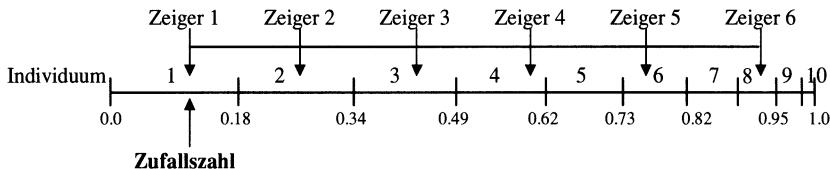


Abb. 3-3. Auswahl der Individuen durch *stochastic universal sampling*

In Abb. 3-3 ist ein Beispiel für die Auswahl von 6 Individuen aus einem Selektionspool von 10 Individuen durch *stochastic universal sampling* dargestellt. Individuum 1 hat die höchste Fitneß (hier 18%), Individuum 10 die kleinste Fitneß (hier 2%). Da 6 Individuen auszuwählen sind, werden 6 Zeiger mit einem Abstand voneinander von  $1/6$  angeordnet. Jetzt wird eine Zufallszahl gleichverteilt im Bereich zwischen 0 und  $1/6$  gezogen (hier 0,1). Diese Zufallszahl bestimmt die Verschiebung der Kette von Zeigern vom Nullpunkt. Das heißt, der erste Zeiger zeigt auf 0,1, der zweite auf  $0,1+1/6=0,267$  und so weiter. Im letzten Schritt werden die Individuen ausgewählt, die unter den Zeigern liegen (hier die Individuen 1, 2, 3, 4, 6 und 8).

Durch dieses Verfahren, das nur die Ziehung einer Zufallszahl beinhaltet, wird ein *bias* von null und ein minimaler *spread* garantiert. *Stochastic universal sampling* ist damit ein optimales Selektionsverfahren in Bezug auf *bias* und *spread*. Für die fitneßproportionale Selektion ist bis heute kein besseres (schnelleres) Verfahren gefunden worden. Damit ist *stochastic universal sampling* das Verfahren der Wahl für die fitneßproportionale Selektion.

### 3.2.3 Turnierselektion

Die Turnierselektion [GD91] führt Turniere zwischen mehreren Individuen zur Auswahl von Individuen durch. Sieger eines Turniers und damit ausgewählt ist das jeweils beste Individuum (höchste Fitneß). Die Auswahl der Individuen, die an einem Turnier teilnehmen, wird gleichmäßig zufällig im Selektionspool durchgeführt. Die Anzahl der Individuen in einem Turnier, die Turniergröße *Tour*, ist der direkte Parameter der Turnierselektion. Die Turniergröße kann Werte von 1 (kein Selektionsdruck, da gleichmäßig zufällige Auswahl) bis zur Anzahl der Individuen im Selektionspool annehmen. Es werden so viele Turniere durchgeführt, wie Individuen auszuwählen sind.

Bei der Turnierselektion wird nur die Rangfolge der Fitneßwerte der Individuen verwendet (wie bei der reihenfolgebasierten Fitneßzuweisung (*ranking*) in

Unterabschn. 3.1.2). Der absolute Wert der Fitneß wird nicht in Betracht gezogen. Daher ist die Turnierselektion keine fitneßproportionale Selektion.

Für die Fitneßzuweisung kann deshalb jede Fitneßfunktion verwendet werden, solange sie eine monotone Zuweisung von Fitneßwerten entsprechend der Rangfolge der Zielfunktionswerte vornimmt, s. Gl. 3-12.

Monotone Fitneßfunktion:

$$\text{Fitneß}(\text{Ind}_i) \leq \text{Fitneß}(\text{Ind}_j) \Leftarrow \alpha \cdot \text{ZFW}(\text{Ind}_i) \leq \alpha \cdot \text{ZFW}(\text{Ind}_j) \quad \forall i, j \in [1 \dots N_{\text{ind}}] \quad (3-12)$$

Minimierung:  $\alpha = -1$ , Maximierung:  $\alpha = 1$

Alle in Abschn. 3.1 angeführten Fitneßfunktionen erfüllen diese Forderung.

### 3.2.4 Truncation-Selektion

Bei der Truncation-Selektion werden nur die besten Individuen ausgewählt. Dazu werden die Individuen entsprechend ihrer Fitneß sortiert. Alle Individuen mit einer Fitneß, die größer als eine festgelegte Schwelle ist, werden ausgewählt. Die anderen Individuen werden verworfen, sie bekommen keine Chance zur Produktion von Nachkommen. Die Schwelle wird als Anteil der Individuen des Selektionspools definiert, die ausgewählt werden und heißt Truncation-Schwelle *Trunc* (Abschneideschwelle). In Abb. 3-4 wird dieser Prozeß nochmals verdeutlicht. Entsprechend seinem Rang im sortierten Selektionspool erhält ein Individuum eine Selektionswahrscheinlichkeit von 1 (gute Individuen) oder 0 (schlechte Individuen).

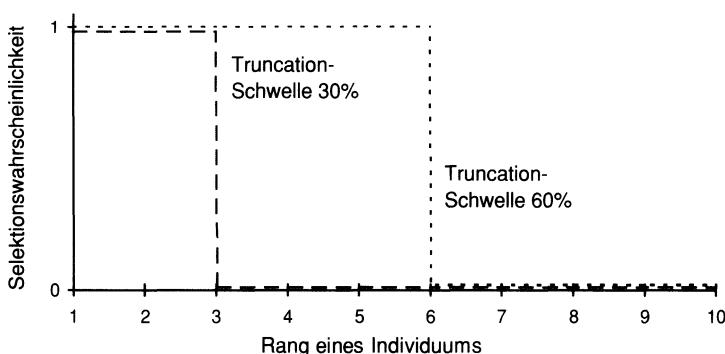


Abb. 3-4. Prinzip der Truncation-Selektion: Selektionswahrscheinlichkeit eines Individuums in Abhängigkeit seines Ranges

Auch bei der Truncation-Selektion wird, ähnlich zur Turnierselektion, nur die Rangfolge der Fitneßwerte der Individuen verwendet. Der absolute Wert der Fit-

neß wird nicht in Betracht gezogen. Deshalb kann für die Fitneßzuweisung jede Fitneßfunktion verwendet werden, solange sie eine monotone Zuweisung von Fitneßwerten entsprechend der Reihenfolge der Zielfunktionswerte vornimmt, s. Gl. 3-12.

Im Vergleich zu den bisherigen Selektionsmethoden, die Prinzipien der natürlichen Selektion modellieren, ist die Truncation-Selektion ein künstliches Selektionsverfahren. Es wird vor allem von Züchtern für sehr große Populationen bzw. die Massenselektion angewendet.

### 3.2.5 Analyse der Selektionsverfahren

Bisher wurden die einzelnen Selektionsverfahren nur in ihrem direkten Ablauf vorgestellt. Für eine Bewertung bzw. Analyse der Selektionsverfahren werden im folgenden die Größen Selektionsintensität (*SelInt*), *Loss of Diversity* (*LossDiv*) und Selektionsvarianz (*SelVar*) verwendet.

In diesem Unterabschnitt wird die Abhängigkeit dieser Bewertungsgrößen von dem jeweiligen direkten Selektionsparameter (Selektionsdruck bei den fitneßproportionalen Selektionsverfahren, Turniergröße bei der Turnierselektion, Truncation-Schwelle bei der Truncation-Selektion) dargestellt. Diese Ergebnisse bilden die Grundlage für den Vergleich der Verfahren untereinander in Unterabschn. 3.2.6.

Die Angaben in diesem und dem folgenden Unterabschnitt basieren auf [BT95] und [Bli97]. Grundlage für die hier gemachten Angaben ist, daß die verwendeten Daten (Zielfunktionswerte der Individuen einer Population) einer bivariaten Normalverteilung unterliegen. Dies kann für reale Daten nur selten gewährleistet werden. Daher sind die hier gemachten Angaben von ihrer qualitativen Seite aus zu betrachten (und nicht von ihrer exakten quantitativen). Trotz dieser Einschränkungen sind die Ergebnisse sehr eingängig und verständlich und ermöglichen einen Vergleich der Selektionsverfahren untereinander. Viele Fragen über die Einsatzbereiche der einzelnen Verfahren (wann welches Selektionsverfahren eingesetzt werden soll und wie) lassen sich daraus beantworten.

Für die Analyse der Selektionsverfahren werden die Gleichungen aufgeführt, die den Zusammenhang zwischen den Bewertungsgrößen und dem Selektionsparameter wiedergeben. Meist ist dies mit einer exakten Gleichung möglich (unter den oben erwähnten Voraussetzungen und Einschränkungen), einige Gleichungen können allerdings nur als Approximation ( $\approx$ ) angegeben werden. Für den Anwender am interessantesten sind die aus den Gleichungen erzeugten Diagramme, welche die entsprechenden Zusammenhänge grafisch veranschaulichen.

Für weitergehende Informationen sowie die Herleitung und Berechnung der hier angegebenen Gleichungen sei auf [BT95] und [Bli97] verwiesen.

#### ***Fitneßproportionale Selektion mit linearem Ranking***

Die fitneßproportionalen Selektionsverfahren (Rouletteselektion und *stochastic universal sampling*) müssen immer im Zusammenhang mit der verwendeten Fitneßzuweisung betrachtet werden. Im folgenden werden die Zusammenhänge für

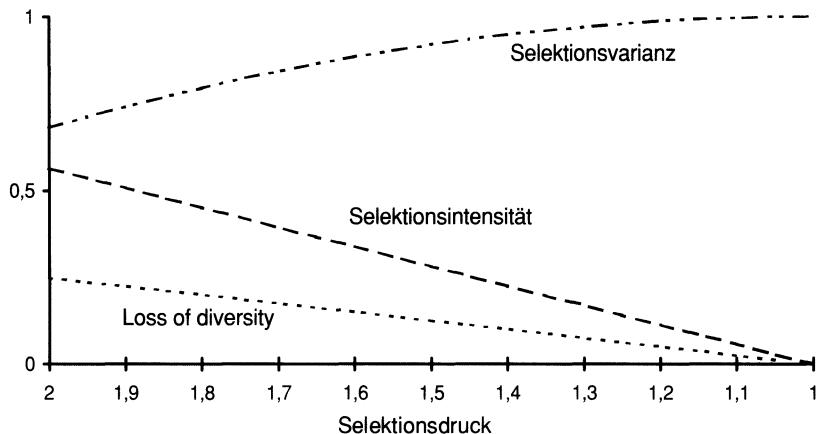
eine fitneßproportionale Selektion mit linearem Ranking (lineare Ranking-Selektion) aufgeführt. Der Parameter der fitneßproportionalen Selektion ist der Selektionsdruck ( $SP$ ).

$$SelInt_{LinRank}(SP) = (SP - 1) \cdot \frac{1}{\sqrt{\pi}} \quad (3-13)$$

$$LossDiv_{LinRank}(SP) = \frac{(SP - 1)}{4} \quad (3-14)$$

$$SelVar_{LinRank}(SP) = 1 - \frac{(SP - 1)^2}{\pi} = 1 - SelInt_{LinRank}(SP)^2 \quad (3-15)$$

Die Eigenschaften der linearen Ranking-Selektion sind in Abb. 3-5 in Abhängigkeit des Selektionsdruck grafisch veranschaulicht.



**Abb. 3-5.** Eigenschaften der linearen Ranking-Selektion (fitneßproportionale Selektion mit linearem Ranking als Fitneßzuweisung)

### Turnierselektion

Der Parameter der Turnierselektion ist die Turniergröße ( $Tour$ ).

$$SelInt_{Turnier}(Tour) \approx \sqrt{2 \cdot \left( \ln(Tour) - \ln(\sqrt{4,14 \cdot \ln(Tour)}) \right)} \quad (3-16)$$

$$LossDiv_{Turnier}(Tour) = Tour^{\frac{-1}{Tour-1}} - Tour^{\frac{-Tour}{Tour-1}} \quad (3-17)$$

$$SelVar_{Turnier}(Tour) \approx \frac{0,918}{\ln(1,186 + 1,328 \cdot Tour)} \quad (3-18)$$

Diese Eigenschaften der Turnierselektion sind in Abb. 3-6 in Abhängigkeit des Parameters der Turnierselektion, der Turniergröße, grafisch veranschaulicht.

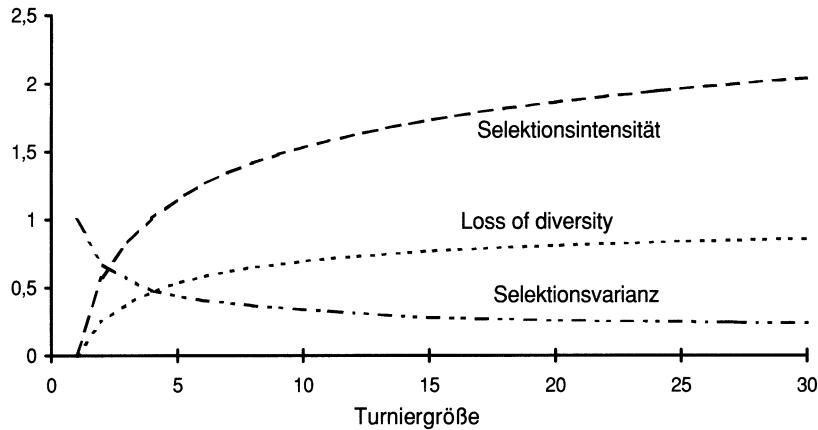


Abb. 3-6. Eigenschaften der Turnierselektion

### Truncation-Selektion

Der Parameter der Truncation-Selektion ist die Truncation-Schwelle (*Trunc*).

$$SelInt_{Truncation}(Trunc) = \frac{1}{Trunc} \cdot \frac{1}{\sqrt{2 \cdot \pi}} \cdot e^{-\frac{f_c^2}{2}} \quad (3-19)$$

$$LossDiv_{Truncation}(Trunc) = 1 - Trunc \quad (3-20)$$

$$SelVar_{Truncation}(Trunc) = 1 - SelInt_{Truncation}(Trunc) \cdot \\ (SelInt_{Truncation}(Trunc) - f_c) \quad (3-21)$$

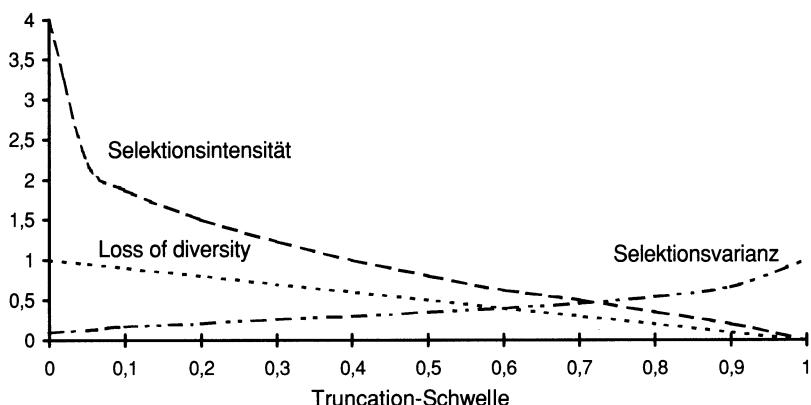


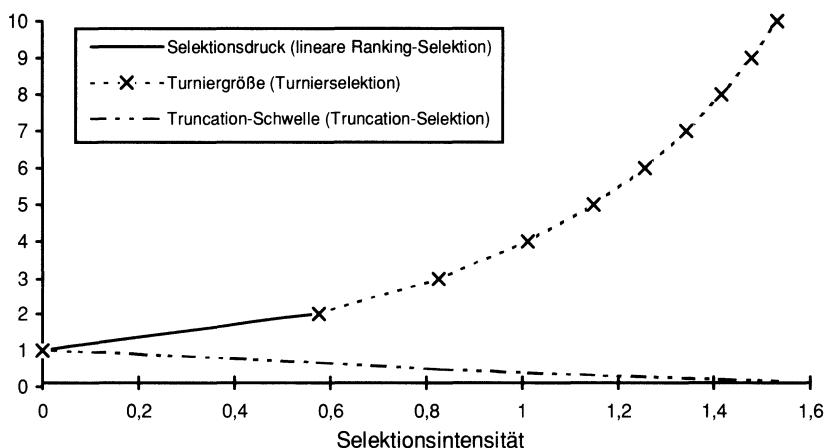
Abb. 3-7. Eigenschaften der Truncation-Selektion

Die Eigenschaften der Truncation-Selektion sind in Abb. 3-7 in Abhängigkeit des Parameters der Truncation-Selektion, der Truncation-Schwelle, grafisch veranschaulicht. Dieselben Resultate wurden auf einem anderen Weg in [CK70] hergeleitet.

### 3.2.6 Vergleich der Selektionsverfahren

Der Vergleich der verschiedenen Selektionsverfahren aus [BT95] wurde bei der Beschreibung der einzelnen Verfahren schon mehrfach erwähnt.<sup>8</sup> Hier sollen die Eigenschaften der verschiedenen Selektionsverfahren miteinander verglichen werden. Abschließend sollen daraus schlußfolgernd Hinweise zur Anwendung der Verfahren abgeleitet werden.

Die verschiedenen Selektionsverfahren können miteinander verglichen werden, wenn die verschiedenen Eigenschaften auf einen gemeinsamen Parameter zurückzuführen sind. Dazu bieten sich die Selektionsintensität und der Selektionsdruck an. In der Literatur wird die Selektionsintensität bevorzugt und deshalb auch hier genutzt.



**Abb. 3-8.** Abhängigkeit der Selektionsparameter von der Selektionsintensität

In Abb. 3-8 werden die jedem Selektionsverfahren eigenen Parameter (Selektionsdruck für lineare Ranking-Selektion, Truncation-Schwelle für Truncation-Selektion und Turniergröße für Turnierselektion) in Abhängigkeit von der Selektionsintensität dargestellt. Durch den bekannten Zusammenhang zwischen Selektionsintensität und Selektionsdruck könnte diese Abhängigkeit auch gegen-

<sup>8</sup> In [MSV95] wurde auch eine Analyse der Truncation-Selektion unternommen. Diese ist im Vergleich zu den Ergebnissen in [BT95] aber stark eingeschränkt.

über dem Selektionsdruck für die verschiedenen Verfahren dargestellt werden. An dieser Stelle sei darauf hingewiesen, daß die Turnierselektion nur diskrete Werte zuläßt.

Obwohl die Parameter der Selektionsverfahren auf die gleiche Selektionsintensität zurückgeführt werden können, zeigen sich Unterschiede beim Vergleich der weiteren Eigenschaften der Verfahren. Die Abbildungen 3-9 und 3-10 vergleichen den *Loss of diversity* bzw. die Selektionsvarianz der Verfahren in Abhängigkeit von der Selektionsintensität.

Aus Abb. 3-9 ist zu sehen, daß für ein und dieselbe Selektionsintensität bei der Truncation-Selektion ein höherer *Loss of diversity* auftritt, als dies bei der Turnierselektion und der linearen Ranking-Selektion der Fall ist. Bei der Truncation-Selektion ist die Wahrscheinlichkeit größer, daß nicht so gute Individuen durch bessere Individuen ersetzt werden, als dies bei den anderen Selektionsverfahren der Fall ist.<sup>9</sup>

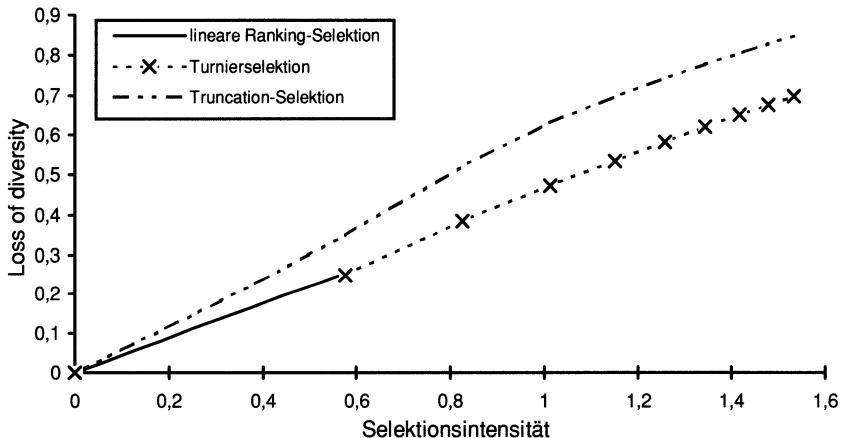


Abb. 3-9. Abhängigkeit des *Loss of diversity* von der Selektionsintensität

Turnierselektion und lineare Ranking-Selektion sind in ihrem Verhalten bezüglich des *Loss of diversity* ähnlich zueinander. An den Randpunkten des Definitionsbereiches der linearen Ranking-Selektion sind beide Verfahren gleich. Für die anderen Bereiche ist jeweils nur eines der Verfahren definiert (komplementäres Verhalten beider Verfahren).

Für dieselbe Selektionsintensität kommt es bei der Truncation-Selektion zu einer deutlich kleineren Selektionsvarianz als bei der linearen Ranking-Selektion

<sup>9</sup> Dieser höhere *Loss of diversity* kann leicht aus der Definition der Truncation-Selektion nachvollzogen werden. Die schlechten Individuen werden nicht ausgewählt und erhalten damit gar keine Chance zur Weitergabe von Information. Bei den anderen Verfahren dagegen hat fast jedes Individuum eine Chance zur Auswahl, auch wenn sie noch so klein ist.

bzw. der Turnierselektion, s. Abb. 3-10. Dies korrespondiert mit den Ergebnissen aus Abb. 3-9, in welcher der *Loss of diversity* verglichen wurde. Ein direkter Vergleich zwischen linearer Ranking-Selektion und Turnierselektion ist nur an den Randpunkten des Definitionsbereiches der linearen Ranking-Selektion möglich. Dort verhalten sich beide Verfahren identisch.<sup>10</sup>

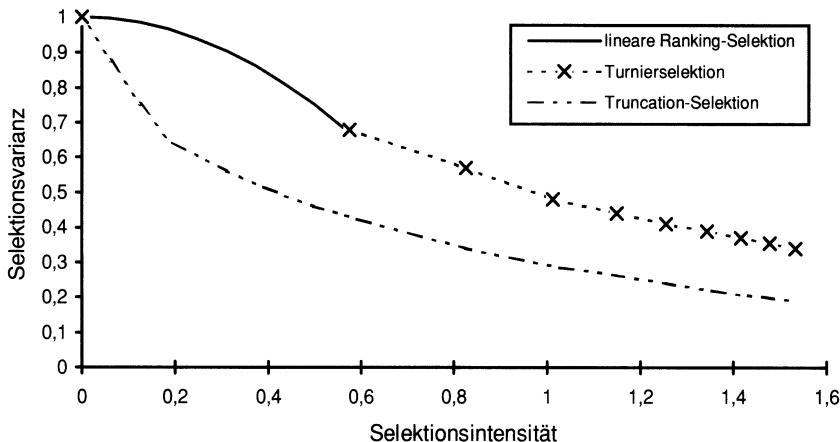


Abb. 3-10. Abhängigkeit der Selektionsvarianz von der Selektionsintensität

In [BT95] wurde bewiesen, daß sich lineare Ranking-Selektion für einen Selektionsdruck von 2 und Turnierselektion für eine Turniergröße von 2 zueinander identisch verhalten (für beide Verfahren ergeben sich eine identische Selektionsintensität von  $0,56 = 1/\sqrt{\pi}$  ).

Für den Einsatz ergibt sich aus dem komplementären Definitionsbereich von linearer Ranking-Selektion und Turnierselektion, daß bei einer Selektionsintensität  $\leq 1/\sqrt{\pi}$  lineare Ranking-Selektion zum Einsatz kommt, bei einer Selektionsintensität  $\geq 1/\sqrt{\pi}$  Turnierselektion. Mit diesen beiden Verfahren kann damit eine Selektionsintensität und ähnliches Verhalten in einem großen Bereich erreicht werden. Bei Verwendung nur eines der beiden Verfahren wäre dies nicht möglich.

Welche Aussagen sind bezüglich des Einsatzes von Truncation-Selektion oder der Kombination lineare Ranking-Selektion und Turnierselektion zu machen? Unter der Voraussetzung, daß ein Selektionsverfahren, das bei gleicher Selektionsintensität eine höhere Selektionsvarianz und/oder einen geringeren *Loss of*

<sup>10</sup> Wenn allerdings die Approximationsgleichung für die Selektionsvarianz der Turnierselektion, Gl. 3-18, auch für den Bereich zwischen Turniergröße eins und zwei betrachtet würde, ergäbe sich, daß die Turnierselektion eine niedrigere Selektionsvarianz als die lineare Ranking-Selektion hätte. In Abb. 3-10 ist dies nicht dargestellt.

*diversity* aufweist, günstiger ist<sup>11</sup>, wäre die Kombination lineare Ranking-Selektion und Turnierselektion zu bevorzugen. Beide Verfahren haben eine höhere Selektionsvarianz und einen niedrigeren *Loss of diversity* als die Truncation-Selektion.

Truncation-Selektion hat bei Problemen Vorteile, bei denen die Suche stärker fokussiert werden soll, ohne einen zu hohen Selektionsdruck ausüben zu müssen. Wie schon weiter oben erwähnt, bekommen bei der Truncation-Selektion die schlechten Individuen gar keine Chance, ausgewählt zu werden. Dies führt, wie in den Abbildungen 3-9 und 3-10 gezeigt, bei gleicher Selektionsintensität zu einem schnelleren Verlust an Information, gleichzeitig aber auch zu einer stärkeren Einengung des Suchbereiches auf vielversprechende Regionen.

Wenn zwei Evolutionäre Algorithmen, einer unter Verwendung der Truncation-Selektion, der andere mit der linearen Ranking-Selektion bzw. Turnierselektion, ein und dasselbe Problem mit ansonsten identischen Operatoren und Parametern lösen können, dann ist der Evolutionäre Algorithmus unter Verwendung der Truncation-Selektion schneller.

In mehreren Arbeiten von MÜHLENBEIN und SCHLIERKAMP-VOOSEN (u.a. [MSV93a], [Müh94]) wurden die Vorteile des Einsatzes der Truncation-Selektion bei der Parameteroptimierung gezeigt. Der von ihnen eingesetzte *Breeder Genetic Algorithm* verwendet die Truncation-Selektion standardmäßig.

### 3.3 Rekombination

Durch die Rekombination werden aus zwei oder mehr Elternindividuen die Nachkommen gebildet. Dies geschieht durch Kombination der Variablenwerte der Eltern. In Abhängigkeit der Repräsentation und der zu lösenden Probleme kommen verschiedene Verfahren zum Einsatz.

Im Unterabschn. 3.3.1 wird die diskrete Rekombination erläutert, die sich bei der Parameteroptimierung auf alle Repräsentationen von Variablen anwenden lässt. Im Unterabschn. 3.3.2 werden Verfahren speziell für reelle Variablen vorgestellt und im Unterabschn. 3.3.3 für binäre Variablen.

Verfahren für ganzzahlige Variablen lassen sich aus den Verfahren für reelle Variablen ableiten. Die Verfahren für binäre Variablen, die alle eingeschränkte Spezialfälle der diskreten Rekombination sind, lassen sich natürlich auch für reelle und ganzzahlige Variablen verwenden.

Für Probleme der kombinatorischen Optimierung werden in Unterabschn. 3.3.4 spezielle Rekombinationsoperatoren vorgestellt, die für die Anwendung auf diese Problemklassen besonders geeignet sind.

---

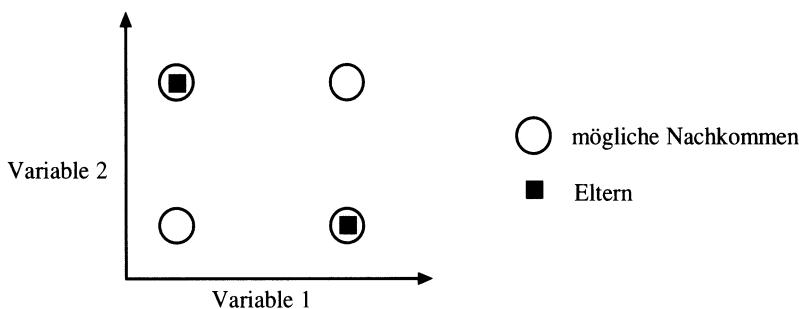
<sup>11</sup> Eine höhere Selektionsvarianz und/oder ein kleinerer *Loss of diversity* bei gleicher Selektionsintensität bedeuten eine stärkere Breitensuche bei gleicher Bevorzugung der besten Individuen.

### 3.3.1 Diskrete Rekombination

Diskrete Rekombination führt einen Austausch der Variablen zwischen den Elterindividuen zur Bildung der Nachkommen durch. Für jede Variablenposition wird entschieden, von welchem Elter der Variablenwert verwendet wird.

$$\begin{aligned} Var_i^N &= Var_i^{E_1} \cdot a_i + Var_i^{E_2} \cdot (1-a_i) \quad i \in (1, 2, \dots, N_{\text{var}}), \\ a_i &\in \{0,1\} \text{ mit gleicher Wahrscheinlichkeit,} \\ a_i &\text{ für jedes } i \text{ neu bestimmt} \end{aligned} \quad (3-22)$$

Die diskrete Rekombination kann Nachkommen auf den Eckpunkten eines Hyperwürfels erzeugen, der durch die Variablenwerte der Eltern aufgespannt wird. Eine deutsche Benennung als Eck-Rekombination würde deutlicher die Funktionsweise zum Ausdruck bringen. Abbildung 3-11 zeigt diesen geometrischen Effekt der diskreten Rekombination.



**Abb. 3-11.** Mögliche Positionen der Nachkommen nach einer diskreten Rekombination in Bezug auf die Position der Eltern

Eine Erweiterung der hier beschriebenen diskreten Rekombination mit zwei Eltern auf die Verwendung einer höheren Anzahl von Eltern ist offensichtlich. Für jede Variablenposition wird die Variable des Nachkommen von einem der Eltern ausgewählt. Dabei haben alle Eltern dieselbe Wahrscheinlichkeit, Variablen zu einem Nachkommen beizusteuern.

Diskrete Rekombination kann auf alle Repräsentationen der Variablen (reell, ganzzahlig, binär, Symbole) angewendet werden.

### 3.3.2 Rekombination reeller und ganzzahliger Variablen

Die in diesem Unterabschnitt beschriebenen Rekombinationsverfahren können bei der Rekombination von Individuen mit reellen und ganzzahligen Variablen Anwendung finden. Die Algorithmen sind in ihrem Ablauf für reelle und ganzzahlige Variablen identisch. Bei der Anwendung auf ganzzahlige Variablen muß zusätzlich gesichert sein, daß die Nachkommen nur ganzzahlige Variablen ent-

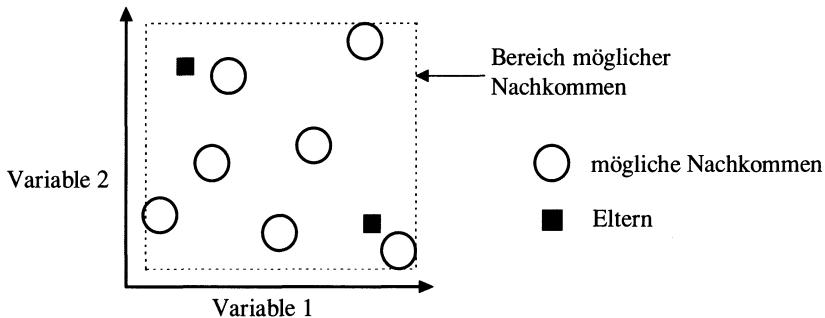
halten. Dies kann durch ein Runden nach der Anwendung des Operators erfolgen.

### Intermediäre Rekombination

Intermediäre Rekombination (*intermediate recombination*) [MSV93a] kombiniert die Variablen der Eltern nach dem folgenden Verfahren:

$$\begin{aligned} Var_i^N &= Var_i^{E_1} \cdot a_i + Var_i^{E_2} \cdot (1-a_i) \quad i \in (1, 2, \dots, Nvar), \\ a_i &\in [-d, 1+d] \text{ gleichverteilt, } d = 0.25, \\ a_i &\text{ für jedes } i \text{ neu bestimmt} \end{aligned} \quad (3-23)$$

Der Skalierungsfaktor  $a$  wird gleichverteilt im Intervall  $[-d, 1+d]$  für **jede Variable** neu gewählt. Mit der Größe des Wertes  $d$  wird festgelegt, wie groß der Bereich ist, in dem die Variablen der Nachkommen liegen können. Für einen Wert von  $d = 0$  ist der mögliche Bereich der Nachkommen genau so groß wie der Bereich, der von den Eltern aufgespannt wird. Da die meisten Variablen der Nachkommen aber nicht auf den Grenzen des Bereiches liegen, kommt es allein durch die Anwendung der intermediären Rekombination zu einer schrittweisen Verkleinerung des Variablenbereiches. Dies kann durch ein größeres Intervall für den Wert von  $d$  verhindert werden. Mit einem Wert von  $d = 0,25$  wird erreicht, daß der Bereich, der durch die erzeugten Nachkommen aufgespannt wird (über viele Versuche gesehen), genau so groß ist, wie der Bereich der Eltern.



**Abb. 3-12.** Mögliche Positionen der Nachkommen nach einer intermediären Rekombination in Bezug auf die Position der Eltern

Die intermediäre Rekombination kann Nachkommen in einem Hyperwürfel erzeugen, der etwas größer ist, als der durch die Variablenwerte der Eltern aufgespannte Hyperwürfel. Wieviel größer der Hyperwürfel der Nachkommen ist, hängt vom Wert des Parameters  $d$  ab. Eine deutsche Benennung als Raum-Rekombination würde deutlicher die Funktionsweise dieses Operators zum Ausdruck bringen. Abbildung 3-12 veranschaulicht den geometrischen Effekt der intermediären Rekombination.

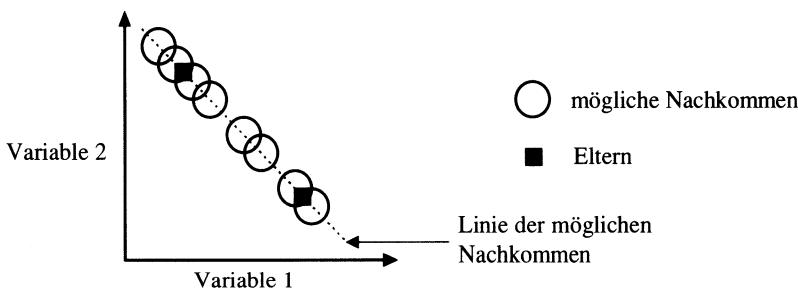
### Linien-Rekombination

Linien-Rekombination [MSV93a] ist der intermediären Rekombination ähnlich. Im Unterschied zur intermediären Rekombination wird bei der Linien-Rekombination nur ein und derselbe Skalierungsfaktor  $a$  für alle Variablen verwendet, um den Anteil der Variablenwerte der beiden Eltern zu bestimmen.

$$\begin{aligned} Var_i^N &= Var_i^{E_1} \cdot a_i + Var_i^{E_2} \cdot (1-a_i) \quad i \in (1, 2, \dots, N_{var}), \\ a_i &\in [-d, 1+d] \text{ gleichverteilt, } d=0.25, \\ a_i &\text{ für alle } i \text{ gleich} \end{aligned} \quad (3-24)$$

Für daß Intervall, in dem  $a$  gleichverteilt ausgewählt wird, gelten die gleichen Aussagen, wie für die intermediäre Rekombination.

Linien-Rekombination kann Nachkommen auf einer Linie generieren, die durch die Variablenwerte der Eltern festgelegt wird. Die Nachkommen können nicht nur auf der Strecke zwischen den Eltern liegen, sondern auch außerhalb der Strecke zwischen den Eltern. Wie bei der intermediären Rekombination wird die Ausdehnung dieser Strecke durch den Wert von  $d$  festgelegt. In Abb. 3-13 wird der geometrische Effekt dieses Operators in zwei Dimensionen dargestellt.



**Abb. 3-13.** Mögliche Positionen der Nachkommen nach einer Linien-Rekombination in Bezug auf die Position der Eltern

Ein Spezialfall der Linien-Rekombination (und damit auch der intermediären Rekombination) ist die Mittelwertrekombination. Bei dieser wird der Wert von  $a$  gleich 0.5 gesetzt. Die Variablen der Nachkommen bilden durch diese Festlegung von  $a$  die Mittelwerte der Variablen der Eltern. In Abb. 3-13 würden die Nachkommen dadurch genau in der Mitte zwischen den Eltern liegen. Durch die eindeutige Festlegung von  $a$  fehlt bei der Mittelwertrekombination das stochastische Element. Statt dessen handelt es sich um einen deterministischen Operator, eben die Mittelwertbildung. Verwendet wird dieser Operator im Bereich der Evolutionsstrategien zur Rekombination der Strategieparameter (Größe der Mutations schrittweiten) und wird dort als intermediäre Rekombination bezeichnet.

### Erweiterte Linien-Rekombination

Die erweiterte Linien-Rekombination [Müh94] erzeugt die Nachkommen auf einer Linie, die durch die Variablenwerte der Eltern festgelegt wird. Im Gegensatz zur Linien-Rekombination beschränkt sich die erweiterte Linien-Rekombination nicht auf eine Strecke, die zwischen den Eltern sowie ein Stück außerhalb liegt. Bei der erweiterten Linien-Rekombination definieren die Eltern nur die Linie, auf der Nachkommen erzeugt werden. Die Länge des Bereiches, in dem die möglichen Nachkommen liegen, wird in Abhängigkeit des Definitionsbereichs der jeweiligen Variablen festgelegt. In diesem Bereich werden die Nachkommen nicht gleichverteilt erzeugt, sondern Nachkommen werden mit einer hohen Wahrscheinlichkeit in der Nähe und nur mit einer geringen Wahrscheinlichkeit weit entfernt vom ersten Elter erzeugt.

Wenn die Fitneß der Eltern verfügbar ist, kann diese Information dazu benutzt werden, daß die Nachkommen um den besseren Elter erzeugt werden. Außerdem ist es möglich, Nachkommen öfters in der Richtung vom schlechteren zum besseren Elter zu erzeugen (gerichtete Rekombination).

Die erweiterte Linien-Rekombination benutzt zur Berechnung der Variablenwerte der Nachkommen das folgende Schema:

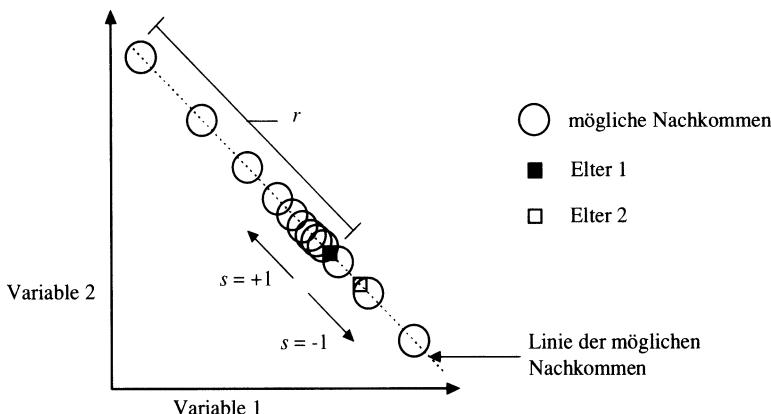
$$\begin{aligned} Var_i^N &= Var_i^{E_1} + s \cdot r_i \cdot a \cdot \frac{Var_i^{E_2} - Var_i^{E_1}}{\|Var_i^{E_1} - Var_i^{E_2}\|} \quad i \in \{1, 2, \dots, Nvar\}, \\ a &= 2^{-k \cdot u}, \quad k: \text{Präzisionsparameter, } u \in [0, 1] \text{ gleichverteilt,} \\ r_i &= r \cdot \text{Domain,} \quad r: \text{Schrittweitenbereich,} \\ &\quad \text{Domain: Definitionsbereich der Variable} \\ s &\in \{-1, +1\}, \quad \text{gleichverteilt: ungerichtete Rekombination} \\ &\quad +1 \text{ mit Wahrscheinlichkeit } > 0,5: \text{gerichtete Rekombination} \end{aligned} \tag{3-25}$$

Die erweiterte Linien-Rekombination benutzt dasselbe Schema zur Festlegung der Schrittweite, das von der in Unterabschn. 3.4.1 beschriebenen Mutation für reelle (und ganzzahlige) Variablen verwendet wird. Der Parameter  $a$  bestimmt die relative Schrittweite, durch den Parameter  $r$  wird der mögliche Bereich der Schrittweite vorgegeben und der Parameter  $s$  bestimmt die Richtung, in die der Rekombinationsschritt durchgeführt wird.

Abbildung 3-14 unternimmt den Versuch der Darstellung des Effekts der erweiterten Linien-Rekombination.

Der Parameter  $k$  bei der Berechnung von  $a$  bestimmt die Präzision, mit der Schritte erzeugt werden können. Je größer  $k$  ist, um so kleinere Schritte können erzeugt werden. Für alle Werte von  $k$  ist der maximale Wert von  $a = 1$  ( $u = 0$ ). Der minimale Wert von  $a$  ist abhängig von  $k$  und beträgt  $a = 2^{-k}$  ( $u = 1$ ). Typische Werte für den Präzisionsparameter  $k$  liegen im Bereich von 4 bis 20.

Die Festlegung des Parameters  $r$  mit 10% des Definitionsbereiches der Variablen sollte als robuster Startwert verstanden werden. Je nach Art der angegebenen Definitionsbereiche der Variablen sollte eine Anpassung dieses Parameters an die Aufgabenstellung erfolgen. Außerdem kann durch Verkleinerung dieses Parameters während eines Laufs die Suche weiter eingeschränkt werden.



**Abb. 3-14.** Mögliche Positionen der Nachkommen nach einer erweiterten Linienrekombination in Bezug auf die Position der Eltern und den Definitionsbereich der Variablen

Wenn der Parameter  $s$  mit gleicher Wahrscheinlichkeit -1 oder +1 sein kann, dann findet keine gerichtete Erzeugung von Nachkommen statt. Wird  $s$  dagegen mit einer höheren Wahrscheinlichkeit zu +1 gesetzt, so werden die Nachkommen öfters in Richtung vom schlechteren zum besseren Elter erzeugt (vorausgesetzt, daß der erste Elter der bessere Elter ist).

Es sei nochmals darauf hingewiesen, daß die in diesem Unterabschnitt vorgestellten Verfahren zur Rekombination von Individuen sich ohne Probleme auf Individuen mit ganzzahligen Variablen anwenden lassen. Ob die Einhaltung des diskreten Charakters der ganzzahligen Variablen bei der Berechnung des Skalierungsfaktors bzw. der Schrittweite (Parameter  $a$ ) oder durch ein nachfolgendes Runden sichergestellt wird, ist dem Anwender überlassen bzw. hängt von der Art der Implementierung ab. Der Berechnungsablauf und die Eigenschaften der Verfahren bleiben übertragen erhalten.

### 3.3.3 Rekombination binärer Variablen (crossover)

Die in diesem Unterabschnitt beschriebenen Rekombinationsverfahren können bei der Rekombination von Individuen mit binären Variablen Anwendung finden. Für die Bezeichnung dieser Verfahren wird im allgemeinen der Begriff *crossover* verwendet, der deswegen in diesem Unterabschnitt zur Anwendung kommt.

Bei der Rekombination binärer Variablen kommt es nur zu einem Austausch der einzelnen Variablen zwischen den Individuen. Die Verfahren unterscheiden sich darin, in wie viele Stücke die Variablen der Individuen vor dem Austausch zerlegt werden.

### **Multi-point crossover**

Beim *multi-point crossover* wird eine Anzahl von Schnittpunkten gleichverteilt zufällig ausgewählt. Die Schnittpunkte werden der Größe nach sortiert. Danach werden die Variablen innerhalb aufeinanderfolgender Schnittpunkte zwischen den Elternindividuen zur Bildung zweier Nachkommen ausgetauscht. Abbildung 3-15 veranschaulicht diesen Prozeß.

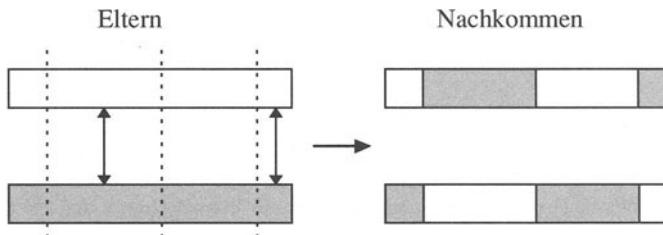


Abb. 3-15. Ablauf des *multi-point crossover*

### **Single-point crossover**

Wenn beim *multi-point crossover* nur eine Schnittposition festgelegt wird, ergibt sich der Spezialfall des *single-point crossover*. In Abb. 3-16 ist dieser Vorgang illustriert.

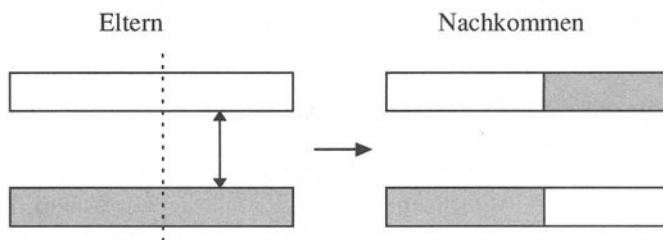


Abb. 3-16. Ablauf des *single-point crossover*

Entsprechend kann durch Festlegung von zwei Schnittpunkten *double-point crossover* bzw. *two-point crossover* durchgeführt werden.

### **Uniform crossover**

Beim *multi-point crossover* wurde eine Anzahl von Schnittpunkten festgelegt, zwischen denen die Variablen der Eltern ausgetauscht werden. *Uniform crossover* [Sys89] erweitert dieses Schema soweit, daß zwischen allen Positionen ein Austausch der Variablen zwischen den Eltern stattfinden kann. Das heißt, für jede einzelne Variable eines Nachkommen wird festgelegt, welcher Elter diese Variable liefert. Dieses Schema ist identisch zu dem der diskreten Rekombination, die in Unterabschn. 3.3.1 vorgestellt wurde.

### **Shuffle crossover**

*Shuffle crossover* [CES89] (*shuffle* (engl.): mischen) ist dem *single-point* und *uniform crossover* sehr ähnlich. Vor dem crossover werden die Variablen in beiden Eltern zufällig gemischt. Danach wird ein *single-point crossover* durchgeführt. Nach dem crossover werden die Variablen wieder auf ihre ursprünglichen Positionen „entmischt“. Dadurch wird eine Bevorzugung bestimmter Positionen verhindert, da die Variablen vor jeder Rekombination aufs Neue zufällig gemischt werden.

### **Crossover mit reduced surrogate**

Der Operator *reduced surrogate* [Boo87] beschränkt *crossover* soweit, daß, wenn möglich, neue Individuen produziert werden. Dies wird dadurch erreicht, daß die Positionen der Schnittpunkte auf solche Positionen beschränkt werden, in denen sich die Variablen der Eltern unterscheiden.

### **Zusammenfassung crossover**

Der Grundgedanke hinter *multi-point crossover* und vielen anderen *crossover* Varianten ist, daß die Variablen eines Individuums, die am meisten zur Performance beitragen, nicht in einem zusammenhängenden Teil eines Individuums enthalten sind [Boo87]. Außerdem wird der auseinanderreißende Effekt des *multi-point crossover*, im Vergleich zu *single-point* und *double-point crossover*, als verstärkte Breitensuche im Suchraum angesehen, anstatt sehr zeitig in der Suche sehr gute Individuen zu bevorzugen. Dadurch soll die Suche robuster werden [SDJ91b].

*Uniform crossover*, ähnlich wie *multi-point crossover*, beansprucht, daß die Bevorzugung bestimmter Positionen in Abhängigkeit der Länge der binären Repräsentation und der Kodierung reduziert wird. Dies soll helfen, die Bevorzugung kurzer Variablenketten, wie beim *single-point crossover*, zu verhindern, ohne ein präzises Verständnis der Wichtigkeit einzelner Variablenpositionen zu benötigen. SPEARS und DE JONG [SDJ91a] haben außerdem gezeigt, wie *uniform crossover* parametrisiert wird, indem eine Wahrscheinlichkeit auf den Austausch von Variablen angewendet wird. Dadurch kann der Umfang des „Zerreißens“ während des *crossover* unabhängig von der Länge der verwendeten Repräsentation kontrolliert werden.

### 3.3.4 Rekombination bei kombinatorischen Problemen

Im Bereich der kombinatorischen Optimierung werden in der Literatur viele unterschiedliche Rekombinationsoperatoren vorgestellt. Jeder dieser Operatoren hat bestimmte Eigenschaften in Bezug auf die Bewahrung von Informationen. Damit ist jeder Operator besonders zur Anwendung auf ausgewählte Probleme innerhalb der kombinatorischen Optimierung geeignet.

Bei der Lösung eines TSP Problems liegt die Zielstellung des Rekombinationsoperators auf der Bewahrung der Nachbarschaftsinformation zwischen den Knoten. Der Operator sollte deshalb die gemeinsamen Kanten der Eltern im Nachkommen erhalten. Wenn dagegen ein Reihenfolgeproblem (*scheduling*) bearbeitet wird, ist die Erhaltung der relativen Ordnung der Knoten von größerer Bedeutung. Damit ist für diese Probleme ein Operator am günstigsten, der diese relative Ordnung erhält. Bei wieder anderen Problemen (z.B. Zuweisungsprobleme – *assignment problem*) kann die Bewahrung der absoluten Anordnung der Knoten wichtig sein. Weitergehende Hinweise dazu werden in Abschn. 7.5 gegeben.

Die hier vorgestellten Rekombinationsoperatoren beschränken sich auf die in der Anwendung jeweils erfolgreichsten. Neben einer Beschreibung der Arbeitsweise der Operatoren wird Wert auf die Beschreibung der besonderen Eigenschaften eines jeden kombinatorischen Rekombinationsoperators gelegt. Dadurch wird die Auswahl der günstigsten Operatoren bei der Lösung eines bestimmten kombinatorischen Problems erleichtert.

#### **Edge recombination EX**

*Edge recombination EX* wurde in [WSF89] vorgestellt. Dieser Operator soll Nachbarschaftsbeziehungen (Kanten) zwischen den Knoten bewahren. Diese Eigenschaft ist insbesondere für die Lösung von TSP Problemen wichtig.

Im Verlauf der letzten Jahre wurde EX mehrfach überarbeitet und verbessert. Die verbesserten Versionen heißen *edge-2* [SMW91], *edge-3* bis hin zu *edge-4* [DW94]. Jede dieser Versionen baut auf der vorherigen auf. Auf die Unterschiede zwischen den einzelnen Versionen wird im folgenden an den entsprechenden Stellen eingegangen.

Ablauf von EX:

1. Im ersten Schritt werden die Nachbarschaftsbeziehungen in einer Kantentabelle (*edge table*) gesammelt. Für jeden Knoten werden die Knoten aufgelistet, die mit diesem in direkter Nachbarschaft stehen. Damit enthält die Tabelle alle Kanten beider Eltern.

In der Erweiterung *edge-2* wird in der Tabelle zusätzlich vermerkt, wenn eine Kante in beiden Eltern vorkommt (*common edges*). Diese Kanten werden entsprechend markiert. Diese Information wird in der weiteren Konstruktion von Nachkommen verwendet.<sup>12</sup>

---

<sup>12</sup> Kanten, die in beiden Eltern vorkommen, sollten auf jeden Fall im Nachkommen enthalten sein. In der ursprünglichen Version von EX konnte dies nicht gesichert werden, da diese Information nicht ausgewertet wurde. Dadurch kam es öfters zu einer ‘Mutation durch weglassen’ [WSS91].

2. Zu Beginn der Konstruktion des Nachkommen wird ein Knoten zufällig aus der Tabelle ausgewählt. Dieser Knoten ist der aktuelle Knoten. Alle Verweise auf diesen Knoten werden aus der Tabelle entfernt.
3. Bei der Suche nach dem nächsten Knoten, wird in die Nachbarschaftsliste des aktuellen Knoten geschaut, ob dort eine Kante steht, die bei beiden Eltern vorkommt (*common edge*). Wenn ja, wird dieser Knoten als nächstes verwendet<sup>13</sup>. Diese Verwendung gemeinsamer Kanten ist eine Erweiterung ab *edge-2*. Ansonsten wird der Knoten von der Nachbarschaftsliste ausgewählt, dessen eigene Nachbarschaftsliste am kürzesten ist. Treten mehrere mit gleich kurzen Listen auf, wird einer der Knoten zufällig ausgewählt. Die Verweise auf den ausgewählten Knoten werden aus der Tabelle entfernt.
4. Der vorige Schritt 3 wird so oft wiederholt, bis die Konstruktion des Nachkommen abgeschlossen ist oder ein Knoten auftritt, der keine Einträge mehr in seiner Nachbarschaftsliste hat (*dead terminal* – totes Ende).
  - Im einfachsten Fall kann an dieser Stelle unter den verbleibenden Knoten einer zufällig ausgewählt werden und die Konstruktion des Nachkommen geht weiter.
  - In *edge-3* wird beim Auftreten eines toten Endes getestet, ob in der Nachbarschaftsliste des anderen Endes des Nachkommen noch Einträge vorhanden sind (*live terminal* – lebendes Ende). Wenn ja, wird dieses Ende zum aktuellen und die Konstruktion wird mit Schritt 3 fortgesetzt. Dieser Schritt entspricht dem Invertieren des bisher erzeugten Nachkommen. (Je Nachkommen kann dieses Invertieren höchstens einmal vorgenommen werden. Es kann nicht garantiert werden, ob durch die bisherigen Konstruktionsschritte das andere Ende nicht schon ein totes Ende ist.)
5. Wenn beide Enden tot sind, kann einer der verbleibenden Knoten zufällig als nächster ausgewählt werden. Mit *edge-4* kann versucht werden, durch teilweises Invertieren des bisherigen Nachkommen wieder ein lebendes Ende zu erreichen. Dieses Vorgehen wurde bisher noch nicht viel getestet. Für weitere Informationen zu *edge-4* sei auf [DW94] und [Whi97b] verwiesen.

Wenn EX nur zusammen mit einem Mutationsoperator verwendet wird, dann ist dieser Rekombinationsoperator sehr gut für TSP Probleme geeignet [SMW91]. Wird dagegen eine lokale Optimierung der Individuen durchgeführt, dann ist *Maximal preservative crossover MPX* besser geeignet.

Auf der anderen Seite ist EX nicht für die Anwendung auf *scheduling* Probleme geeignet [SMW91].

---

<sup>13</sup> Außer beim allerersten Knoten kann immer nur eine gemeinsame Kante auftreten. Wenn es zwei gemeinsame Kanten in der ursprünglichen Tabelle gegeben hätte, wäre die eine Kante der Weg zum aktuellen Knoten gewesen und damit bereits gelöscht worden.

### **Maximal preservative crossover MPX**

Der Rekombinationsoperator *Maximal preservative crossover MPX* wurde von GORGES-SCHLEUTER [GS91] und MÜHLENBEIN [Müh91] für die Anwendung auf TSP Probleme vorgeschlagen. Eine Abwandlung zu MPX2 stellt [GS97a] vor.

Ablauf von MPX:

1. Ein Segment wird im ersten Elter ausgewählt. Dazu wird die Länge dieses Segments (*MPXlength*)<sup>14</sup> und die Anfangsposition (*MPXstart*) zufällig ausgewählt. Bei MPX2 wird die Anfangsposition danach solange verschoben, bis eine Kante gefunden wurde, die bei den Eltern unterschiedlich ist. Dadurch wird sichergestellt, das selbst bei sehr ähnlichen Eltern die Unterschiede zwischen diesen durch MPX2 gefunden werden.
2. Das ausgewählte Segment wird direkt in den Nachkommen kopiert, die Position bleibt erhalten.
3. Schrittweise werden weitere Knoten/Kanten zum Nachkommen hinzugefügt, bis ein gültiger Nachkomme erstellt wurde. Dabei kommen die folgenden Regeln in der angegebenen Reihenfolge zur Anwendung:
  - Gibt es eine Kante im zweiten Elter, die am letzten Knoten des Nachkommen beginnt? Wenn ja, dann wird diese Kante dem Nachkommen hinzugefügt.
  - Wenn im zweiten Elter keine gültige Kante gefunden wurde, so findet die Suche nach einer solchen Kante im ersten Elter statt. Eine gültige Kante wird dem Nachkommen zugefügt.
  - Wenn in beiden Eltern keine gültige Kante vorhanden ist, wird der nächste Knoten vom zweiten Elter ausgewählt und dem Nachkommen hinzugefügt. Dadurch wird eine neue Kante im Nachkommen erzeugt (entspricht einer Mutation).

MPX erzeugt Nachkommen, die so viele Kanten beider Eltern wie möglich in den Nachkommen bewahren.

MPX bewahrt die Nachbarschaftsinformation der Eltern nicht so gut wie *edge recombination*. Es wird mit einer geringeren Vorschau gearbeitet, um einen Bruch in der Tourenkonstruktion zu verhindern. Bei MPX wird ein nahegelegener Knoten gewählt (und kein zufälliger wie bei *edge recombination*), wenn eine neue Kante im Nachkommen erzeugt werden muß, die in keinem der Eltern auftritt.

Wenn MPX mit einem lokal optimierenden Operator (z.B. *2-opt* oder ähnlich) kombiniert wird (hybride Optimierung), zeigt MPX ein besseres Verhalten als *edge recombination*. Die Auswahl eines nahegelegenen Knoten zusammen mit dem Kopieren eines zusammenhängenden Segmentes vom Elter zum Nachkommen gibt MPX beim Vorliegen von lokal optimierten Individuen einen Vorteil bei der Anwendung auf TSP Probleme ([MW92], [DW94]).

---

<sup>14</sup> Gute Erfahrungen wurden mit einer Länge des Segments von etwa  $\frac{1}{3}$  der Gesamtlänge der Individuen berichtet [MW92]. In einer Variante von MPX [GS89] wurde mit einer Segmentlänge zwischen 3 und  $\frac{1}{3}$  der Gesamtlänge der Individuen gearbeitet. In [GS97a] wird für MPX2 eine Segmentlänge *MPXlength* von  $\frac{1}{10}$  bis  $\frac{1}{4}$  der Gesamtlänge der Individuen verwendet.

### **Order crossover-2 OX2 und Position crossover PX**

Die Rekombinationsoperatoren *order crossover-2 OX2*<sup>15</sup> und *position crossover PX* wurden von SYSWERDA [Sys91] vorgeschlagen.

Ablauf von OX2:

1. Es werden eine Anzahl (*OX2length*) von zufälligen Positionen im zweiten Elter ausgewählt. Die Anzahl der auszuwählenden Elemente sollte zwischen  $\frac{1}{3}$  bis  $\frac{1}{2}$  der Gesamtlänge der Individuen liegen.
2. Der erste Elter wird komplett in den Nachkommen kopiert. Danach werden die im zweiten Elter ausgewählten Elemente im Nachkommen (Kopie des ersten Elter) gesucht. Diese Elemente werden innerhalb ihrer Positionen in der Reihenfolge sortiert, in der sie im zweiten Elter auftreten.

Im Ergebnis enthält der Nachkomme die Elemente, die nicht im zweiten Elter ausgewählt wurden, genau an den Positionen, an denen sie im ersten Elter auftraten. Alle anderen Elemente (welche im zweiten Elter zufällig ausgewählt wurden) tauchen an den verbleibenden Positionen im Nachkommen auf und genau in der Reihenfolge (relative Ordnung), in der sie im zweiten Elter auftraten. Diese Beschreibung enthält indirekt den Ablauf von PX.

Ablauf von PX:

1. Zuerst werden eine Anzahl (*PXlength*) von zufälligen Positionen im ersten Elter ausgewählt. Die Elemente an diesen Positionen werden direkt in den Nachkommen unter Beibehaltung der Position kopiert.
2. Im zweiten Schritt werden die verbleibenden Elemente in der Reihenfolge in den Nachkommen kopiert, wie sie im zweiten Elter vorkommen.

Wenn  $PXlength = 1 - OX2length$  sind beide Operatoren in ihrem erwarteten Verhalten identisch. Entsprechende Überlegungen sind in [WY95] und [Whi97b] dargelegt.

Der von DAVIS in [Dav91] vorgeschlagene Rekombinationsoperator *uniform crossover* ist identisch zu PX (und OX2).

OX2 und PX bewahren die relative Ordnung der Elemente in den Eltern. Sie sind deshalb besonders zur Anwendung auf *scheduling* Probleme geeignet.

### **Weitere Rekombinationsoperatoren**

Neben den beschriebenen Rekombinationsoperatoren für kombinatorische Probleme werden in der Literatur eine große Anzahl weiterer vorgestellt. Einige von ihnen sollen im folgenden kurz angesprochen werden. Für weitere Operatoren von spezieller Bedeutung sei auf die Übersicht in [Whi97b] verwiesen.

Zwei der ersten Operatoren speziell für kombinatorische Probleme waren *partially matched crossover* [GL85] und DAVIS' *order crossover* [Dav85]<sup>16</sup>. In

<sup>15</sup> Hier wird in Anlehnung an [SMW91] und [Whi97b] der Begriff *order crossover-2 OX2* für SYSWERDA's Operator [Sys91] verwendet. Damit soll eine Unterscheidung von DAVIS' *order crossover* ermöglicht werden.

<sup>16</sup> Eine Beschreibung von *partially matched crossover* [GL85] und DAVIS' *order crossover* kann [SMW91] bzw. [Whi97b] entnommen werden.

der Anwendung und im Vergleich zu anderen Operatoren [SMW91] zeigte sich, daß beide Operatoren für TSP Probleme nicht so gut wie *edge recombination EX* geeignet sind. Bei *scheduling* Problemen wiederum werden sie von *order crossover-2 OX2* bzw. *position crossover PX* übertroffen.

Ähnliche Aussagen in Bezug auf die Anwendung können von *cycle crossover* [OSH87] berichtet werden. Dieser Operator brachte im Vergleich zu anderen Operatoren für die verschiedenen Problemklassen schlechte Ergebnisse [SMW91].

Ganz anders sieht es dagegen für den noch sehr neuen Rekombinationsoperator *edge assembly crossover EAX* [NK97] aus. Dieser Operator wurde speziell für die Anwendung auf TSP Probleme entwickelt. In der Anwendung zeigte EAX durchweg bessere Ergebnisse als *edge-3 recombination EX* [WRE98]. Damit scheint EAX der Rekombinationsoperator der Wahl für TSP Probleme zu sein. EAX hat aber Nachteile, welche die einfache Anwendung dieses Operators in einigen Fällen erschweren. Zum ersten ist der Ablauf von EAX deutlich komplexer im Vergleich zu den bisher beschriebenen Operatoren. Außerdem verwendet EAX für die Suche innerhalb des Operators lokale Information. Diese lokale Information ist nicht für alle TSP Probleme verfügbar. Zu guter Letzt ist aus den vorliegenden Veröffentlichungen nicht genau ablesbar, wie viele zusätzliche Berechnungen der Tourenlänge innerhalb des Operators ausgeführt werden. Sobald diese Frage beantwortet wird und eine Implementierung des Operators vorliegt, steht einer Anwendung von EAX aber nichts mehr im Wege.

## 3.4 Mutation

Durch die Mutation erfolgen zufällige Veränderungen der Individuen. Diese Veränderungen (Mutationsschritte) sind meist nur relativ gering und werden mit einer geringen Wahrscheinlichkeit (Mutationswahrscheinlichkeit bzw. Mutationsrate) auf die Variablen der Individuen angewendet.

Die Mutation wird im allgemeinen an den Nachkommen durchgeführt, nachdem diese durch Rekombination erzeugt wurden.

Für die Festlegung des angewendeten Mutationsschrittes und der Mutationswahrscheinlichkeit gibt es zwei grundlegend verschiedene Vorgehensweisen: beide Parameter bleiben in ihrer Anwendung während eines Laufs konstant oder es erfolgt eine stetige Anpassung in Abhängigkeit der bereits durchgeföhrten Mutationen des Laufs. In die erste Kategorie fallen die im folgenden erläuterten Verfahren zur Mutation reeller und ganzzahliger Variablen, Unterabschn. 3.4.1, und zur Mutation binärer Variablen, Unterabschn. 3.4.3. Zur zweiten Kategorie gehören die in Unterabschn. 3.4.2 erläuterten Verfahren zur Adaption der Schrittweiten der Mutation, die aus dem Bereich der Evolutionsstrategien stammen.

In Unterabschn. 3.4.4 werden Mutationsoperatoren für Probleme der kombinatorischen Optimierung vorgestellt, die für die entsprechenden Problemklassen geeignet sind.

### 3.4.1 Mutation reeller und ganzzahliger Variablen

Mutation bei reellen und ganzzahligen Variablen bedeutet, daß zufällig erzeugte Werte zu den Variablen mit einer geringen Wahrscheinlichkeit addiert werden. Bei der Mutation der Variablen muß damit bestimmt werden, mit welcher Wahrscheinlichkeit eine Variable mutiert wird (Mutationsrate) und wie groß die auszuführenden Veränderungen (Mutationsschritte) sind.

Die Unterschiede in der Behandlung von reellen und ganzzahligen Variablen sind bei der Mutation gering. Im Gegensatz zu reellen Variablen ist bei ganzzahligen Variablen die minimale Größe eines Mutationsschrittes 1 (Definition der ganzen Zahlen). Wenn eine ganzzahlige Variable zur Mutation ausgewählt wurde (Mutationswahrscheinlichkeit), so muß der minimale Mutationsschritt 1 betragen. Außerdem werden alle größeren Mutationsschritte jeweils auf die nächste ganze Zahl gerundet. Dadurch wird sichergestellt, daß nach der Mutation eine ganze Zahl vorliegt. Zudem unterscheidet sich eine mutierte Variable durch den minimalen Mutationsschritt von 1 wirklich von der nicht mutierten Variable. Im folgenden werden nur die Algorithmen für reelle Variablen beschrieben, eine Erweiterung auf ganzzahlige Variablen ist mit den gegebenen Hinweisen einfach.

Für die Festlegung der Mutationsrate werden in verschiedenen Artikeln ähnliche Ergebnisse berichtet. In [MSV93a] wird eine optimale Mutationsrate von  $\frac{1}{n}$  ( $n$ : Anzahl der Variablen eines Individuums) angegeben. Dies bedeutet, daß pro Mutation eines Individuums im Durchschnitt genau eine Variable mutiert wird. Ähnliche Ergebnisse werden in [Bäc93] und [Bäc96] für eine binäre Repräsentation berichtet. Für unimodale Funktionen ist eine Mutationsrate von  $\frac{1}{n}$  die beste Wahl. Eine Erhöhung der Mutationsrate zu Beginn des Laufs verbunden mit einer Verringerung auf  $\frac{1}{n}$  zum Ende des Laufs ergab nur eine minimale Beschleunigung der Suche. Für multimodale Funktionen werden keine solchen Aussagen gemacht, außer daß eine Adaption der Mutationsrate sinnvoll sein müßte. Die hier angegebenen Werte für die Mutationsrate gelten für separierbare Funktionen<sup>17</sup>. Die meisten in praktischen Anwendungen anzutreffenden Funktionen sind allerdings nicht vollständig separierbar. Für diese Funktionen sind aber keine anderen Angaben verfügbar, so daß auch hier die Verwendung einer Mutationsrate von  $\frac{1}{n}$  empfohlen wird.

Die Bestimmung der Größe des Mutationsschrittes ist schwierig. Einerseits hängt die beste Schrittweite von der Art des Problems ab, und andererseits kann sich die beste Schrittweite während eines Laufs verändern. Es ist allerdings bekannt, daß vorteilhafte Mutationen meist kleine Veränderungen der Variablen als Grundlage haben. Dies trifft besonders dann zu, wenn ein Individuum schon gut angepaßt ist. Anders ist die Situation, wenn neue Bereiche erkundet werden. In diesem Fall können größere Veränderungen schnell zu einem Erfolg führen. Als Schlußfolgerung kann abgeleitet werden, daß ein Mutationsoperator oft mit kleinen Mutationsschritten arbeiten sollte, aber ab und zu auch große Mutationsschritte angewendet werden.

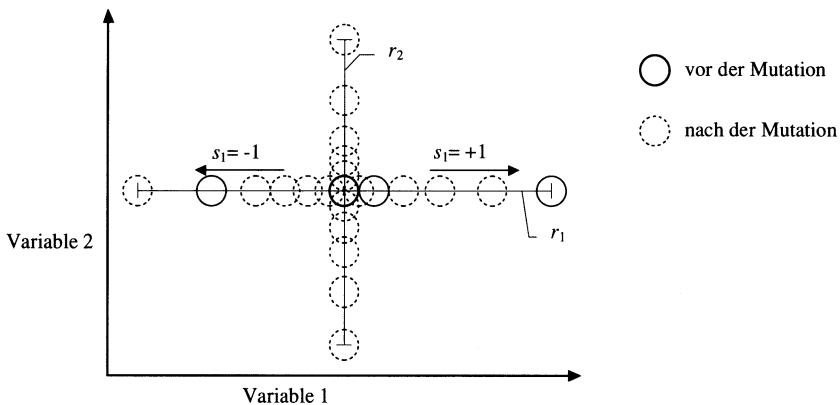
<sup>17</sup> Für separierbare Funktionen gilt:

$$f(x) = \sum_{i=1}^n f_i(x_i) \text{ mit } f_i: \text{eindimensionale Unterfunktion.}$$

Ein Mutationsoperator, der die genannten Forderungen erfüllt, wird in Gl. 3-26 gezeigt ([MSV93a], [Müh94]):

$$\begin{aligned}
 Var_i^{Mut} &= Var_i + s_i \cdot r_i \cdot a_i \quad i \in \{1, 2, \dots, n\} \text{ gleichverteilt,} \\
 s_i &\in \{-1, +1\} \text{ gleichverteilt} \\
 r_i &= r \cdot Domain_i, \quad r: \text{Mutationsbereich (mutation range),} \\
 &\quad Domain_i: \text{Definitionsbereich der Variable } i, \\
 a_i &= 2^{-u^k}, \quad u \in [0, 1] \text{ gleichverteilt,} \\
 k &: \text{Mutationspräzision (mutation precision)}
 \end{aligned} \tag{3-26}$$

Damit wird eine ausgewählte Variable verändert. Der Wert von  $s$  (Suchrichtung) gibt die Richtung des Mutationsschrittes vor. Die maximale Größe eines Mutationsschrittes wird durch den Definitionsbereich der Variablen und den Parameter  $r$  (Mutationsbereich – *mutation range*) vorgegeben. Für einen vorgegebenen Mutationsbereich von  $r = 0,1$  kann der Mutationsschritt damit maximal 10% des Definitionsbereiches der Variablen groß sein (obere Grenze des Mutationsschrittes). Eine untere Grenze des Mutationsschrittes wird durch den Parameter der Mutationspräzision  $k$  (*mutation precision*) definiert. Diese beträgt  $1/2^k$  in Bezug auf die obere Grenze des Mutationsschrittes. Der Wert von  $a$  bestimmt die relative Größe des Mutationsschrittes im vorgegebenen Intervall. Die Verteilung der Mutationsschritte innerhalb dieses Intervalls wird indirekt durch die Verwendung der Funktion  $2^{-u^k}$  festgelegt. Diese Verteilung hat sich in der Anwendung bewährt.



**Abb. 3-17.** Mögliche Positionen eines Individuums nach der Mutation reeller oder ganzzahliger Variablen

Dieser Algorithmus ist in der Lage, die meisten Punkte in einem Hyperwürfel zu generieren, der durch die Variablen des Individuums und den möglichen Bereich der Mutation (definiert durch den Mutationsbereich  $r$  (*mutation range*) und den Definitionsbereich der Variablen) aufgespannt wird. Es werden mit einer ho-

hen Wahrscheinlichkeit Individuen erzeugt, die sehr nah an dem nicht mutierten Individuum liegen und nur wenige Individuen, die weiter weg vom Ausgangsindividuum liegen. Abbildung 3-17 versucht davon einen Eindruck zu vermitteln.

Wie oben bereits beschrieben, bestimmt die Mutationspräzision  $k$  die minimal zu erzeugende Mutationsschrittweite. Die minimale Schrittweite beträgt  $1/2^k$ . Bis zu dieser Genauigkeit kann das Optimum einer Funktion erreicht werden. Der größte Schritt beträgt 1, immer bezogen auf die Größe des Mutationsbereiches. Daraus ergibt sich ein absoluter Bereich für die Mutationsschritte von  $[r, r \cdot 2^k]$ , bezogen auf den Definitionsbereich der Variablen.

Der in Gl. 3-26 angegebene Mutationsoperator mutiert mit einer Mutationsrate von  $\frac{1}{n}$ . Damit wird nur eine Variable pro mutiertem Individuum verändert<sup>18</sup>. Wie schon weiter oben angedeutet, können dadurch nur separierbare Funktionen sicher bearbeitet werden. Bei nicht separierbaren Funktionen kommt es mit einer hohen Wahrscheinlichkeit zu einem Steckenbleiben der Optimierung, da ab einer bestimmten Anpassung des Individuums durch Veränderung nur einer Variablen keine Verbesserung erreicht werden kann.

In [SVM96] wird eine Erweiterung des Mutationsoperators aus Gl. 3-26 vorgestellt, die dieses Problem ansatzweise mindert. Durch die Erweiterung wird jetzt nicht genau eine Variable mutiert, sondern zusätzlich werden auch die anderen Variablen des Individuums mutiert; allerdings sind bei diesen die Veränderungen wesentlich geringer. Dazu wurde ein dritter Parameter  $v$  eingeführt, der die Verteilung des bisher auf eine Variable beschränkten Mutationsschrittes auf die anderen Variablen dieses Individuums steuert.

Verteilte Mutation (Unterschiede zu Gl. 3-26):

$$\begin{aligned} Var_j^{Mut} &= Var_j + s_i \cdot r_j \cdot a_i \cdot v^{|j-i|}, \\ j &= (i, i-1, i+1, i-2, i+2, \dots), \quad i \in \{1, \dots, n\}, \\ r_j &= r \cdot \text{Domain}_j, \\ v &: \text{Mutationsverteilung} \quad (v = 0,5), \end{aligned} \tag{3-27}$$

Diese verteilte Mutation der Variablen führt dazu, daß eine Variable mit Index  $i$  wie bisher mutiert wird. Zusätzlich werden alle anderen Variablen auch mutiert, wobei die Größe des Mutationsschrittes mit steigendem Abstand zur Variablen  $i$  exponentiell abnimmt. Dies führt zu einer gekoppelten Veränderung der Variablen, die zueinander benachbart sind. Damit ist dieser Mutationsoperator besser für die Bearbeitung von Funktionen geeignet, bei denen benachbarte Variablen miteinander gekoppelt sind. Für einen Wert des Verteilungsparameters  $v$  von  $v = 0$  ergibt sich der Mutationsoperator aus Gl. 3-26.

Durch eine Veränderung der Parameter können sehr unterschiedliche Suchstrategien des Mutationsoperators eingestellt werden:

- Kleine Werte für die Mutationspräzision  $k$  definieren eine grobe Suche im Mutationsbereich, große Werte eine sehr feine Suche in der unmittelbaren Umgebung der Individuen. Ein Wert für eine robuste Suche ist  $k = 16$ .

<sup>18</sup> Durch Erhöhung der Mutationsrate über den Wert von  $\frac{1}{n}$  hinaus kann die Wahrscheinlichkeit der Mutation einer Variable erhöht werden. Bei einem Wert von  $\frac{4}{n}$  werden z.B. 4 Variablen pro Individuum entsprechend Gl. 3-26 mutiert.

- Der Bereich, in dem die Suche durchgeführt wird, kann durch den Mutationsbereich  $r$  stark eingeschränkt werden. Zu Beginn eines Laufs erreicht man durch einen Wert von  $r = 0,1$  eine global orientierte Suche, die den Definitionsbereich der Variablen gut durchsucht. Bei längeren Läufen sollte der Mutationsbereich schrittweise eingeschränkt werden, da es sonst, bedingt durch die minimal erreichbare Mutationsschrittweite, zu keinen weiteren Verbesserungen kommen kann. Eine Verringerung des Mutationsbereiches führt zu einer entsprechenden Verkleinerung der absoluten Mutationsschritte und ermöglicht dadurch eine immer weiter fortschreitende Verbesserung der Anpassung der Individuen. Dabei darf aber nicht vergessen werden, daß eine Einschränkung des Mutationsbereiches gleichzeitig zu einer stärkeren lokalen Suche führt.
- Wenn bekannt ist, daß die benachbarten Variablen der Zielfunktion miteinander gekoppelt sind, sollte die Mutationsverteilung  $v$  verwendet werden. Ein guter Startwert ist  $v = 0,5$ . Ansonsten sollte die Mutationsverteilung auf 0 gesetzt werden.

Typische Werte für die Parameter der Mutationsoperatoren aus Gl. 3-26 und 3-27 sind:

$$\begin{aligned} \text{Mutationsbereich } r: \quad r &\in [0,1, 10^{-6}] \\ \text{Mutationspräzision } k: \quad k &\in \{4, 5, \dots, 20\} \\ \text{Mutationsverteilung } v: \quad v &\in [0,1, 0,7] \end{aligned} \tag{3-28}$$

Für eine Veränderung der Parameter der Mutationsoperatoren während eines Laufs gibt es zwei Möglichkeiten:

- Es wird ein fester Plan vorgegeben, wie in Abhängigkeit vom Zustand oder Fortgang des Laufs (z.B. Anzahl der Generationen, erreichte Zielfunktionswerte, Unterschiede der Variablen der Individuen) die Parameter verändert werden. Dieses Vorgehen erfordert eine gute Einsicht in den zu erwartenden Verlauf der Optimierung und ein tiefes Verständnis der Wirkung der geänderten Parameter.
- Es werden gleichzeitig verschiedene Sätze von Parametern angewendet, wobei die zum jeweiligen Zeitpunkt erfolgreichsten Strategien größere Ressourcen erhalten. Dieses Prinzip ist ausführlich in Abschn. 4.6, ab S.102 dargestellt.

Die in diesem Unterabschnitt in Gl. 3-26 und 3-27 vorgestellten Mutationsoperatoren für reelle Variablen führen mit den hier angegebenen Parametereinstellungen für eine große Klasse von praktisch auftretenden Zielfunktionen eine robuste Suche durch. Durch die zu Beginn des Unterabschnittes beschriebenen Anpassungen können diese Operatoren auch auf ganzzahlige Variablen angewendet werden.

### 3.4.2 Mutation reeller Variablen mit Adaption der Schrittweiten

Für die Durchführung der Mutation reeller Variablen gibt es die Möglichkeit, Richtung und Schrittweite erfolgreicher Mutationen durch Adaption dieser Größen zu erlernen. Solche Mutationsverfahren sind z.B. Bestandteil der Evolutionsstrategien [Sch81] und der Evolutionären Programmierung [Fdb95].

Weiterentwicklungen dieser Verfahren bzw. neue Verfahren wurden in den letzten Jahren in mehreren Arbeiten veröffentlicht:

- Adaption von  $n$  (Anzahl der Variablen) Schrittweiten ([OGH93], [OGH94]: *ES-algorithm with derandomized mutative step-size control using accumulated information*),
- Adaption von  $n$  Schrittweiten und einer Richtung ([HOG95]: *derandomized adaptation of  $n$  individual step sizes and one direction – A II*),
- Adaption von  $n$  Schrittweiten und  $n$  Richtungen ([HOG95]: *derandomized adaptation of the generating set – A I*)
- Adaption der Kovarianzmatrix ([Han98]).

Um die zu adaptierenden Schrittweiten und Richtungen zu speichern, werden an jedes Individuum zusätzliche Variablen angehängt. Die Anzahl dieser zusätzlichen Variablen ist abhängig von der Anzahl der Variablen  $n$  und dem Verfahren. Für die Speicherung der adaptierten Schrittweiten werden  $n$  zusätzliche Schrittweiten und für die Speicherung einer adaptierten Richtung  $n$  zusätzliche Variablen benötigt. Für die Speicherung von  $n$  Richtungen würden damit  $n^2$  zusätzliche Variablen benötigt. Die Kovarianzmatrix benötigt auch  $n^2$  zusätzliche Variablen.

Weiterhin ist zu beachten, daß für die Adaption von  $n$  Schrittweiten  $n$  Generationen mit jeweils mehreren Individuen berechnet werden müssen. Bei  $n$  Schrittweiten und einer Richtung (A II) erhöht sich diese Zahl auf  $2n$  Generationen und bei  $n$  Richtungen bzw. der Kovarianzmatrix auf  $n^2$  Generationen.

Aus dem Speicherbedarf für zusätzliche Parameter und den notwendigen Berechnungen zur Adaption der Schrittweiten und Richtungen ergibt sich, daß nur die ersten beiden Verfahren für die praktische Anwendung relevant sind. Nur diese beiden führen die Adaption mit einem vertretbaren Aufwand durch. Die Adaption von  $n$  Richtungen (A I) bzw. der Kovarianzmatrix läßt sich mit den heute zur Verfügung stehenden Mitteln nur auf kleine Probleme anwenden.

Die Algorithmen für die Mutationsoperatoren mit Schrittweitenanpassung sollen hier nicht wiedergegeben werden, da diese komplex und lang sind. Statt dessen wird auf die oben genannten Arbeiten sowie [Ost97] und [Han98] verwiesen, in denen die Algorithmen beschrieben sind. Dort finden sich weitere Hinweise und Untersuchungsergebnisse zu den einzelnen Verfahren. Eine Beispielimplementierung ist außerdem in [GEATbx] enthalten. Auf weitere Anmerkungen, die für die Anwendung wichtig sind und nicht bzw. nur teilweise in den obigen Arbeiten enthalten sind, wird im folgenden eingegangen.

Die Mutationsoperatoren mit Schrittweitenanpassung benötigen eine etwas andere Parametrisierung des Evolutionären Algorithmus, als die Mutationsoperatoren für reelle Variablen. Sie arbeiten mit einer geringen Anzahl von Indivi-

duen, die jeweils viele Nachkommen produzieren. Von den Nachkommen werden nur die besten in die Population eingefügt, wobei alle Eltern ersetzt werden. Der Selektionsdruck ist 1, da alle Individuen gleich viele Nachkommen produzieren. Eine Rekombination wird nicht durchgeführt. Übliche Werte und Bereiche dieser Parameter sind:

- 1 (1-3) Individuum pro Population oder Unterpopulation,
- 5 (3-10) Nachkommen pro Individuum => *generation gap* = 5,
- die besten Nachkommen ersetzen alle Eltern => Wiedereinfügerate = 1,
- kein Selektionsdruck => *SP* = 1,
- keine Rekombination.

Es gibt eine zweite Variante, die im Prinzip gleich arbeitet, in der praktischen Anwendung jedoch einige Vorteile hat. Dabei wird mit einer größeren Population gearbeitet, bei der nur die allerbesten Individuen Nachkommen produzieren. Hier ersetzen (fast) alle Nachkommen die Eltern und bilden die neue Population:

- 5 -20 Individuen pro Population,
- so viele Nachkommen wie Eltern produzieren => *generation gap* = 1,
- Selektion: Abschneideselektion (*truncation selection*) mit Selektionsdruck von 3-10 (nur die besten Individuen werden ausgewählt und produzieren jeweils mehrere Nachkommen).

Diese zweite Variante ermöglicht eine bessere Auswertung, da alle Nachkommen zumindest kurzzeitig in der Population enthalten sind. Bei der ersten Variante werden die schlechten Nachkommen nie in die Population eingefügt und können damit nicht in die Auswertung einbezogen werden. In der praktischen Anwendung hat sich die zweite Variante bewährt.

Beim Einsatz dieser Mutationsoperatoren zeigte sich ein Problem sehr deutlich: die Festlegung der Anfangsgröße der individuellen Schrittweiten. In den Veröffentlichungen ist dort nur ein Wert von eins für alle Schrittweiten angegeben. Mit diesem Wert können aber nur einige Testfunktionen gut gelöst werden, bei denen zusätzlich alle Variablen denselben Definitionsbereich besitzen. Für die praktische Anwendung müssen die individuellen Anfangsschrittweiten in Abhängigkeit des Definitionsbereiches der Variablen festgelegt werden und zusätzlich sollte eine problemabhängige Skalierung der Größenordnung der Anfangsschrittweiten stattfinden. Dafür bietet sich, analog zu den in Unterabschn. 3.4.1 erläuterten Mutationsoperatoren für reelle Variablen, die Verwendung des Parameters Definitionsbereich *r* an.

Typische Werte für den Definitionsbereich zur Festlegung der Anfangsschrittweiten sind:

$$\text{Definitionsbereich } r: \quad r \in [10^{-3}, 10^{-7}] \quad (3-29)$$

Dieser Definitionsbereich bestimmt nur die Initialisierung der Schrittweiten zu Beginn eines Laufs. Bei der nachfolgenden Schrittweitenanpassung sind die Schrittweiten nicht beschränkt.

Ein größerer Wert für den Definitionsbereich hat zur Folge, daß die individuellen Schrittweiten zu Beginn groß sind. Dadurch werden die Nachkommen weiter entfernt von den Eltern erzeugt. Dies führt am Anfang zu einer groben Suche.

Ein kleiner Wert für den Mutationsbereich dagegen bedingt eine anfänglich feinere Suche. In der praktischen Anwendung muß zwischen beiden Extremen abgewogen werden. Bei einer zu groben Suche kann es passieren, daß keine Adaption der Schrittweiten stattfindet, die Optimierung kommt nicht vorwärts. Eine zu feine Suche andererseits kann erstens sehr lange benötigen, bis es zu einer gerichteten Suche kommt und zweitens in jedem lokalen Minimum steckenbleiben.

Der Einsatz der in diesem Unterabschnitt besprochenen Mutationsoperatoren ist besonders bei solchen Problemen vielversprechend, die korrelierte Variablen enthalten. Durch die Adaption der Schrittweiten können die Wechselbeziehungen zwischen den Variablen erlernt werden. Manche Probleme können dadurch sehr schnell und gut gelöst werden, wie z.B. ROSEN BROCK's Funktion (s. Unterabschn. 8.2.3).

Schwierig wird der Einsatz dieser Mutationsoperatoren, wenn die Zielfunktion verrauscht ist bzw. viele kleine Minima aufweist. Dann bleiben diese Operatoren in einem der lokalen Minima hängen.

### 3.4.3 Mutation binärer Variablen

Die Mutation von Individuen mit binären Variablen bedeutet das Umschalten der Variablenwerte, da jede Variable nur zwei Zustände haben kann. Die Länge des Mutationsschrittes ist damit immer 1. Für jedes Individuum werden die zu verändernden Variablen ausgewählt (meist gleichverteilt zufällig) und dann die Mutation durchgeführt. Abbildung 3-18 veranschaulicht diesen Vorgang für ein Individuum mit 11 Variablen, von denen Variable 4 mutiert wird.

vor Mutation	0 1 1 <b>I</b> 0 0 1 1 0 1 0
↓	
nach Mutation	0 1 1 <b>0</b> 0 0 1 1 0 1 0

Abb. 3-18. Mutation eines Individuums mit binären Variablen

Bisher wurde davon ausgegangen, daß die Repräsentation eines Individuums identisch zur Repräsentation des zu lösenden Problems ist. Dies ist aber nicht immer der Fall. Gerade im Bereich der Genetischen Algorithmen werden Probleme, deren Variablen reelle oder ganzzahlige Zahlen sind, in eine binäre Repräsentation umgewandelt und auf diese binäre Repräsentation wird ein Evolutionärer Algorithmus angewendet. Für die Umwandlung der binären Repräsentation in reelle Zahlen können verschiedene Kodierungen und Skalierungen eingesetzt werden. Die am meisten verwendeten Kodierungen sind die binäre und die gray Kodierung. Zusätzlich kann statt einer linearen Skalierung eine logarithmische Skalierung angewendet werden. Die Auswirkungen der verschiedenen Kodierungen und Skalierungen auf ein und dasselbe Individuum mit binären Variablen zeigt Abb. 3-19. Dabei wird angenommen, daß das Individuum in Abb. 3-18 eine reelle Zahl im Bereich [1, 10] kodiert.

Skalierung	linear		logarithmisch	
Kodierung	binär	gray	binär	gray
vor der Mutation	5.0537	4.2887	2.8211	2.3196
nach der Mutation	4.4910	3.3346	2.4428	1.8172

**Abb. 3-19.** Ergebnis und Auswirkungen einer binären Mutation für unterschiedliche Skalierungen und Kodierungen, wenn das Individuum in Abb. 3-18 eine reelle Zahl im Bereich [1, 10] repräsentiert

Welche Kodierung die günstigere ist, hängt stark vom zu lösenden Problem ab. Dafür lassen sich keine einheitlichen Aussagen machen. Einige Arbeiten berichten bessere Ergebnisse bei Verwendung der *gray* Kodierung (u.a. [CS88], [Whi99]). JONES und FORREST [JF95] untersuchten die Schwere einer Anzahl bekannter Testfunktionen für einen Genetischen Algorithmus. Dabei wurde auch die Abhängigkeit von binärer und *gray* Kodierung untersucht. In ihren Untersuchungen war es von der Länge der Kodierung abhängig, welche Kodierung etwas günstiger war. Als Maß verwendeten sie die Fitneß-Distanz-Korrelation. Aus den Ergebnissen ihrer Untersuchung läßt sich schlußfolgern, daß es keine allgemeine Empfehlung für eine Kodierung geben kann. Insgesamt gesehen zeichnet sich aber eine Bevorzugung der *gray* Kodierung ab.

Die Anwendung der linearen oder logarithmischen Skalierung ist problemabhängig, kann aber vom Anwender besser überblickt werden als die Auswahl der zu verwendenden Kodierung. Bei der logarithmischen Skalierung wird die Variable nicht über den gesamten Wertebereich gleichmäßig diskretisiert, sondern im unteren Teil des Definitionsbereiches der Variablen findet eine feinere Diskretisierung als im oberen Teil statt. Dies kann zum einen die Suche im wenig erfolgversprechenden oberen Teil verkürzen und gleichzeitig die erreichbare Genauigkeit im fein diskretisierten unteren Teil erhöhen.

Mit den zur Verfügung stehenden leistungsfähigen Mutationsoperatoren für reelle Variablen besteht allerdings kein Grund für die Kodierung reeller Variablen in eine binäre Repräsentation und die Anwendung entsprechender Operatoren. Die Vorteile des Einsatzes von Operatoren für reelle Variablen werden in verschiedenen Arbeiten gezeigt (u.a. [Mic94] und [Dav91]).

### 3.4.4 Mutation bei kombinatorischen Problemen

Das Grundprinzip der Mutationsoperatoren für kombinatorische Probleme besteht darin, daß eine Neuanordnung der Variablen innerhalb eines Individuums vorgenommen wird. Dies gewährleistet, daß die Variablen der mutierten Individuen weiterhin eine Permutation bilden.

Dazu wird von den Operatoren eine Anzahl von Positionen (Variablen) in einem Individuum ausgewählt und die Variablen innerhalb dieser Positionen neu angeordnet. Die einzelnen Operatoren unterscheiden sich darin, wieviele Positio-

nen ausgewählt werden, wie diese Positionen zueinander angeordnet sind und wie die Variablen neu angeordnet werden. Im folgenden wird kurz auf die einzelnen Varianten eingegangen.

Über die Größe der Mutationsrate kann gesteuert werden, wieviele dieser Neuanordnungsvorgänge bzw. Mutationen pro Individuum stattfinden. Eine Mutationsrate von  $1/n$  ( $n$ : Anzahl der Positionen/Variablen eines Individuums) führt zu einer der im folgenden beschriebenen Mutationen. Bei einer höheren Mutationsrate werden entsprechend mehrere der Mutationen durchgeführt. Dadurch kann vom Anwender gesteuert werden, wie stark die Veränderung des Individuums durch den Mutationsoperator ist.

Für kombinatorische Probleme existieren viele Spezialverfahren, die für die Lösung eines speziellen Problems bzw. einer Problemklasse entworfen wurden. Wenn für das zu lösende kombinatorische Problem ein solches Verfahren bekannt ist, sollte immer zuerst dieses Verfahren angewendet werden. Wenn dies nicht zum gewünschten Erfolg führt, können diese speziellen Verfahren manchmal als lokal optimierende Mutationsoperatoren eingesetzt werden (s. die Beschreibung von *reverse mutation* und *2-opt* ab S.56).

### **Insert mutation**

Bei der *insert mutation* (Einfügemutation) werden 2 Positionen an beliebiger Stelle des Individuum ausgewählt. Danach wird eine der Variablen an diesen Positionen vor bzw. hinter der anderen Position eingefügt. Es findet damit ein Verschieben einer Variable statt. (Die Bezeichnung *move mutation* würde diesen Operator gut beschreiben.)

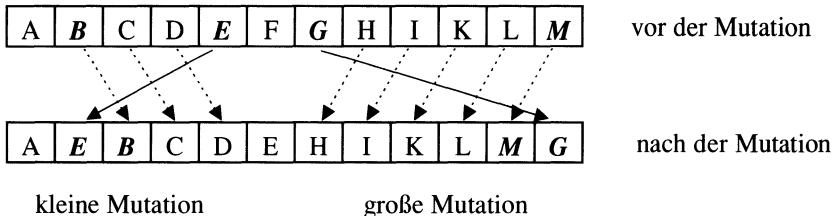


Abb. 3-20. Ablauf der *insert mutation* (Einfügemutation)

Die Entscheidung, ob die erste oder zweite ausgewählte Variable verschoben wird, findet gleichverteilt statt. Die zweite Entscheidung, ob vor oder hinter der anderen Variable eingefügt wird, kann auch gleichverteilt stattfinden. Allerdings ist es möglich, diese Entscheidung festzulegen: die zweite Variable wird vor der ersten eingefügt, die erste nach der zweiten Variable. Dadurch sind alle Positionen durch diesen Mutationsoperator erreichbar.

### Swap mutation

Die *swap mutation* (Austauschmutation) ist in ihrem Ablauf sehr ähnlich zur *insert mutation*. Es werden 2 Positionen an beliebiger Stelle des Individuums ausgewählt. Danach werden die beiden Variablen an diesen Positionen ausgetauscht.

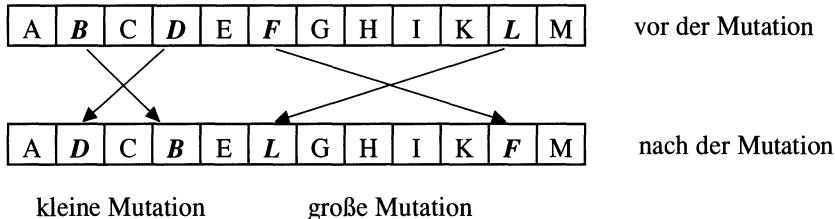


Abb. 3-21. Ablauf der *swap mutation* (Austauschmutation)

### Reverse mutation (2-opt, 3-opt, k-opt)

Auch bei der *reverse mutation* werden 2 Positionen an beliebiger Stelle des Individuums ausgewählt. Im Gegensatz zu den bisherigen Operatoren werden aber nicht nur die 2 Variablen neu angeordnet. Statt dessen wird das gesamte Segment zwischen den beiden Positionen umgekehrt (*reverse*). Dieser Operator ist auch als *2-opt* (bzw. *2-opt move*) bekannt [LK73].

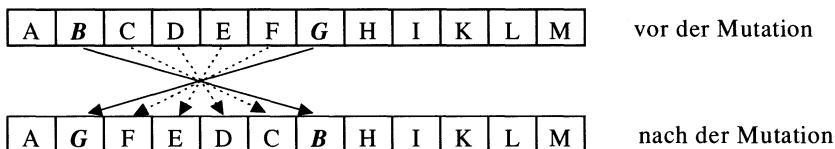


Abb. 3-22. Ablauf der *reverse mutation* (auch *2-opt move*)

Die Anwendung von *2-opt* auf alle möglichen Kanten in einem Individuum (dies ergibt  $n \cdot (n-1)/2$  Operationen) führt eine lokale Optimierung des Individuums durch. Oftmals wird solch ein lokaler Optimierungslauf auch als *2-opt* bezeichnet.

Dieser Operator kann natürlich auf mehr als 2 Positionen erweitert werden. Das Individuum kann in 3 (*3-opt*) oder mehr (*k-opt*) Abschnitte geteilt werden, die neu angeordnet (verschoben und umgekehrt) werden. Verglichen mit *2-opt* gibt es eine wesentlich größere Anzahl von Möglichkeiten, das Individuum zu zerteilen und die einzelnen Segmente neu anzurichten. Ein Test aller möglichen *3-opt* Züge ist deutlich aufwendiger als bei *2-opt* (8fach für symmetrische TSP und bis zu 48fach für *scheduling* Probleme).

### Scramble mutation

Bei der *scramble mutation* wird eine Anzahl von Positionen an beliebigen Stellen des Individuums ausgewählt. Die Variablen an diesen Positionen werden zufällig neu angeordnet. Die Variablen an allen anderen Positionen bleiben unverändert.

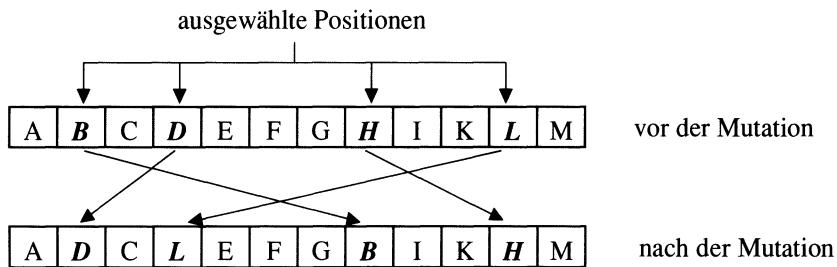


Abb. 3-23. Ablauf der *scramble mutation*

Die Anzahl der ausgewählten Positionen entscheidet darüber, wie stark dieser Operator ein Individuum verändert. Je mehr Positionen ausgewählt werden, um so größer ist der Unterschied zum ursprünglichen Individuum. Wenn dagegen nur 2 Variablen ausgewählt werden, geht dieser Operator in *swap mutation* über.

## 3.5 Wiedereinfügen (Reinsertion)

Nachdem die Nachkommen durch Rekombination und Mutation produziert und durch die Anwendung der Zielfunktion bewertet wurden, müssen sie in die Population eingefügt werden. Dies geschieht durch das Wiedereinfügen (*reinsertion*). Dabei wird entschieden:

- wieviele und welche Nachkommen in die Population eingefügt werden und
- welche Individuen der Population durch die einzufügenden Nachkommen ersetzt werden.

Eine Auswahl der Nachkommen muß nur durchgeführt werden, wenn nicht alle produzierten Nachkommen in die Population eingefügt werden sollen. Der Parameter dafür ist die Wiedereinführerate (*reinsertion rate*). Die Wiedereinführerate gibt an, wieviele der Individuen einer Population maximal durch Nachkommen zu ersetzen sind. Bei einer Wiedereinführerate von 1,0 hieße dies, daß alle Individuen der Population durch Nachkommen ersetzt werden können. Wenn die Populationsgröße konstant sein soll, kann die Wiedereinführerate maximal einen Wert von 1,0 annehmen.

Wieviele Nachkommen produziert wurden, hängt vom Parameter *generation gap* (engl.: „Generationslücke“) ab.

Beide Parameter zusammen – *generation gap* und Wiedereinfügerate – bestimmen, wieviele der Nachkommen eingefügt werden. Ist die Wiedereinfügerate  $\geq$  *generation gap*, so werden alle Nachkommen in die Population eingefügt. Ist die Wiedereinfügerate dagegen  $<$  *generation gap*, so wird nur ein Teil der Nachkommen in die Population eingefügt.

Diese verallgemeinernde Betrachtungsweise durch beide Parameter, *generation gap* und Wiedereinfügerate, ermöglicht die einschließende Angabe verschiedener Varianten, die bisher als eigenständige Einheiten betrachtet wurden:

1. Einfaches Wiedereinfügen (*pure reinsertion*):
  - *generation gap* = 1,0 und *Wiedereinfügerate* = 1,0:  
es werden genau so viele Nachkommen wie Individuen in der Population erzeugt und alle Nachkommen ersetzen alle Individuen der Population.
2. Zufälliges bzw. elitest Wiedereinfügen (*uniform* bzw. *elitest reinsertion*):
  - *generation gap* < 1,0 oder *Wiedereinfügerate* < 1,0:  
es werden weniger Nachkommen als Individuen in der Population in die Population eingefügt, die Individuen der Population werden durch die Nachkommen gleichverteilt zufällig (*uniform reinsertion*) ersetzt bzw. die schlechtesten Individuen der Population (*elitest reinsertion*) werden ersetzt.
3. Wiedereinfügen mit Auswahl der Nachkommen (*reinsertion with offspring selection*):
  - *generation gap* > *Wiedereinfügerate*:  
es wird nur ein Teil der Nachkommen [*Wiedereinfügerate/generation gap %*] in die Population eingefügt. Zum Einfügen werden die besten Nachkommen verwendet (Truncation-Selektion).

Das einfache Wiedereinfügen wird in vielen einfachen Evolutionären Algorithmen verwendet. Jedes Individuum hat dabei eine Lebensdauer von genau einer Generation. Da die bisher besten Individuen in jeder Generation durch Nachkommen ersetzt werden, ist nicht gesichert, daß ihre Information in der Population erhalten bleibt. Die Güte des besten Individuums der Population kann schlechter werden.

Dies kann durch den Einsatz des elitest Wiedereinfügen verhindert werden. Dabei wird sichergestellt, daß die bisher besten Individuen der Population in der Population verbleiben und nur ersetzt werden, wenn bessere Individuen gefunden wurden. Das elitest Wiedereinfügen entspricht dem, was in vielen Veröffentlichungen als elitest Selektion bezeichnet wird. Vom algorithmischen Ablauf her hat dies aber nichts mit der Selektion zu tun, sondern muß im Zusammenhang des Wiedereinfügens betrachtet werden. Das elitest Wiedereinfügen wird bei vielen Evolutionären Algorithmen verwendet, um sicherzustellen, daß das beste Individuum/die besten Individuen nicht verloren gehen. Dazu wird das *generation gap* auf einen Wert knapp unter eins gesetzt, es werden fast so viele Nachkommen wie Individuen in der Population erzeugt und alle Nachkommen werden in die Population eingefügt, wobei die besten Individuen nicht ersetzt werden. Es sollte beachtet werden, daß das beste Individuum nicht in die neue Population

übernommen wird, sondern die Nachkommen ersetzen das beste Individuum der Population nicht. Einen Spezialfall davon stellen die *steady-state* Evolutionären Algorithmen dar. Bei diesen ist das *generation gap* so klein, daß nur wenige Individuen pro Generation produziert werden, die danach in die Population eingefügt werden (zufälliges oder elitest Wiedereinfügen bzw. Ersetzen der Eltern).

Die dritte Variante, bei der nur ein Teil der Nachkommen zum Einfügen ausgewählt wird, kann einmal zu einer vollständigen Ersetzung der Individuen der Population führen (*Wiedereinführerate* = 1,0) oder es wird das elitest Wiedereinfügen (*Wiedereinführerate* < 1,0) verwendet. Durch die Selektion zwischen den Nachkommen, bevor diese in die Population eingefügt werden, erhalten die schlechten Nachkommen keine Chance zur Produktion von eigenen Nachkommen. Diese Variante wird dann benutzt, wenn ein Individuum im Durchschnitt mehrere Nachkommen produziert, wie dies z.B. bei den Verfahren der Evolutionsstrategie der Fall ist, s. Unterabschn. 3.4.2, ab S.51. Jedes Individuum produziert dabei z.B. fünf Nachkommen (*generation gap* = 5), von denen im Durchschnitt nur eins ausgewählt wird (*Wiedereinführerate* = 1,0). Durch diese Sichtweise ist es möglich, den Ablauf einer Evolutionsstrategie im selben Kontext wie andere Evolutionäre Algorithmen zu betrachten.

Beim Wiedereinfügen ist zu beachten, daß die Nachkommen in genau den Selektionspool eingefügt werden, aus dem die Eltern ausgewählt wurden (globales Modell: gesamte Population, regionales Modell: Unterpopulation, lokales Modell: Nachbarschaft). Für das globale und regionale Modell gelten direkt die hier gemachten Angaben. Auf die Besonderheiten des lokalen Wiedereinfügen, das bei der Verwendung einer lokalen Selektion verwendet werden muß, wird in Unterabschn. 4.3.4, ab S.83 eingegangen.

Die bisherigen Varianten des Wiedereinfügens können erweitert werden, wenn das Alter eines Individuums mit in Betracht gezogen wird. Dazu wird zuerst betrachtet, wie lange bzw. wieviele Generationen bei den bisherigen Varianten ein Individuum in der Population bleibt:

- einfaches Wiedereinfügen:
  - die Lebensdauer eines Individuums beträgt genau eine Generation,
- elitest Wiedereinfügen: die durchschnittliche Lebensdauer eines Individuums ist umgekehrt proportional zum Anteil der neuen Individuen pro Generation,
  - bei einem *generation gap* nahe 1 (Wiedereinführerate wird auf 1,0 gesetzt) beträgt die Lebensdauer der meisten Individuen nur eine Generation, die besten Individuen können aber viele Generationen in der Population verbleiben (elitest Selektion),
  - bei einem sehr kleinen *generation gap* (Wiedereinführerate wird auf 1,0 gesetzt), dies entspricht einem *steady state* Algorithmus, würden die Individuen viele Generationen in der Population verbleiben.

Die durchschnittliche Lebensdauer eines Individuums der Population ergibt sich dabei aus dem invertierten Produkt von Wiedereinführerate und *generation gap* (in Generationen):

$$\text{Lebensdauer} = \frac{1}{\text{Wiedereinführerate} \cdot \text{generation gap}} \quad (3-30)$$

Wenn das Alter der Individuen berücksichtigt wird, kann ein maximales Alter für ein Individuum angegeben werden. Diese maximale Lebensdauer sollte im Vergleich zur durchschnittlichen Lebensdauer angegeben werden und muß immer größer 1 sein. Sobald ein Individuum die maximale Lebensdauer erreicht hat, wird es zuerst von neuen Nachkommen ersetzt, noch bevor jüngere Individuen ersetzt werden (auch wenn die jüngeren Individuen schlechter als die zu alten Individuen sind).

Durch die Definition eines maximalen Lebensalters kann verhindert werden, daß einmal sehr gute Individuen für eine zu große Zahl an Generationen in der Population verbleiben. Dies wird besonders wichtig, wenn mit einem kleinen *generation gap*, elitest Wiedereinfügen und einem höheren Selektionsdruck gearbeitet wird. Die wenigen bisher guten Individuen werden immer wieder als Eltern ausgewählt und die Nachkommen ersetzen immer nur die schlechtesten Individuen, der Algorithmus kann steckenbleiben. Durch ein Ersetzen und damit Verwerfen der alten Individuen nach einiger Zeit kann mehr neue Information in die Population eingebracht und unter Umständen ein Steckenbleiben verhindert werden.

Das Prinzip der Einbeziehung des Alters von Individuen findet sich überall in der Natur. Bei einfachen Lebewesen leben Individuen oft nur über eine Generation bzw. die gleichzeitig in einer Population lebenden Individuen haben alle etwa ein Alter. Bei etwas höher entwickelten Lebewesen können einzelne besonders gute Individuen mehrere Generationen erleben bzw. ein höheres Alter im Vergleich zu den anderen Individuen erreichen. Bei den höheren Lebewesen wird auf einmal nur ein kleiner Teil der Population neu gebildet und jedes Individuum erreicht ein Alter, daß in etwa proportional seiner Güte ist. Dieses Alter umfaßt im allgemeinen ein mehrfaches dessen, was jeweils an Zeit zur Produktion von Nachkommen benötigt wird. Trotzdem kann ein maximales Alter nicht überschritten werden bzw. ab einem bestimmten Alter können keine Nachkommen mehr erzeugt werden. Es spricht damit einiges dafür, abgesehen von den Bemerkungen im vorigen Absatz, daß besonders bei komplexeren Algorithmen bzw. Problemen die Berücksichtigung des Alters sinnvoll ist.

Ähnliche Überlegungen werden auch in anderen Arbeiten angestellt, u.a. in [SK92] („Verhinderung eines genetischen Einfrierens“).

### 3.6 Initialisierung der Individuen

In diesem Abschnitt wird auf die verschiedenen Varianten eingegangen, die Individuen der Anfangspopulation zu erstellen. Dies beginnt bei der zufälligen Erstellung, über die Einbeziehung weniger guter Individuen bis zur Verwendung bekannter Lösungen als Schwerpunkt der Initialisierung.

In der bisherigen Beschreibung Evolutionärer Algorithmen wurde die Initialisierung der Individuen der Anfangspopulation nicht weiter betrachtet. Es wurde einfach von einer zufälligen Initialisierung der Individuen innerhalb des vorgegebenen Suchraumes ausgegangen. Die zufällige Initialisierung ist die normale Variante der Erstellung der Anfangspopulation. Ein Abriß wird in Unterabschn. 3.6.1 gegeben.

Oftmals liegt aber Vorwissen über ein zu lösendes Problem in Form einiger bekannter guter Lösungen (Individuen) vor. Diese bekannten hochqualitativen Lösungen können und sollten in die Optimierung einbezogen werden. Eine Form der Einbeziehung dieses Wissens ist die nicht-zufällige Initialisierung der Anfangspopulation, auch als *inoculation* [Ray94] bezeichnet. Diese Methode wird im folgenden unter dem Begriff der erweiterten Initialisierung zusammengefaßt. In Unterabschn. 3.6.2 wird sie vorgestellt und bewertet.

### 3.6.1 Zufällige Initialisierung

Bei der zufälligen Initialisierung werden die Variablen der Individuen gleichverteilt zufällig aus dem vorgegebenen Definitionsbereich der Variablen ausgewählt. Für die verschiedenen Repräsentationen sind nur wenige Besonderheiten zu beachten.

Bei Verwendung der reellen Repräsentation kann die folgende Gleichung verwendet werden:

$$Var_i^{reell} = \text{rand}_i \cdot (\text{Domain}_{upper_i} - \text{Domain}_{lower_i}) + \text{Domain}_{lower_i} \quad (3-31)$$

*rand*: gleichverteilte Zufallszahl in [0,1]

Für ganzzahlige Variablen kann das gleiche Prinzip wie bei reellen Variablen mit einer anschließenden Rundung auf den nächsten ganzzahligen Wert verwendet werden. Zu beachten ist dabei, daß dann die Grenzen des Definitionsbereichs nur mit halber Wahrscheinlichkeit ausgewählt werden. Dies kann durch eine Vergrößerung des Definitionsbereiches (nur für die Initialisierung) um fast 0,5 an beiden Grenzen umgangen werden.

Binäre Variablen können nur zwei Werte annehmen, 0 oder 1. Eine einfache Variante der zufälligen Initialisierung von binären Variablen kann mit der folgenden Gleichung erreicht werden:

$$Var_i^{binär} = \text{rand}_i > 0,5 \quad (3-32)$$

Der Vergleich liefert 0 für Zufallswerte zwischen 0 und 0,5 sowie 1 für Zufallszahlen größer als 0,5 bis 1. Damit ist der Anteil der Werte 0 und 1 in den Individuen bei der Verwendung von 0,5 gleich groß. Wird der Vergleich dagegen mit Werten kleiner 0,5 durchgeführt, erhöht sich der Anteil der 1'en (ein Vergleich mit 0,1 führt zu 90% 1'en und 10% 0'en). Entsprechend wird der Anteil der Werte von 0 größer, wenn zum Vergleich eine Zahl größer als 0,5 verwendet wird.

Bei kombinatorischen Optimierungsproblemen ist vorher bekannt, welche Elemente in den Variablen vorkommen können. Eine zufällige Initialisierung heißt hier, eine Permutation der Elemente zu erstellen. Ob zur Repräsentation der einzelnen Elemente einfach Zahlen verwendet werden, wobei jede Zahl eindeutig für ein bestimmtes Element steht, oder Buchstaben bzw. komplexere Zeichen, ist eine Frage der Implementierung.

Eine einfache Variante zur Erstellung einer zufälligen Permutation von Elementen ist die Sortierung eines Vektors von Zufallszahlen. Dabei kommt es nicht

auf die sortierten Elemente an. Die Reihenfolge, in der die Elemente nach der Sortierung im Vergleich zum unsortierten Vektor vorliegen, ergibt eine zufällige Permutation der Zahlen von 1 bis zur Länge des Vektors.

### 3.6.2 Erweiterte Initialisierung (nicht-zufällig)

Beim Vorliegen einer oder mehrerer guter Lösungen können die Informationen dieser Individuen bei der Initialisierung der Anfangspopulation verwendet werden. Gute Lösungen können sein:

- Ergebnis vorangegangener Optimierungen (z.B. auch durch Anwendung anderer Optimierungsverfahren),
- bekanntes Wissen von Fachleuten (Experten- bzw. Spezialistenwissen),
- bisher bekannte Lösungen („bisher wurde diese Variante angewendet“),
- problemspezifische Informationen, die aus der Zielfunktion ableitbar sind (z.B. durch Voruntersuchungen).

Auf welchen Wegen lassen sich diese Lösungen in die Initialisierung einbinden?

In der einfachsten Variante werden die bekannten Lösungen direkt und ohne Veränderung in die Anfangspopulation eingefügt. Alle anderen Individuen werden zufällig erstellt. Wenn die eingefügten Lösungen nützliche Informationen enthalten, kann dies die Suche unterstützen und zu einer effektiveren Optimierung führen. Es darf aber nicht vergessen werden, daß die verwendete Repräsentation und die eingesetzten Operatoren einen großen Einfluß auf den Verlauf der Optimierung haben.

Bei der zweiten Variante werden die bekannten Lösungen nicht direkt eingefügt, sondern vorher verändert. Für diese Veränderung kann im Prinzip jeder geeignete Mutationsoperator verwendet werden, bei dem die Mutationsrate auf einen sehr hohen Wert gesetzt wird. Diese Methode der Veränderung durch einen Mutationsoperator muß insbesondere bei kombinatorischen Problemen angewendet werden.

Für Parameteroptimierungen steht neben der Verwendung von Mutationsoperatoren eine weitere Variante zur Verfügung. Die bekannten Lösungen werden als Erwartungswert einer zufälligen Initialisierung verwendet. In einem festzulegenden Bereich um diese Erwartungswerte (die guten Lösungen) werden Individuen zufällig erzeugt. Ob nun eine Gleichverteilung in diesem Bereich stattfindet oder eine Normalverteilung verwendet wird, hängt von der Anwendung ab. Algorithmisch erfolgt diese erweiterte Initialisierung analog zu Gl. 3-31 mit den angepaßten Bereichen und neuem Mittelpunkt.

Die beschriebenen Methoden der erweiterten Initialisierung können untereinander und mit der zufälligen Initialisierung kombiniert werden. Die guten Lösungen werden unverändert verwendet, weitere Individuen durch zufällige Veränderung der guten Lösungen hinzugefügt und die restlichen Individuen zufällig erstellt.

In der Anwendung der erweiterten Initialisierung gibt es damit einige Varianten und Parameter, die eingestellt werden müssen:

- erfolgt eine direkte Übernahme der vorliegenden Lösungen,

- wieviele zufallsbehaftete Individuen basierend auf den vorliegenden Lösungen werden erstellt und mit welchem Verfahren,
- wie stark werden diese Individuen zufällig verändert (Größe des Bereichs der Zufallsauswahl),
- werden außerdem vollständig zufällig erstellte Individuen in die Anfangspopulation eingefügt?

Mit den beschriebenen Verfahren der erweiterten Initialisierung ist es einfach, vorliegende Lösungen in den Suchprozeß einzubinden. Alle Methoden der erweiterten Initialisierung können außerdem dazu verwendet werden, einen beendeten Optimierungslauf neu zu starten. Die beste oder besten Lösungen des Optimierungslaufes werden zur erweiterten Initialisierung der Anfangspopulationen des neuen Laufes verwendet.

Diese Variante setzte ESHELMAN [Esh91] beim CHC-Algorithmus ein. Bei einem Neustart wurden die besten bisher gefundenen Lösungen einer Massenmutation unterzogen, der Mutationsoperator wurde mit einer Mutationsrate von 35% auf diese Lösungen angewendet. Die erhaltenen Individuen bildeten die Anfangspopulation des nächsten Laufes.

Bei der Anwendung der erweiterten Initialisierung sollte beachtet werden, daß die in die Anfangspopulation eingefügten guten Lösungen die Gefahr bergen, die Population zu dominieren. Dadurch wird die Vielfalt in der Population verringert und es kommt zu einer vorzeitigen Konvergenz der Suche. Deshalb muß in der Anwendung das richtige Maß zwischen dem Einbringen guter Lösungen und der Auswahl der Operatoren und deren Optionen gefunden werden. Ansonsten kann es dazu kommen, daß zwar sehr schnell etwas bessere Lösungen gefunden werden, diese aber schlechter sind, als wenn mit einer zufällig erstellten Anfangspopulation gestartet worden wäre.

Für eine Untersuchung zur erweiterten Initialisierung und Ergebnisse der praktischen Anwendung sei auf [SR96] verwiesen.

## 3.7 Abbruchkriterien

Bei einer Optimierung mit Evolutionären Algorithmen läuft ein Prozeß über eine Anzahl von Generationen, bis ein definiertes Abbruchkriterium erreicht wird.

Dieses Kriterium (*termination criteria*) sollte so definiert sein, daß:

- die Optimierung erst beendet wird, wenn ein ausreichend gutes Ergebnis erreicht wurde und
- nicht mehr (Zielfunktions-) Berechnungen durchgeführt werden, als notwendig zur Erreichung obigen Ziels sind.

In diesem Abschnitt werden eine Reihe von Abbruchkriterien vorgestellt und in ihrer Anwendbarkeit untersucht. Diese Kriterien lassen sich in direkte und abgeleitete Kriterien unterscheiden. Im folgenden wird nur kurz auf die direkten Abbruchkriterien eingegangen, da diese sehr gut zu überschauen sind. Vertieft werden die Untersuchungen der abgeleiteten Abbruchkriterien, da deren Wirkung vom Anwender nicht einfach zu erfassen bzw. zu überblicken ist.

### 3.7.1 Direkte Abbruchkriterien

Die direkten Abbruchkriterien ergeben sich unmittelbar aus dem Kontext der Optimierung. Die folgenden direkten Kriterien lassen sich unterscheiden:

- maximale Generationen: maximale Anzahl von Generationen bzw. maximale Anzahl an Zielfunktionsberechnungen,
- maximale Rechenzeit: absolut oder CPU-Zeit,
- „globales“ Optimum: Differenz des besten Zielfunktionswertes zu einem vorgegebenen Wert (z.B. ein ausreichend guter Wert bzw. das bekannte globale Optimum).

Die maximale Anzahl von Generationen (bzw. Zielfunktionsberechnungen) ist das am häufigsten eingesetzte Abbruchkriterium bei der Anwendung Evolutionärer Algorithmen. Sein Vorteil ist die gute Überschaubarkeit und der garantierte Abbruch der Optimierung. Zugleich kann durch die Vorgabe dieses Kriteriums sehr leicht der Aufwand der Optimierung eingestellt werden. Da die Rechenzeit in den meisten Fällen linear mit der Anzahl der Generationen steigt, lässt sich außerdem das Ende des Optimierungslaufes recht gut vorhersagen.

Das Abbruchkriterium maximale Rechenzeit dient insbesondere Vergleichsrechnungen. Das Kriterium kann aber auch eingesetzt werden, wenn ein Ergebnis zu einer bestimmten Zeit vorliegen muß bzw. nicht mehr Zeit zur Verfügung steht. Zu beachten ist, ob die Zeitmessung in „echter“ Zeit (Uhrzeit) erfolgt oder in CPU-Zeit. Beide Werte können (auf UNIX Systemen) stark voneinander abweichen. Wenn es um eine vorgegebene Rechenzeit gehen soll, ist es besser, mit der CPU-Zeit zu arbeiten. Das Abbruchkriterium maximale Rechenzeit garantiert den Abbruch der Optimierung und ist gut zu überschauen bzw. einzusetzen.

Für das Abbruchkriterium Differenz zu einem vorgegebenen Wert („globales“ Optimum) gibt es zwei Einsatzbereiche. Beim ersten ist das Optimum für eine Zielfunktion bekannt und es wird vorgegeben, wie nah die Optimierung diesem bekannten Optimum kommen soll, bevor sie abgebrochen wird. Dies dient einem Vergleich verschiedener Optimierungsverfahren; wie schnell bzw. wie gut sie einen definierten Wert erreichen. Dies wird oftmals beim *Benchmarking* von Algorithmen eingesetzt. Der zweite Einsatzbereich liegt dort, wo die Größenordnung eines zufriedenstellenden Zielfunktionswertes bekannt ist. Dann kann das Erreichen dieses Zielfunktionswertes als Grundlage für die Beendigung der Optimierung dienen. Dies wird insbesondere dann auftreten, wenn ein bekanntes Problem in einer neuen Variation bearbeitet wird. Ein Problem des Abbruchkriteriums Differenz zum globalen Optimum ist, daß eine Beendigung der Optimierung nicht garantiert werden kann.

#### **Was muß bei Abbruchkriterien ohne garantiierten Abbruch getan werden?**

Wenn der verwendete Optimierungsalgorithmus das eingestellte Abbruchkriterium nicht erreicht, würde es zu keinem Ende der Optimierung kommen. Deshalb sollte ein Kriterium ohne garantiiertes Ende immer mit einem der beiden garantiiert endenden Kriterien (maximale Generationen oder maximale Rechenzeit) kombiniert werden. Dies kann durch zusätzliche Anwendung eines dieser beiden

---

garantierten Abbruchkriterien erreicht werden. Das erste erreichte Abbruchkriterium führt dann zum Beenden der Optimierung.

### 3.7.2 Abgeleitete Abbruchkriterien

Die abgeleiteten Abbruchkriterien ergeben sich aus der Berechnung von Hilfsgrößen. In diesem Abschnitt werden die folgenden abgeleiteten Abbruchkriterien untersucht:

- **Standardabweichung:** Standardabweichung der Zielfunktionswerte der aktuellen Generation,
- **laufender Mittelwert:** Differenz zwischen Mittelwert der besten Zielfunktionswerte der letzten Generationen und dem besten Zielfunktionswert der aktuellen Generation,
- **GuterSchlechtester:** Differenz zwischen dem Zielfunktionswert des besten und schlechtesten Individuums der aktuellen Generation,
- **Phi:** Quotient aus Mittelwert aller Zielfunktionswerte und dem besten Zielfunktionswert der aktuellen Generation,
- **Kappa:** Gleichartigkeit der Variablenwerte der Individuen der Population.

Die ersten 3 Kriterien werden in [Pad97] vorgeschlagen, die letzten beiden in [Vös97].

Eine Bewertung der abgeleiteten Abbruchkriterien ist deutlich schwieriger als bei den direkten Abbruchkriterien. Die Entwicklung dieser Kriterien während eines Optimierungslaufes ist nur in den seltensten Fällen genau vorhersehbar oder einigermaßen linear, wie dies bei den Kriterien maximale Generationen oder maximale Rechenzeit der Fall ist. Zudem ist der Verlauf und die Skalierung der Abbruchwerte zum einen von der Art des zu lösenden Problems und zum anderen von den eingesetzten Operatoren abhängig.

Bis auf das Abbruchkriterium *laufender Mittelwert* ist für keines der abgeleiteten Kriterien das Erreichen des Abbruchpunktes garantiert. Deshalb gilt für alle diese Kriterien (sicherheitshalber einschließlich *laufender Mittelwert*) das am Ende von Unterabschn. 3.7.1 (S.64) Gesagte zur Kombination der Kriterien mit einem der garantierten Abbruchkriterien maximale Generationen und maximale Rechenzeit, um ein Ende des Optimierungslaufes sicherzustellen.

Im folgenden wird auf jedes der abgeleiteten Abbruchkriterien kurz eingegangen, die Funktionalität beschrieben sowie auf die auftretenden Probleme beim Einsatz dieses Kriteriums hingewiesen.

#### **Standardabweichung**

Beim Abbruchkriterium *Standardabweichung* wird aus allen Zielfunktionswerten der aktuellen Generation deren Standardabweichung berechnet. Wenn der berechnete Wert den vorgegebenen Abbruchwert erreicht oder unterschreitet, wird die Optimierung beendet.

Standardabweichung (*std*) der Zielfunktionswerte einer Generation:

$$std_{Gen} = \sqrt{\frac{1}{Nind} \sum_{i=1}^{Nind} (ZFW_{Gen}^i - \overline{ZFW}_{Gen})^2}; \quad (3-33)$$

*Gen*: Generation, *ZFW*: Zielfunktionswert, *Ind*: Anzahl der Individuen

Zwei Probleme treten bei der Anwendung dieses Kriteriums auf:

- die Skalierung des Abbruchwertes hängt von der Skalierung der Zielfunktionswerte des zu lösenden Problems ab und
- einer oder wenige schlechte Zielfunktionswerte können ein sehr klein werden dieses Kriteriums verhindern, so daß der Abbruchwert nicht erreicht wird.

Die problemabhängige Skalierung lässt sich nicht allgemein lösen, da kein Vergleichswert eines schlechten bzw. guten Zielfunktionswertes für jedes Problem zur Verfügung steht. Sobald aber etwas mit einem Problem gearbeitet wurde, ist die Größenordnung meist bekannt und der Abbruchwert kann für weitere Optimierungen eingestellt werden.

Das zweite Problem, bei dem einige aus der Reihe fallende (sprich sehr schlechte) Zielfunktionswerte den berechneten Wert überwiegen, ist zum einen von den verwendeten Operatoren des Evolutionären Algorithmus abhängig und kann nur schwierig umgangen werden.

Der Hintergrund der Verwendung der *Standardabweichung* als Abbruchkriterium ist, daß sich im allgemeinen die Zielfunktionswerte der Individuen im Verlauf einer Optimierung immer ähnlicher werden und dadurch die *Standardabweichung* entsprechend kleiner. Dies gilt aber nur, wenn wirklich alle Zielfunktionswerte immer ähnlicher werden.

Sehr oft kommen aber Operatoren zum Einsatz, die im Verlauf der Optimierung mit konstanten Mutationen (große, mittlere, kleine Mutationsschrittweite) oder anderen gleichbleibenden Veränderungen der Individuen arbeiten. Insbesondere im fortgeschrittenen Verlauf einer Optimierung führen diese Operatoren für einige Nachkommen zu schlechten Zielfunktionswerten (im Vergleich zu den aktuell besten Zielfunktionswerten). Das heißt, selbst am Ende einer Optimierung, wenn die meisten Individuen sehr ähnlich sind, werden in jeder Generation eine Anzahl von Individuen erzeugt, die sehr schlechte Zielfunktionswerte haben. Dies führt natürlich zu einem entsprechend hohen Wert der Standardabweichung der Zielfunktionswerte.

Das Problem der schlechten Zielfunktionswerte tritt kaum auf, wenn adaptive Operatoren (z.B. die Mutationsoperatoren von Evolutionsstrategien aus Unterabschn. 3.4.2) verwendet werden, bei denen z.B. die Mutationsschrittweite angepaßt wird. Bei diesen Operatoren kommt es kaum zur Erzeugung sehr schlechter Zielfunktionswerte bei einer weit fortgeschrittenen Optimierung. Damit kann beim Einsatz solcher Operatoren die *Standardabweichung* gut eingesetzt werden. Leider haben diese adaptiven Operatoren wieder andere Nachteile (z.B. keine Definition für integer Variablen), so daß sie nicht immer eingesetzt werden können.

Eine Möglichkeit, die Probleme mit schlechten Zielfunktionswerten zu umgehen bzw. abzuschwächen wäre, nur einen gewissen Anteil der Individuen der Po-

pulation zur Berechnung der *Standardabweichung* heranzuziehen. Allerdings ist die Festlegung des Anteils der zu verwendenden Individuen nicht ganz einfach. Wenn zu wenige Individuen einbezogen werden, kann der Abbruch zu früh erfolgen.

Im Verlauf einer Optimierung wird es beim Abbruchkriterium *Standardabweichung* zu starken Sprüngen von Generation zu Generation kommen. Es gibt keine stetige Entwicklung des Wertes hin zum vorgegebenen Abbruchwert. Bei der Verwendung adaptiver Operatoren sind diese Schwankungen wesentlich geringer und es ist eine gewisse stetige Verringerung der *Standardabweichung* zu beobachten.

Die bei Verwendung der *Standardabweichung* auftretenden Probleme haben ihre Ursache in der Verwendung des Mittelwertes aller Zielfunktionswerte. Bei einigen der folgenden Abbruchkriterien wird wiederum der Mittelwert von Zielfunktionswerten oder der schlechteste Zielfunktionswert eingesetzt. Damit treten dort dieselben Probleme auf. Entsprechend gelten dort die hier gemachten Aussagen zum Umgehen der Probleme bzw. wann sie nicht auftreten.

### **Laufender Mittelwert (running mean)**

Beim Abbruchkriterium *Laufender Mittelwert* wird aus den besten Zielfunktionswerten der letzten Generationen der Mittelwert gebildet und von diesem Mittelwert die Differenz zum besten Zielfunktionswert der aktuellen Generation berechnet. Unterschreitet diese Differenz den vorgegebenen Abbruchwert, wird der Optimierungslauf beendet. Für dieses Abbruchkriterium muß die Anzahl der Generationen (*RunMeanGen*) festgelegt werden, die in die Bildung des Mittelwertes der besten zurückliegenden Generationen einbezogen werden.

*Laufender Mittelwert* der zurückliegenden Zielfunktionswerte:

$$\text{runmean}_{\text{Gen}} = \text{ZFW}_{\text{Gen}}^{\text{best}} - \frac{1}{\text{RunMeanGen}} \sum_{i=\text{Gen}-\text{RunMeanGen}}^{\text{Gen}-1} \text{ZFW}_i^{\text{best}}; \quad (3-34)$$

Gen: aktuelle Generation, ZFW: Zielfunktionswert,

RunMeanGen: Anzahl der zu verwendenden zurückliegenden Generationen

Dieses Abbruchkriterium ist relativ gut überschaubar. Da nur die besten Zielfunktionswerte verwendet werden, treten nicht die Skalierungsprobleme durch schlechte Zielfunktionswerte auf, die beim Abbruchkriterium *Standardabweichung* besprochen wurden. Allerdings ist auch hier die Größenordnung des Abbruchwertes problemabhängig. Dies muß bei der Angabe des entsprechenden Wertes beachtet werden.

Die wichtigste Anwendung des Abbruchkriteriums *Laufender Mittelwert* liegt darin, daß der Optimierungslauf beendet wird, wenn über einige Generationen (*RunMeanGen*) hinweg keine oder nur eine kleine Verbesserung des besten Zielfunktionswertes stattfindet. Wenn über *RunMeanGen* Generationen keine Verbesserung erzielt wurde, wird *runmean* zu 0 und das Abbruchkriterium ist in jedem Fall erfüllt.

Es muß sichergestellt werden, daß die Anzahl der zu verwendenden Generationen *RunMeanGen* so groß ist, daß durch eine kurzzeitige Stagnation der Op-

timierung kein Abbruch hervorgerufen wird. In vielen Versuchen mit verschiedenen Zielfunktionen ergab sich der folgende Wert als robust und wird daher empfohlen:

$$\text{RunMeanGen} = 15 \quad (3-35)$$

Es kann bei einigen Problemen vorkommen, daß es selbst nach 10 oder 15 Generationen der Stagnation doch noch zu weiteren deutlichen Verbesserungen der Zielfunktionswerte kommt. Dann könnte einfach die Anzahl der zu verwendenden Generationen *RunMeanGen* erhöht werden. Allerdings wird dies nicht empfohlen, da eine Stagnation über 15 und mehr Generationen darauf hindeutet, daß der Optimierungsalgorithmus nicht genügend leistungsfähig ist (die Suche entartet in eine Zufallssuche). Deshalb sollte der verwendete Optimierungsalgorithmus verändert werden. Dies könnte durch die Verwendung von mehr Individuen und/oder Unterpopulationen geschehen. Außerdem können andere Operatoren bis hin zu unterschiedlichen Strategien eingesetzt werden, die für diese Problemklasse besser geeignet sind.

Das Abbruchkriterium *Laufender Mittelwert* kann sehr gut bei vielen Optimierungen eingesetzt werden. Der große Vorteil ist, daß eine Stagnation bzw. ein sehr langsames Fortschreiten der Optimierung sicher erkannt wird und innerhalb der angegebenen Anzahl von Generationen der Optimierungslauf beendet wird.

### **GuterSchlechtester**

Für das Abbruchkriterium *GuterSchlechtester* wird die Differenz zwischen dem besten und schlechtesten Zielfunktionswert der aktuellen Generation berechnet. Sobald dieser Wert eine vorgegebene Schwelle unterschreitet, wird der Optimierungslauf abgebrochen.

$$\text{goodworst}_{\text{Gen}} = \text{ZFW}_{\text{Gen}}^{\text{worst}} - \text{ZFW}_{\text{Gen}}^{\text{best}}; \quad (3-36)$$

Gen: Generation, ZFW: Zielfunktionswert,

Hintergrund dieses Kriteriums ist, daß ein Optimierungslauf beendet wird, sobald der schlechteste Zielfunktionswert einer Generation sehr nah am besten Zielfunktionswert ist. Dies spricht für ein starkes Zusammenziehen der Population auf einen kleinen Bereich und ist oft auf eine Konvergenz auf einen Punkt zurückzuführen.

Da bei diesem Kriterium der schlechteste Zielfunktionswert der aktuellen Generation verwendet wird, treten dieselben Probleme auf, die beim Kriterium *Standardabweichung* angesprochen wurden (selbst zum Ende der Optimierung sind in der Population einige sehr schlechte Zielfunktionswerte enthalten). Und genau wie bei der *Standardabweichung* gelten die dort gemachten Aussagen, wie diese Probleme umgangen werden können.

Das Abbruchkriterium *GuterSchlechtester* kann wiederum bei der Verwendung adaptiver Operatoren eingesetzt werden. Siehe dazu die Anmerkungen beim Kriterium *Standardabweichung*. Im allgemeinen hat sich die Anwendung dieses Abbruchkriteriums nicht bewährt und kann daher nicht empfohlen werden.

## **Phi**

Das Kriterium *Phi* wird in [Vös97] als Konvergenzkriterium definiert. Hier wird es als Abbruchkriterium genutzt.

Der Parameter *Phi* als Konvergenzkriterium ist der Quotient aus dem besten Zielfunktionswert der aktuellen Generation und dem Mittelwert aller Zielfunktionswerte der aktuellen Generation. Im Falle der Konvergenz der Population nähert sich dieser Wert 1. Deshalb wird für das Abbruchkriterium *Phi* die Differenz zwischen 1 und dem obigen Quotienten gebildet.

Berechnung von *Phi*:

$$\Phi_{Gen} = 1 - \left( \frac{ZFW_{Gen}^{best}}{\overline{ZFW}_{Gen}} \right); \quad (3-37)$$

*Gen*: Generation, *ZFW*: Zielfunktionswert

Hintergrund dieses Kriteriums ist, daß ein Optimierungslauf beendet wird, sobald alle Zielfunktionswerte einer Generation sehr nah am besten Zielfunktionswert sind. Dies spricht für ein starkes Zusammenziehen der Population auf einen kleinen Bereich und ist oft auf eine Konvergenz auf einen Punkt zurückzuführen.

Da bei diesem Kriterium alle Zielfunktionswerte der aktuellen Generation verwendet werden, treten dieselben Probleme auf, die bei den Kriterien *Standardabweichung* und *GuterSchlechtester* angesprochen wurden. Es gelten die dort gemachten Aussagen, wie diese Probleme umgangen werden können bzw. wann das Abbruchkriterium *Phi* eingesetzt werden kann (Verwendung adaptiver Operatoren).

## **Kappa**

Das Kriterium *Kappa* wird in [Vös97] als Konvergenzkriterium definiert. Hier wird es als Abbruchkriterium genutzt.

Berechnung von *Kappa*:

$$\begin{aligned} \kappa_{Gen} &= \frac{\sum_{i=1}^{Nind-1} \sum_{j=i+1}^{Nind} Dist_{Gen}^{ij}}{\kappa_{max}}, \\ Dist_{Gen}^{ij} &= \sqrt{\sum_{k=1}^{Nvar} \frac{(Var_j^k - Var_i^k)^2}{(Bound_{high}^k - Bound_{low}^k)^2}}, \quad \kappa_{max} = \frac{Nind^2 - Nind}{2}; \end{aligned} \quad (3-38)$$

*Gen*: Generation, *Nind*: Anzahl der Individuen, *Nvar*: Anzahl der Variablen

Das Kriterium *Kappa* bewertet die räumliche Verteilung der Individuen der aktuellen Generation im gesamten Suchraum. Dazu wird die normalisierte Distanzmatrix zwischen allen Individuen gebildet. Dies ist möglich, da für die Variablen der Individuen Definitionsbereiche vorgegeben sind, die als Bezugsgrenzen dienen. Aus der Summe aller Distanzen (wobei nur die obere Dreiecksmatrix verwendet wird, da in der unteren Dreiecksmatrix dieselben Werte noch einmal stehen) ergibt sich ein Wert, der immer kleiner wird, je ähnlicher die Individuen

sind. Durch Bildung des Quotienten dieser Summe mit dem maximal möglichen  $Kappa_{max}$  sowie der Bewertung mit der Variablenzahl wird  $Kappa$  in den Bereich  $[0, 1]$  gebracht. Kleinere Werte von  $Kappa$  zeigen eine stärkere Konvergenz der Individuen an.

In die Berechnung des Abbruchwertes  $Kappa$  werden alle Individuen der aktuellen Population einbezogen. Dadurch können bei der Anwendung Probleme auftreten. Zum Ende einer Optimierung sind fast alle Individuen einer Population sehr ähnlich. Allerdings können einige schlechte Individuen mit deutlich anderen Variablenwerten in der Population enthalten sein. Diese wenigen Individuen verhindern, daß der Wert von  $Kappa$  sehr klein wird. Dadurch wird der Abbruchwert nie erreicht. Dieses Problem kann gemildert werden, wenn nur ein Teil der Individuen der Population in die Berechnung von  $Kappa$  einbezogen wird.

Das Problem der Skalierung tritt beim Abbruchkriterium  $Kappa$  nicht auf, da für die Variablen durch den vorgegebenen Definitionsbereich eine problemspezifische Skalierung vorliegt.

Bei der Festlegung des Abbruchwertes muß die Dimension des Problems beachtet werden. Für ein Problem mit 10 Variablen ergeben sich Werte für  $Kappa$  in einer anderen Größenordnung als für ein Problem mit 200 Variablen, selbst wenn die Individuen in einem scheinbar ähnlich kleinen Raum liegen.  $Kappa$  ist eine Maßzahl für die Größe des Raumes in dem die Individuen liegen im Vergleich zum gesamten Suchraum. Bei hohen Dimensionen ergeben sich sehr kleine Werte für  $Kappa$ .

Im Zusammenhang mit den verwendeten Operatoren und Einstellungen des Evolutionären Algorithmus muß beachtet werden, daß ein hoher Selektionsdruck ( $SP > 1,8$ ) zu einer schnellen Beschränkung der Individuen auf einen kleinen Bereich im Suchraum führen kann. Durch diese recht frühe Beschränkung, die aber meist nicht zu einem Stillstand der Optimierung führt, könnte das Abbruchkriterium  $Kappa$  zu früh ausgelöst werden.

Wenn ein multimodales Problem (mehrere Extremwerte) bearbeitet wird, so ist es wahrscheinlich, daß sich die Individuen in mehreren Bereichen des Suchraums verteilen (entsprechend den mehreren Extremwerten). Durch diese Verteilung der Individuen wird der Wert von  $Kappa$  nicht klein. Ein Abbruch kann nicht garantiert werden. Dasselbe gilt bei einer mehrkriteriellen Optimierung, bei der sich die Individuen nicht auf einen Punkt konzentrieren, sondern entlang der PARETO-Front (s. Unterabschn. 3.1.3).

### **3.7.3 Einsatz der Abbruchkriterien**

In der vorangegangenen Darstellung der einzelnen Abbruchkriterien wurden die Probleme angesprochen, die bei einigen der Abbruchkriterien auftreten und wie diese zumindest zum Teil umgangen werden können bzw. wann sie kaum auftreten. An dieser Stelle sollen in einer kurzen Zusammenfassung Empfehlungen für den praktischen Einsatz der Abbruchkriterien gegeben werden.

Um eine ordnungsgemäße Beendigung der Optimierung zu sichern, sollte immer eines der garantierten Abbruchkriterien maximale Generationen oder maximale Zeit eingesetzt werden. In der bisherigen Arbeit hat sich dafür maximale

Generationen bewährt. Damit kann der maximale Rechenaufwand nach oben sicher begrenzt und das Ende der Optimierung gut vorhergesagt werden.

Um zu verhindern, daß eine Optimierung noch lange läuft, obwohl keine besseren Zielfunktionswerte im Verlauf mehrerer Generationen gefunden wurden, sollte das Abbruchkriterium *Laufender Mittelwert* (*running mean*) zusätzlich eingesetzt werden. Wird der entsprechende Abbruchwert auf 0 gesetzt, so erfolgt nur dann ein Abbruch, wenn über 15 Generationen (Parameter *RunMeanGen*) überhaupt kein besseres Individuum gefunden wurde. Dies ist ein Zeichen, daß mit den eingesetzten Operatoren kaum etwas besseres gefunden werden kann, selbst wenn wesentlich länger weiter gesucht werden würde.

Bei der Optimierung werden im allgemeinen Operatoren eingesetzt, die während eines Laufs etwa gleichbleibende Veränderungen an den Individuen vornehmen. Gegen Ende der Optimierung führt dies öfters zu sehr schlechten Individuen (verglichen mit den besten gefundenen Individuen). Dadurch wird man mit den noch verbliebenen abgeleiteten Abbruchkriterien kaum zu einer vernünftigen Beendigung der Optimierung kommen. Damit kann der Einsatz der Abbruchkriterien *Standardabweichung*, *GuterSchlechter*, *Phi* und *Kappa* für die Anwendung nicht empfohlen werden.

Wenn dagegen adaptive Operatoren im Evolutionären Algorithmus eingesetzt werden, müßte über die Anwendung der gerade abgelehnten Abbruchkriterien neu nachgedacht werden.

Das Abbruchkriterium *Globales Optimum* bietet sich hauptsächlich für vergleichende Untersuchungen verschiedener Evolutionärer Algorithmen an, bei denen es auf den Abbruch der Optimierung beim Erreichen eines bestimmten Zielfunktionswertes ankommt.

## 3.8 Zusammenfassung

In diesem Kapitel wurden grundlegende Operatoren Evolutionärer Algorithmen vorgestellt. Diese Erläuterungen bauen auf der in Kap. 2 vorgenommenen Beschreibung von Aufbau und Struktur Evolutionärer Algorithmen auf. Die dort begonnene Betrachtung und Betonung der Gemeinsamkeiten stand auch bei der Beschreibung der evolutionären Operatoren im Vordergrund.

Als ein Beispiel sei hier die Rekombination genannt. Das allgemeinste Verfahren ist die intermediäre Rekombination. Aus dieser läßt sich durch Einschränkung die Linien-Rekombination ableiten. Eine weitere Einschränkung führt zur diskreten Rekombination. Aus der diskreten Rekombination wiederum lassen sich die Verfahren der Rekombination binärer Variablen ableiten. Aus dieser immer weitergehenden Einschränkung ergibt sich direkt die Größe der Bereiche, in denen die Nachkommen erzeugt werden können. Die größten Freiheiten bietet die intermediäre Rekombination, bei allen anderen Verfahren ist der Bereich der möglichen Nachkommen immer weiter eingeschränkt. Diese Betrachtungsweise vereinfacht einen Vergleich der Auswirkungen der Operatoren deutlich.

Durch die in Kap. 2 vorgestellte Struktur sowie die einheitliche Beschreibung der Operatoren Evolutionärer Algorithmen in diesem Kapitel eröffnen sich neue Wege des Vergleichs bisher getrennt betrachteter Verfahren. Diese neuen Ein-

sichten bieten erweiterte Möglichkeiten für die Auswahl der für ein Problem am besten geeigneten Operatoren und die Zusammenstellung angepaßter Algorithmen (s. Kap. 7).

## **4 Populationen, verschiedene Strategien und Konkurrenz**

Zu Beginn der Arbeit mit Evolutionären Algorithmen wurde ausschließlich eine gesamte Population verwendet. Mit der Verfügbarkeit von Parallelrechnern wurden Versuche zur Parallelisierung der Berechnungen durchgeführt. Die Motivation dafür kam aus verschiedenen Bereichen. Zunächst stand der Gedanke dahinter, daß auf diese Weise die Berechnungen auf mehrere Prozessoren verteilt werden können, wodurch sich die Rechenzeiten deutlich verringern lassen. Dadurch ergab sich gleichzeitig die Möglichkeit, Evolutionäre Algorithmen auf deutlich größere Probleme als bisher anzuwenden.

Für die Parallelisierung der Berechnungen mußte eine Unterteilung der Gesamtpopulation vorgenommen werden. Es wurden verschiedene Modelle und Varianten der Unterteilung der Gesamtpopulation bzw. der Verteilung der Individuen auf die verschiedenen Prozessoren vorgestellt. Gemeinsam ist diesen Unterteilungen der Population, daß die Teile voneinander getrennt sind und trotzdem miteinander in Verbindung stehen. Je nach Unterteilung der Gesamtpopulation und der Art der Verbindung der Teile lassen sich verschiedene Populationsmodelle unterscheiden.

Bei frühen Anwendungen der Populationsmodelle wurden bald Vorteile dieser erweiterten Evolutionären Algorithmen gefunden, die mit einfachen Evolutionären Algorithmen nicht zu erreichen waren. Diese Vorteile konnten auf die verwendeten Populationsmodelle (und nicht auf die Parallelisierung der Berechnungen) zurückgeführt werden.

In diesem Kapitel werden die verschiedenen Populationsmodelle vorgestellt und teilweise ausführlich erläutert. Ausgehend davon werden Erweiterungen eines der Populationsmodelle eingeführt und ausführlich in Struktur, Funktion und Parametern sowie ihrer Anwendung erläutert. Die Beschreibungen in diesem Kapitel sind unabhängig davon, ob der Evolutionäre Algorithmus auf paralleler Hardware implementiert oder auf serielle Weise abgearbeitet wird. Die hier in einer seriellen Implementierung gewonnenen Erkenntnisse konnten nachfolgend direkt auf eine parallele Implementierung übertragen werden.

Im ersten Abschnitt wird eine Übersicht zu Klassifikationen von Populationsmodellen gegeben. Nach einer Bewertung der verschiedenen Klassifikationen wird die Unterteilung der Populationsmodelle nach dem Bereich der Selektion in globales, regionales und lokales Modell ausgewählt und als Grundlage für die weiteren Ausführungen erläutert.

In den nachfolgenden Abschnitten wird jeweils auf ein einzelnes Populationsmodell eingegangen. Besonders ausführlich werden das regionale und das lokale Modell vorgestellt. Die umfassenden Darstellungen der Verfahren und Parame-

ter, verbunden mit der Erläuterung der Vor- und Nachteile, charakterisieren die Populationsmodelle ausführlich. „Neue“ Populationsmodelle lassen sich dadurch leicht einem der vorgestellten Modelle zuordnen.

Nach der Vorstellung der Populationsmodelle werden Erweiterungen eines der Populationsmodelle, des regionalen Modells, vorgestellt. Zum ersten ist dies die Anwendung verschiedener Strategien, zum zweiten der Einsatz miteinander konkurrierender Unterpopulationen. Neben der Beschreibung der Prinzipien wird auf den Einsatz der Erweiterungen und deren Vorteile eingegangen. Die Erweiterungen des regionalen Modells stellen leistungsfähige Verfahren zur Untersuchung verschiedener Konfigurationen und zur Kombination der Vorteile mehrerer Verfahren in einem Lauf dar, die mit einer einheitlichen Population oder einem einfachen Populationsmodell nicht erreichbar wären. Weiterhin imitieren diese Erweiterungen Verfahren bzw. Prinzipien, die auch in der biologischen Evolution zu beobachten sind.

## 4.1 Klassifikation von Populationsmodellen

In mehreren Arbeiten wurden Klassifikationen für Populationsmodelle (parallele Modelle) Evolutionärer Algorithmen vorgenommen, u.a. in [CF94], [Can95] und [Swm96]. Für die Parallelisierung der Berechnungen und damit die Unterteilung der Population wurden zwei Ansätze benutzt:

- Änderungen der Struktur der Population und
- neue Methoden zur Selektion der Individuen.

CHIPPERFIELD schlägt eine Unterteilung in folgende Gruppen vor [CF94]:

1. globales Modell (*worker/farmer* Algorithmus oder *farming* Modell),
2. Migrationsmodell und
3. Diffusionsmodell.

CANTU-PAZ unterscheidet die folgenden Modelle [Can95]:

1. globales Modell,
2. grobkörniges Modell (*coarse grained model*),
3. feinkörniges Modell (*fine grained model* ) und
4. hybrides Modell (Kombination von Teilen der ersten drei Modelle).

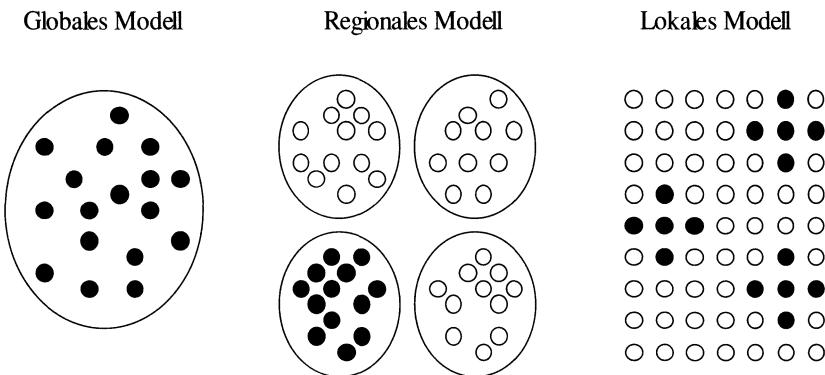
Von SCHWEHM wird ein Vergleich dieser und anderer Klassifikationen vorgenommen [Swm96]. Die von SCHWEHM neu vorgeschlagene Klassifikation unterscheidet danach, in welchem Bereich der Population die Auswahl der Individuen zur Generierung neuer Individuen stattfindet (Reichweite der Selektionsstrategien zur Auswahl der Eltern) und unterteilt sich in:

1. globales Modell,
2. regionales Modell und
3. lokales Modell.

Diese Unterteilung wird als die beste der bekannten Klassifikationen angesehen. Sie ermöglicht eine zutreffende und verständliche Unterscheidung der Populationsmodelle. Außerdem setzt die Benennung nicht auf Nebeneffekte (z.B. Diffusi-

onsmodell: Informationen diffundieren durch die Population) oder Merkmale (z.B. Migration beim Migrationsmodell), die bei den einzelnen Populationsmodellen anzutreffen sind. Statt dessen wird hier ein Unterscheidungsmerkmal genutzt, das bei allen Populationsmodellen auftritt. Dadurch wird eine einheitliche Unterscheidung ermöglicht.

In Abb. 4-1 wird die zur Unterscheidung der Populationsmodelle herangezogene Reichweite der Selektion (Selektionspool) dargestellt (nach [Swm96]). Beim globalen Modell findet die Auswahl der Eltern in der gesamten Population statt. Das regionale Modell beschränkt die Auswahl der Eltern auf voneinander getrennte Teile der Population, die Unterpopulationen genannt werden. Innerhalb der Unterpopulationen findet die Auswahl zwischen allen Individuen statt. Beim lokalen Modell findet die Auswahl der Eltern in lokalen Nachbarschaften statt.



**Abb. 4-1.** Klassifikation der Populationsmodelle nach der Reichweite der Selektion

Der Vorteil der Klassifikation in globales, regionales und lokales Modell zeigt sich auch darin, daß sich die Klassifikationen von CHIPPERFIELD und CANTU-PAZ direkt den entsprechenden Einträgen der Klassifikation von SCHWEHM zuordnen lassen. So entspricht das Migrations- bzw. grobkörnige Modell dem regionalen Modell von SCHWEHM, das Diffusions- bzw. feinkörnige Modell dem lokalen Modell.

## 4.2 Globales Modell

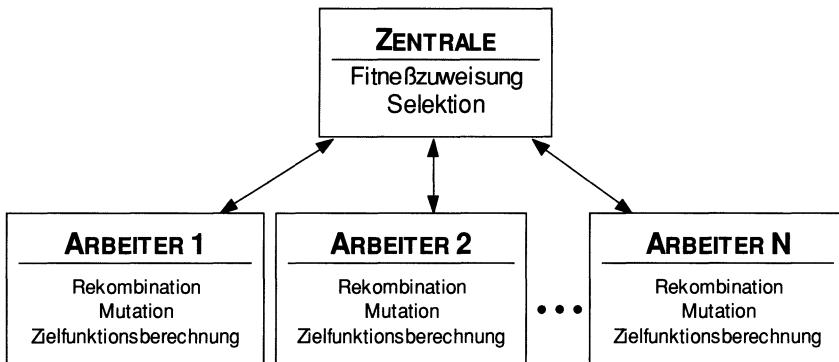
Beim globalen Modell findet keine Unterteilung der Population statt, die Population wird als „unstrukturierte Menge“ betrachtet“ ([Swm96], S.68). Statt dessen wird die dem Evolutionären Algorithmus innewohnende Parallelität, die durch das Arbeiten mit einer Population vorhanden ist, ausgenutzt. Alle Individuen der Population werden zur Fitneßbestimmung verwendet und die Selektion der Individuen zur Reproduktion findet global statt. Individuen aus beliebigen Bereichen der Population können zusammen zur Bildung von Nachkommen ausgewählt

werden. Das globale Modell entspricht dem klassischen Evolutionären Algorithmus.

Die Berechnungen, für welche die gesamte Population benötigt wird – Selektion und Fitneßzuweisung – werden von einer Zentrale ausgeführt. Die restlichen Berechnungen, die jeweils nur ein oder zwei (mehrere) Individuen gleichzeitig verwenden, werden jeweils an einen der vielen gleichartigen Arbeiter verschickt, die jeder für sich und parallel zueinander die Rekombination, Mutation und Zielfunktionsberechnung durchführen (bekannt als synchrone Master-Slave-Struktur, Abb. 4-2). Da diese Berechnungen parallel von mehreren Rechnern ausgeführt werden, läßt sich eine fast lineare Beschleunigung der Gesamtberechnung erzielen.

Für die meisten realen Probleme ist die Zielfunktionsberechnung der mit Abstand aufwendigste Teil der Berechnungen. In diesem Falle können nur die Zielfunktionsberechnungen auf die Arbeiter verteilt werden, der Evolutionäre Algorithmus an sich wird komplett von der Zentrale ausgeführt. Zusätzlicher Vorteil ist, daß statt zweier Individuen nur ein Individuum an den Arbeiter übertragen werden muß.

Falls dagegen der Aufwand zur Zielfunktionsberechnung im Vergleich zum Evolutionären Algorithmus gering ist, kann es zu Engpässen an der Zentrale kommen und das synchrone globale Modell wäre in dieser einfachen Struktur ineffektiv.



**Abb. 4-2.** Populationsmodell: globales Modell (Master-Slave-Struktur)

Die Effizienz der Aufteilung der Berechnungen zwischen Zentrale und Arbeiter hängt entscheidend vom Aufwand der beiden Gruppen von Berechnungen ab. Ein Abgehen von der synchronen Arbeitsweise hin zu einem semi-synchronen oder asynchronen Algorithmus kann einige der Probleme verringern oder lösen [Gol89].

Ein Ansatz ist die Verwendung eines *steady-state* Algorithmus (s. S.59). Bei diesem werden jeweils ein oder zwei Individuen ausgewählt, die dann an einen Arbeiter geschickt werden, wo Rekombination, Mutation und Zielfunktionsberechnung durchgeführt werden. Sobald der Arbeiter fertig ist, werden die neuen

Individuen an die Zentrale übergeben und nachfolgend erhält der Arbeiter neue Individuen. Bei diesem Algorithmus muß nicht auf die Beendigung der Berechnungen der anderen Arbeiter gewartet werden, jeder Arbeiter wird nach Ende der Aufgabe gleich mit einer neuen Berechnung beauftragt, wodurch die Arbeiter selbst immer ausgelastet sind. Voraussetzung ist natürlich, daß die Zentrale in der Lage ist, die Fitneßberechnung, Selektion und das Wiedereinfügen der Individuen in die Population schnell genug auszuführen. Aber auch bei diesem nur teilweise synchronen Algorithmus hängen alle Arbeiter von einer Zentrale ab.

Diesem Problem der Abhängigkeit von einer Zentrale (und damit einer eingeschränkten Robustheit) ist nur mittels eines asynchronen Ansatzes beizukommen. Dabei arbeiten mehrere Prozessoren parallel zueinander, wobei jeder einen kompletten Evolutionären Algorithmus ausführt. Ein Austausch zwischen diesen Prozessen findet entweder von Zeit zu Zeit in einer bestimmten Struktur statt (ähnlich wie nachfolgend bei der Migration beschrieben) oder alle Prozesse greifen gemeinsam auf einen Speicherbereich zu, über den diese Prozesse dann miteinander verbunden sind. Selbst wenn hier einige der Prozessoren ausfallen, läuft die gesamte Optimierung weiter.

ROBERTSON [Rob87] berichtet von einer vollständigen Implementierung des globalen Modells, für die eine *Connection Machine CM-1* mit 65536 Prozessoren eingesetzt wurde. Im Gegensatz zu anderen Implementierungen des globalen Modells wird in [Rob87] auch die Selektion parallel ausgeführt. Weitere Implementierungen des globalen Modells werden unter anderem in [FH91], [HM94] und [Swm96] berichtet.

Das globale Modell ist ein einfacher Weg, sehr lange Rechenzeiten einer Optimierung auf ein erträgliches Maß zu reduzieren. Außerdem lässt sich die Verteilung der Zielfunktionsberechnungen auf mehrere Rechner auch bei allen anderen Unterteilungen der Population anwenden.

## 4.3 Lokales Modell

Beim lokalen Modell (auch Diffusionsmodell oder Nachbarschaftsmodell genannt) wird jedes Individuum als ein separates Element betrachtet. Eine Interaktion findet nur mit den Individuen in der näheren Umgebung statt (und nicht mit denen der Unterpopulation oder der gesamten Population). Die Reichweite der Selektion ist beim lokalen Modell auf die festgelegte Nachbarschaft eines Individuums beschränkt, die Nachbarschaft entspricht dem Selektionspool. Dadurch findet eine fließende Unterteilung der Gesamtpopulation statt. Die Individuen sind untereinander durch die Distanz zwischen ihnen isoliert, je größer die Distanz, um so größer ist die Isolation. Dieser Effekt wird als **Isolation durch Distanz** bezeichnet.

Das lokale Modell wird durch eine Anzahl von Parametern bestimmt:

- Topologie oder Struktur der Nachbarschaft und Distanz zwischen den Nachbarn,
- Selektion innerhalb der Nachbarschaft,
- Wiedereinfügen von Nachkommen innerhalb der Nachbarschaft.

Die Topologie der Nachbarschaft und die Distanz zu den Nachbarn bestimmen die Art und Größe einer Nachbarschaft. Durch die Wahl dieser Parameter kann beeinflußt werden, wie schnell sich Informationen innerhalb der Population ausbreiten können. Dadurch wird bestimmt, ob sich die Population eher wie lose miteinander gekoppelte Bereiche oder wie stark verbundene Bereiche (ähnlich einer Gesamtpopulation mit dem globalen Modell) entwickelt.

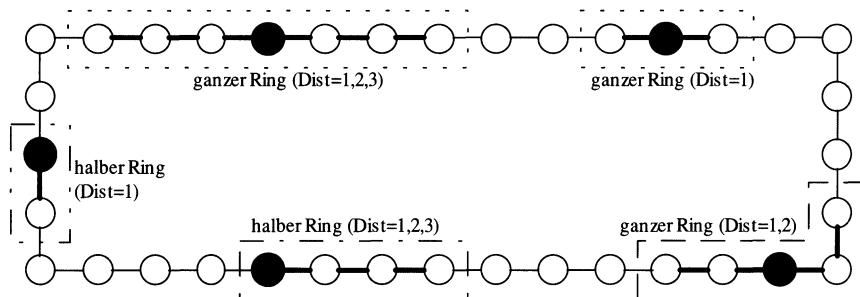
Die Selektion innerhalb der Nachbarschaft und das Wiedereinfügen der Nachkommen in die Nachbarschaft verläuft ähnlich wie bei den entsprechenden Verfahren für eine Population. Auf Grund der oft sehr kleinen Größe der Nachbarschaften können jedoch zusätzlich Spezialfälle betrachtet werden.

### 4.3.1 Topologie von Nachbarschaften

In diesem Unterabschnitt wird eine Auswahl der gebräuchlichsten Topologien für lokale Modelle vorgestellt. In Unterabschn. 4.3.2 wird eine allgemeine Beschreibung für Nachbarschaftstopologien gegeben, in die alle aufgeführten Topologien des regionalen und lokalen Modells als Spezialfälle eingeordnet werden können.

Die gebräuchlichsten Topologien von Nachbarschaften können in die folgenden Bereiche unterteilt werden:

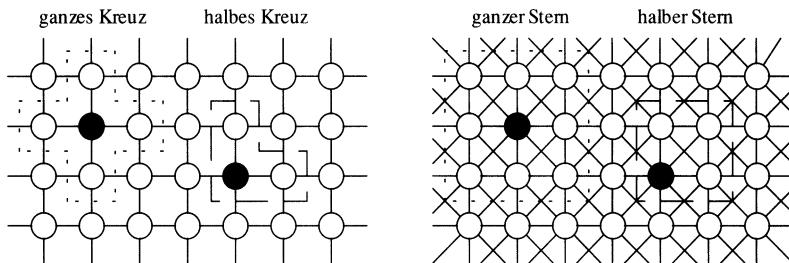
- lineare Topologie (eindimensional):
  - ganzer und halber Ring, Abb. 4-3.
- zweidimensionale Topologien (2-D):
  - ganzes und halbes Kreuz, Abb. 4-4 links,
  - ganzer und halber Stern, Abb. 4-4 rechts,
  - schiefer Stern, Abb. 4-5 links,
  - Kreis, Abb. 4-5 rechts.
- drei- und höherdimensionale Topologien mit entsprechenden Erweiterungen der obigen Varianten.



**Abb. 4-3.** Lineare Nachbarschaftstopologien: ganzer und halber Ring mit unterschiedlicher Distanz

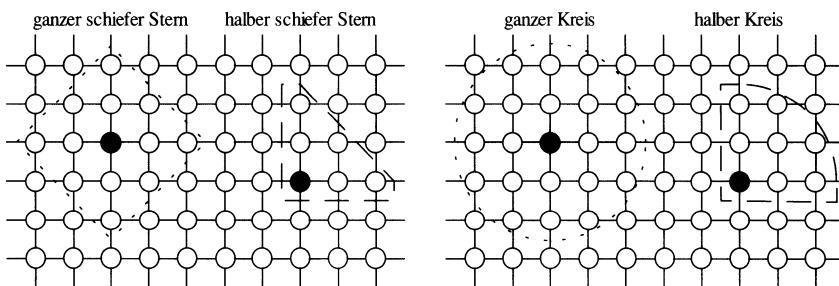
Durch die Distanz *Dist* zu den Nachbarn wird zusammen mit der Topologie die Größe der Nachbarschaft festgelegt. Die Größe der Nachbarschaft bestimmt

einmal den Selektionsdruck, der bei der Auswahl von Individuen in dieser Nachbarschaft herrscht. Außerdem wird durch die Größe der Nachbarschaft die Geschwindigkeit der Ausbreitung von Informationen innerhalb der Population gesteuert. Bei großen Nachbarschaften kommt es zu einer schnellen Ausbreitung von Information, bei kleinen Nachbarschaften zu einer langsamen.



**Abb. 4-4.** Zweidimensionale Nachbarschaftstopologien; links: ganzes und halbes Kreuz, rechts: ganzer und halber Stern, alle mit einer Distanz von 1

Eine hohe Ausbreitungsgeschwindigkeit hat zur Folge, daß sich das Verhalten des lokalen Modells dem des globalen Modells annähert. Bei einer langsamen Informationsausbreitung kommt es zu einer fließenden Isolation räumlich entfernter Bereiche der Population, zu einer 'Isolation durch Distanz'.



**Abb. 4-5.** Zweidimensionale Nachbarschaftstopologien; links: ganzer und halber schiefer Stern mit einer max. Distanz von 2, rechts: ganzer und halber Kreis mit einer max. Distanz von  $\sqrt{5}$

### 4.3.2 Allgemeine Beschreibung von Nachbarschaften

Bei der Beschreibung der Nachbarschaftstopologien des lokalen Modells in Unterabschn. 4.3.1 und der Migrationstopologien des regionalen Modells in Unterabschn. 4.4.2 werden für die einzelnen Topologien und Nachbarschaften verschiedene Namen verwendet. Diese Namen haben sich in der bisherigen Anwen-

dung durchgesetzt. Sie haben aber den Nachteil, daß ein direkter Vergleich der verschiedenen Topologien schwierig und eine Erweiterung auf beliebige Dimensionen nicht offensichtlich ist. In diesem Unterabschnitt wird eine allgemeine Definition von Nachbarschaften gegeben, die sich auf beliebige Dimensionen ausdehnen läßt und alle bisherigen Topologien als Spezialfälle enthält.

Die Beschreibung wird auf Knoten (Unterpopulationen beim regionalen Modell bzw. Individuen beim lokalen Modell) und Kanten (Verbindungen zwischen Knoten) bezogen. Ziel dieser Beschreibung ist es, die Indizes der Knoten zu errechnen, die zu einem gegebenen Knoten und einer gewünschten Nachbarschaftstopologie die Nachbarn dieses Knotens sind.

Für die Beschreibung der Nachbarschaften werden die folgenden Dinge vorausgesetzt:

- die Knoten liegen auf einem quadratischen Gitter,
- die Weite des Gitters (Abstand zwischen direkt benachbarten Knoten) beträgt 1,
- jeder Knoten hat einen eindeutigen Index, der aus einer Zahl besteht, die seine Position in einem eindimensionalen Vektor aller Knoten angibt,
- die Anordnung der Knoten erfolgt in einem verallgemeinerten Torus, d.h. es gibt keine Randknoten, sondern nur innere Knoten (endlich unbegrenzte Menge). Für eine eindimensionale Anordnung bedeutet dies einen Ring, in einer zweidimensionalen Anordnung einen Torus und in höheren Dimensionen einen Hypertorus.

Für eine allgemeine Beschreibung ist es notwendig, daß die Knoten in beliebig vielen Dimensionen angeordnet werden können und die Ausdehnung jeder Dimension beliebig festgelegt werden kann. Im folgenden wird die Anzahl der Dimensionen der Anordnung durch *Dim* angegeben werden und die Anzahl der Knoten in jeder Dimension ist im Vektor *LenDim* (*length of dimension*) enthalten. Jede niedrigere Dimension läßt sich aus einer höheren Dimension ableiten, indem die Anzahl der Knoten aller höheren Dimensionen auf 1 gesetzt wird.

Die Anzahl der Knoten einer Anordnung *AnzKnoten* läßt sich aus der Anzahl der Knoten pro Dimension berechnen:

$$\text{AnzKnoten} = \prod_{i=1}^{\text{Dim}} \text{LenDim}(i) \quad (4-1)$$

Der Index eines Knotens in der eindimensionalen Indexierung, *IxKnoten*, läßt sich aus seiner mehrdimensionalen Indexierung, *IxDimKnoten*, ermitteln. *IxDimKnoten* ist ein Vektor der Länge *Dim*.

$$\text{IxKnoten} = \sum_{i=1}^{\text{Dim}} \left( \text{IxDimKnoten}(i) \cdot \begin{cases} 1 & i = 1 \\ \prod_{j=1}^{i-1} \text{LenDim}(j) & i > 1 \end{cases} \right) \quad (4-2)$$

Ausgehend von einem Knoten kann der Index jedes anderen Knotens in der Anordnung relativ angegeben werden. Dies kann mehrdimensional, *IxRelDimKnoten*, oder eindimensional, *IxRelKnoten*, erfolgen. Eine Umrechnung zwischen beiden Angaben ist mit Gl. 4-2 möglich. Bei der mehrdimensionalen Angabe enthält jedes Element von *IxRelDimKnoten* den rechtwinkligen Abstand

in der jeweiligen Dimension. Eine Angabe von  $IxRelDimKnoten=[3,4,2]$  bedeutet, daß die betrachteten Knoten in Dimension 1 einen Abstand von 3, in Dimension 2 einen Abstand von 4 und in Dimension 3 einen Abstand von 2 haben. Wenn der Abstand von einem Knoten zum anderen gerichtet betrachtet wird, nehmen die Angaben auch negative Werte an.

Der direkte Abstand zwischen zwei Knoten läßt sich einmal aus der absoluten Position beider Knoten oder aus der relativen Position der Knoten zueinander berechnen:

$$\text{Abstand} = \sqrt{\sum_{i=1}^{\text{Dim}} (IxDimKnoten}_1(i) - IxDimKnoten_2(i))^2} / \sqrt{\sum_{i=1}^{\text{Dim}} IxRelDimKnoten(i)^2} \quad (4-3)$$

Für jeden Knoten lassen sich bestimmen:

- die relativen Indizes zu jedem anderen Knoten ( $IxRelDimKnoten$  für jeden Knoten),
- der Abstand zu jedem anderen Knoten ( $\text{Abstand}$  für jeden Knoten),
- eine Reihenfolge der nächsten Knoten in Abhängigkeit vom  $\text{Abstand}$  der Knoten.

Auf der Grundlage dieser Variablen und Begriffe lassen sich jetzt verschiedenste Nachbarschaften definieren, die über alle Dimensionen skaliert werden können. Für die eindeutige Auswahl einer Nachbarschaft reichen dann die folgenden drei Parameter aus:

- Anzahl der Knoten in jeder Dimension ( $\text{LenDim}$ ),
- Typ der Nachbarschaftstopologie (Hyperkreuz, Hyperstern usw.) und
- maximaler Abstand,  $\text{MaxAbstand}$ , zu den Nachbarn (Ausdehnung der Nachbarschaft).

Die folgenden Definitionen entsprechen den vollen Nachbarschaften. Aus einer vollen Nachbarschaft läßt sich die halbe Nachbarschaft ableiten, indem alle Knoten aus der Nachbarschaft entfernt werden, in denen mindestens einer der relativen Indizes (Element von  $IxRelDimKnoten$ ) negativ ist.

Für die Benennung werden meistens die Namen der Topologien verwendet, die für die zweidimensionale Topologie gebräuchlich sind. Der Vorsatz Hyperdrückt aus, daß dieser Begriff über alle Dimensionen gültig sein soll. Außerdem wird angegeben, welcher gebräuchlichen Topologie dies in einer und zwei Dimensionen entspricht.

### 1. Hyperkreuz:

- alle Knoten, bei denen nur ein relativer Index (ein Element in  $IxRelDimKnoten$ ) verschieden von Null und der Betrag dieses Index kleiner gleich  $\text{MaxAbstand}$  ist:

$$\exists j \quad IxRelDimKnoten(j) \neq 0 \cap |IxRelDimKnoten(j)| \leq \text{MaxAbstand} \cap \\ \forall i \quad IxRelDimKnoten(i) = 0 \quad i, j \in 1 \dots \text{Dim}, i \neq j$$

- 1-D: Ring, Abb. 4-3 und 4-7; 2-D: Kreuz, Abb. 4-4 links und Abb. 4-8,
- die Leitertopologie, Abb. 4-9, kann als Spezialfall des zweidimensionalen Kreuzes betrachtet werden, bei der die Ausdehnung in der zweiten Dimension genau zwei Knoten beträgt.

## 2. Hyperstern:

- alle Knoten, bei denen der Betrag aller relativen Indizes (alle Elemente in  $IxRelDimKnoten$ ) kleiner gleich  $MaxAbstand$  ist:

$$\forall i \quad |IxRelDimKnoten(i)| \leq MaxAbstand \quad i \in 1 \dots Dim;$$

- 1-D: Ring; 2-D: Stern, Abb. 4-4 rechts.

## 3. schiefer Hyperstern:

- alle Knoten, bei denen die Summe der Beträge der relativen Indizes kleiner gleich ( $MaxAbstand \cdot Dim$ ) ist:

$$\sum_{i=1}^{Dim} |IxRelDimKnoten(i)| \leq (MaxAbstand \cdot Dim);$$

- 1-D: Ring; 2-D: schiefer Stern, Abb. 4-5 links.

## 4. Hyperkugel:

- alle Knoten, deren Abstand kleiner als  $MaxAbstand$  ist,
- ein Spezialfall der Hyperkugel ist die Angabe der Anzahl der nächsten Nachbarn, die in der Nachbarschaft enthalten sind (nicht der maximale Abstand ist der Parameter, sondern die Anzahl der zu verwendenden nächstgelegenen Knoten),
- 1-D: Ring; 2-D: Kreis, Abb. 4-5 rechts.

Mit diesen allgemein definierten Nachbarschaftstopologien lassen sich die bekannten und gebräuchlichen Nachbarschaftstopologien angeben und können dadurch als Grundlage weiterer Untersuchungen dienen.

Für einen Vergleich der verschiedenen Nachbarschaften müssen einige Maßzahlen definiert werden, die bestimmte Eigenschaften einer jeden Nachbarschaft wiedergeben. Dabei ist  $N$  die Anzahl der Knoten einer Nachbarschaft.

Radius einer Nachbarschaft:

$$RadiusNach = \sqrt{\frac{\sum_{i=1}^{Dim} \sum_{j=1}^N (IxDimKnoten_j^i - mean(IxDimKnoten^i))^2}{N}} \quad (4-4)$$

*mean:* Mittelwert

Durch Verwendung aller Knoten einer Anordnung in Gl. 4-4 lässt sich der Radius einer Anordnung  $RadiusAnord$  berechnen. Der Quotient aus  $RadiusNach$  und  $RadiusAnord$  (Radienquotient) ist eine Maßzahl, die zum Vergleich einiger Anordnungen und Nachbarschaften untereinander verwendet werden kann.

Die hier vorgestellte allgemeine Beschreibung der Nachbarschaftstopologien und ihre einfache Definition in Abhängigkeit weniger Parameter sollte als Grundlage für weitere Arbeiten dienen.

### 4.3.3 Selektion in Nachbarschaft

Die Auswahl der Individuen zur Bildung der Nachkommen erfolgt beim lokalen Modell in zwei Schritten. Zuerst wird über die Population verteilt eine Anzahl von Individuen als Zentrum einer Nachbarschaft festgelegt. Um diese Individuen wird dann jeweils die Nachbarschaft definiert. Im zweiten Schritt werden aus den Nachbarschaften die Individuen zur Produktion der Nachkommen ausgewählt.

Jeder dieser Selektionsschritte kann mit unterschiedlichen Methoden realisiert werden:

1. Auswahl der Zentren der Nachbarschaften:

- gleichmäßig verteilt zufällig oder fitneßabhängig wird eine Anzahl von Individuen (*generation gap*) ausgewählt. Die Auswahl kann mit einem der globalen Selektionsverfahren aus Abschn. 3.2 durchgeführt werden. Die Auswahl eines Teils der Individuen ist nur bei einer seriellen Implementierung sinnvoll, da ein globaler Prozeß (globale Selektion) diesen Schritt durchführen muß,
- jedes Individuum ist Zentrum einer Nachbarschaft (meist bei Implementierungen auf parallelen Rechnern verwendet).

2. Auswahl innerhalb einer Nachbarschaft:

- das Individuum im Zentrum ist ein Elter, die weiteren Eltern (der zweite Elter) werden nach einem der folgenden Verfahren ausgewählt:
  - gleichmäßig zufällig verteilte Auswahl unter den Nachbarn,
  - fitneßproportionale Auswahl unter den Nachbarn (Anwendung eines der Selektionsverfahren aus Abschn. 3.2),
  - bester Nachbar (beste Nachbarn),
- alle Eltern werden in der Nachbarschaft durch ein Selektionsverfahren ausgewählt (alle in Abschn. 3.2 vorgestellten Verfahren sind anwendbar).

In der Anwendung sind zwei Bereiche zu unterscheiden: Verwendung einer parallelen und einer seriellen Implementierung.

Bei einer parallelen Implementierung, bei der jedes Individuum einem Prozessor zugeordnet ist, wird in jeder Generation ein neues Individuum pro Prozessor berechnet. Deshalb bietet es sich an, daß jedes Individuum Zentrum einer Nachbarschaft ist und sich einen zweiten Elter in der Nachbarschaft nach einem einfachen und schnellen Verfahren sucht, z.B. bester oder zufälliger Nachbar.

Bei einer seriellen Implementierung dagegen kann die Anzahl der Nachkommen pro Generation kleiner als die Anzahl der Individuen in der Population sein (definiert durch den Parameter *generation gap*). Für die Auswahl der Eltern ergeben sich dadurch die oben vorgestellten Möglichkeiten, die eine erweiterte Kontrolle des Selektionsdruckes zulassen.

### 4.3.4 Wiedereinfügen in Nachbarschaft

Nachdem die Nachkommen gebildet und bewertet wurden, müssen diese wieder in die Population eingefügt werden. Durch die Verwendung einer lokalen Nachbarschaft, in der alle Prozesse stattfinden, müssen die Nachkommen einer Nach-

barschaft in genau diese Nachbarschaft eingefügt werden. Dieser Prozeß wird als lokales Wiedereinfügen bezeichnet (im Gegensatz zum globalen/regionalen Wiedereinfügen aus Abschn. 3.5).

Das Wiedereinfügen der Nachkommen in die Nachbarschaft kann nach zwei Kriterien gegliedert werden: Welche Nachkommen werden in die Nachbarschaft eingefügt? Welche Individuen der Nachbarschaft werden dadurch ersetzt?

1. Auswahl der Nachkommen:

- alle Nachkommen werden eingefügt,
- nur Nachkommen, die besser als die schlechtesten Individuen der Nachbarschaft sind, werden eingefügt,
- nur Nachkommen, die besser als die Eltern der Nachbarschaft sind, werden eingefügt.

2. Ersetzen der Individuen der Nachbarschaft:

- Nachkommen ersetzen gleichmäßig zufällig verteilt Individuen der Nachbarschaft,
- Nachkommen ersetzen die schlechtesten Individuen der Nachbarschaft,
- Nachkommen ersetzen die Eltern der Nachbarschaft.

Das Verfahren zur Auswahl der Nachkommen bestimmt den Selektionsdruck, der auf die Nachkommen wirkt, bevor diese in die Population eingefügt werden. Das Einfügen aller Nachkommen hat keinen Selektionsdruck zur Folge. Wenn nur Nachkommen eingefügt werden, die besser als die schlechtesten Individuen sind, so kommt es zu einer kontinuierlichen Verbesserung der durchschnittlichen Güte der Population. Wenn die Nachkommen besser als die Eltern sein müssen, dann entspricht dies dem höchsten Selektionsdruck, da die Eltern nicht die schlechtesten Individuen der Nachbarschaft sind. Durch diesen hohen Selektionsdruck kann es aber zu einem Steckenbleiben der Entwicklung der Population kommen, denn in den meisten Fällen ist nur ein kleiner Teil der Nachkommen besser als die Eltern.

Das Ersetzen der Individuen in der Nachbarschaft bestimmt, ob eine elitist Methode angewendet wird oder nicht. Beim gleichmäßigen Ersetzen der Individuen in der Nachbarschaft können die besten Individuen durch schlechtere Nachkommen ersetzt werden. Dasselbe gilt, wenn die Eltern ersetzt werden und die Nachkommen nicht besser als die Eltern sein müssen. Nur wenn die schlechtesten Individuen der Nachbarschaft ersetzt werden, ist gewährleistet, daß die besten Individuen in der Population überleben bzw. nur durch bessere Nachkommen ersetzt werden.

Für die Anwendung wird deshalb empfohlen, in der Nachbarschaft die schlechtesten Individuen durch die Nachkommen zu ersetzen. Als Nachkommen können die ausgewählt werden, die besser als die schlechtesten Individuen sind (empfohlene Methode) oder alle Nachkommen können verwendet werden.

#### 4.3.5 Analyse des lokalen Modells

Durch die auf Nachbarschaften beschränkte Selektion findet eine lokale Interaktion zwischen den Individuen statt, es ergibt sich eine ‘Isolation durch Distanz’. In Abhängigkeit der Nachbarschaften kann die Auswirkung der lokalen Interak-

tion (der Austausch von Informationen) auf die gesamte Population beeinflußt werden. Je größer die Ausdehnung der Nachbarschaft, um so größer sind die Überlappungen zwischen den einzelnen Nachbarschaften und um so schneller werden Informationen in der gesamten Population ausgetauscht. Durch die Größe der Nachbarschaft kann somit zwischen einer schnellen Ausbreitung von Informationen oder der Erhaltung der Verschiedenartigkeit der einzelnen Bereiche der Population (langsame Ausbreitung) gewählt werden.

Meist wird eine hohe Verschiedenartigkeit innerhalb der Population angestrebt, da hiermit Probleme, wie die vorzeitige Konvergenz (*premature convergence*) in ein lokales Minimum, verhindert werden können. Das lokale Modell mit einer kleinen Nachbarschaft arbeitet oft besser als das lokale Modell mit einer großen Nachbarschaft. Trotzdem ist es wichtig, daß die Verbindungen innerhalb der Population erhalten bleiben. Ähnliche Ergebnisse werden aus Berechnungen in [VSB92] abgeleitet (Verwendung einer kleinen zweidimensionalen Struktur, halber Stern, mit einer Distanz von 1; bei größeren Populationen (>100 Individuen) sollte eine entsprechend größere zweidimensionale Nachbarschaft bzw. eine größere Distanz verwendet werden).

Im Verlauf der Anwendung des lokalen Modells bilden sich über die Generationen virtuelle Inseln in der Population (Visualisierung der Zielfunktionswerte der Individuen in ihrer Nachbarschaft). Die virtuellen Inseln verändern ständig ihre Größe und Position. Einen Eindruck davon vermittelt Abb. 5-11, S.133, die sich im Verlauf der Generationen verändernde virtuelle Inseln zeigt.

Der erfolgreiche Einsatz des lokalen Modells wurde seit Ende der 80er Jahre oft berichtet. Der von der Gruppe um MÜHLENBEIN ([MGK88], [Müh89], [MSB91]) entwickelte Parallel Genetic Algorithm (PGA) verwendet lokale Selektion in einer zweidimensionalen Struktur (ganzer Stern mit einer Distanz von 1, dadurch 4 Nachbarn). Durch die Verteilung der Individuen und die starke Lokalität wurde eine fast 100%ige Auslastung der Prozessoren und eine annähernd lineare Beschleunigung der Berechnungen erreicht. Da sich der PGA leicht auf viele Prozessoren verteilen läßt, können sehr große Probleme gelöst werden.

Eine spezielle Struktur der Nachbarschaft wird von GORGES-SCHLEUTER in [GS91] vorgestellt, genannt *ASPARAGOUS*. Die Individuen werden auf den Eckpunkten einer ringförmigen Leiter angeordnet (ähnlich der Leiterstruktur in Abb. 4-9, S.92). Neben der guten Performanz wird eine superlineare Beschleunigung gegenüber einer einzigen Population gleicher Individuenzahl erreicht. Unter anderem werden neue Optima für sehr große Probleme gefunden.

MANDERICK und SPIESSENS ([MS89], [SM91]) berichten von der Implementierung des lokalen Modells eines Genetischen Algorithmus auf einem massiv parallelen Rechner. Dabei untersuchen sie den Einfluß der gewählten Selektionsmethode.

COLLINS und JEFFERSON [CJ91] untersuchen verschiedene Selektionsmethoden und vergleichen das lokale Modell mit einer einheitlichen Gesamtpopulation. Das lokale Modell ist deutlich schneller und robuster als eine Gesamtpopulation. Besonders bei der Lösung schwieriger Probleme ist die Lokalität des lokalen Modells von Vorteil, da sich verschiedene Nischen mit Lösungen etablieren können, was zu einer robusteren Suche führt.

DAVIDOR [Dav91a] verwendet ein zweidimensionales lokales Modell (mit ganzem Stern und Distanz 1, dadurch 8 Nachbarn) und beschreibt die Herausbil-

dung lokaler Nischen und Inseln mit homogeneren Individuen und Fitneßwerten, die jeweils nahe der lokalen Optimalität liegen. Im Verlauf der Optimierung nähern sich diese Inseln dem lokalen Optimum an und einige Inseln mit besonders guten Individuen breiten sich über große Teile der Population aus, wodurch Inseln mit Individuen schlechterer Fitneß aussterben. DAVIDOR verweist darauf, daß sich seine Beschreibung auf größere Dimensionen verallgemeinern läßt.

GORDON und WHITLEY [GW91] vergleichen verschiedene parallele Implementierungen an einer Reihe von Testfunktionen, wobei das lokale Modell (*cellular Genetic Algorithm*) über alle Testfunktionen eine etwas schlechtere Performanz als die Migrationsmodelle hat, bei einigen der Testfunktionen ist es dagegen wesentlich besser.

Von SCHWEHM wird in [Swm96] eine Analyse verschiedener paralleler Implementierungen einschließlich des lokalen Modells vorgenommen. Er untersucht u.a. den Einfluß der Größe der Nachbarschaft und der Nachbarschaftsstruktur des lokalen Modells auf das Konvergenzverhalten ([Swm96], S.116-124). Für kleine Nachbarschaftsgrößen werden weniger Zielfunktionsaufrufe benötigt, als für große Nachbarschaften. Allerdings darf die Nachbarschaft auch nicht zu klein sein. Bei genauerer Untersuchung zeigt sich, daß bei einer zu großen Nachbarschaft die Diversität in der Population sehr schnell abnimmt, was zu einer Verlangsamung der Suche führt. Bei einem Vergleich verschiedener Nachbarschaftsstrukturen können keine eindeutigen Unterschiede zwischen den verschiedenen Nachbarschaften festgestellt werden. Dies zeigt sich noch deutlicher, wenn für jede Nachbarschaft einige der weiteren Parameter optimal eingestellt werden. Dann sind die vorher noch erkennbaren Unterschiede fast vollständig verschwunden. Nur zwei Strukturen (binärer Baum und Ring) sind schlechter als die anderen: beide haben einen sehr großen Durchmesser der Anordnung sowie eine kleine Anzahl von Nachbarn, die jeweils einen geringen Abstand zueinander haben. Bei anderen Strukturen (*shuffle-exchange*) mit vergleichbarem Durchmesser und vergleichbarer Anzahl von Nachbarn, die aber ein besseres Konvergenzverhalten zeigen, sind die Nachbarn dagegen weiter voneinander entfernt. Daraus kann wiederum die Schlußfolgerung gezogen werden, daß die Bereiche der Population nicht zu stark voneinander isoliert sein dürfen. Die Verbindung zwischen allen Bereichen der Population muß gewährleistet sein.

Von SARMA und DEJONG wird in [SDJ96] eine Untersuchung des Einflusses der Nachbarschaftsstruktur und -größe auf die Selektionsintensität innerhalb der Population vorgenommen. Die Untersuchung ist auf zweidimensionale quadratische Anordnungen unterschiedlicher Größe beschränkt. Die untersuchten Nachbarschaftsstrukturen sind alle volle Strukturen (Kreuz mit maximalen Abstand von 1 (als L5 bezeichnet) und 2 (L9), Stern mit maximalem Abstand von 1 (C9) und schiefer Stern mit maximalem Abstand von 2 (C13)). SARMA und DEJONG kommen zu der Schlußfolgerung, daß der Quotient (Radienquotient) aus dem Radius der Nachbarschaft *RadiusNach* und dem Radius der Anordnung *RadiusAnord* zur Charakterisierung der Selektionsintensität in der Population ausreichend ist.

Diese Aussage läßt sich für die von SARMA und DEJONG verwendeten Anordnungen und Nachbarschaftsstrukturen verwenden, die Spezialfälle und Vereinfachungen darstellen. Außerdem kann sie zum Vergleich von Anordnungen und Strukturen verwendet werden, die jeweils nur in ihrer Ausdehnung verändert

werden, wobei die relative Größe zueinander erhalten bleibt. Aber schon beim Vergleich von vollen Nachbarschaftsstrukturen mit halben Nachbarschaftsstrukturen ist der einfache Parameter Quotient aus dem Radius der Nachbarschaft und dem Radius der Anordnung nicht mehr ausreichend. Weitere Größen müssen einbezogen werden, wenn die Anordnung nicht quadratisch ist. In dieser Richtung ist noch Arbeit zu leisten.

Die Selektionsintensität in der Population ist unter Verwendung derselben Selektionsintensität in den jeweiligen Selektionspools bei einer lokalen Selektion und kleinen Radienquotienten kleiner, als bei einer globalen Selektion. Dies ist im Prinzip der lokalen Selektion begründet. Mit steigendem Radienquotient nähert sich die Selektionsintensität der lokalen Selektion dem der globalen Selektion an (SARMA und DEJONG stellen fest, daß bei einem Radienquotient von 0,5 die lokale Selektion die Selektionsintensität der globalen Selektion erreicht.). Ziel der Anwendung des lokalen Modells kann es nicht sein, dieselbe Selektionsintensität wie bei einer globalen Population mit gleich vielen Individuen zu erreichen. Bei einem Radienquotient von 0,5 und größer ist in dem Sinne keine lokale Selektion mehr vorhanden, die 'Isolation durch Distanz' ist verschwindend klein.

Für die Auswahl der zu verwendenden Nachbarschaften und deren Vergleich untereinander ist die Angabe der Selektionsintensität in der Population interessant und der Radienquotient bildet einen ersten Anhaltspunkt. Weitere Untersuchungen für allgemein definierte Nachbarschaften und Anordnungen beliebiger Ausdehnung müssen noch durchgeführt werden.

Die in Unterabschn. 4.3.2 vorgestellte allgemeine Beschreibung von Nachbarschaften sollte als Grundlage für zukünftige Untersuchungen mit dem lokalen Modell verwendet werden.

## 4.4 Regionales Modell

Beim regionalen Modell (auch als Migrationsmodell oder feinkörniges Modell bezeichnet) erfolgt eine Beschränkung der Selektion auf voneinander getrennte Teile der Population. Dadurch erfolgt eine Unterteilung der Gesamtpopulation in einzelne Unterpopulationen. Diese Unterpopulationen entwickeln sich unabhängig voneinander. In jeder Unterpopulation läuft ein eigener Evolutionärer Algorithmus. Die Fitneßberechnung und Selektion, die sonst in der Gesamtpopulation zwischen allen Individuen stattfindet, wird beim regionalen Modell nur zwischen den Individuen der jeweiligen Unterpopulation durchgeführt; es findet eine **regionale** Fitneßberechnung und Selektion statt.

Diese durch die Reichweite der Selektion hervorgerufene Unterteilung der Population wird aber nicht über alle Generationen aufrecht erhalten.<sup>1</sup> Von Zeit zu Zeit wird ein Austausch von Informationen zwischen den Unterpopulationen

---

<sup>1</sup> Die vollständige Trennung der Unterpopulationen würde einer parallelen Ausführung mehrerer Evolutionärer Algorithmen nebeneinander mit jeweils verringriger Individuenzahl entsprechen. Dies hätte eine deutlich schlechtere Leistungsfähigkeit des Gesamtautomaten zur Folge.

durchgeführt. Die Art und Weise sowie der Umfang der ausgetauschten Informationen haben einen großen Einfluß auf die Arbeitsweise des regionalen Modells.

Solange kein Austausch von Informationen stattfindet, entwickeln sich die Unterpopulationen unabhängig voneinander. Dies geschieht über eine Reihe von Generationen und wird als Isolationszeit bezeichnet. Nach dieser Isolationszeit findet der Austausch von Informationen durch den Austausch von Individuen zwischen den Unterpopulationen statt. Dieser Vorgang wird als Migration bezeichnet. Nach der Migration entwickeln sich die Unterpopulationen wieder unabhängig voneinander, bis es nach Ablauf der Isolationszeit wieder zu einer Migration von Individuen kommt.

Durch das regionale Modell wird eine direkte Unterteilung der Population eingeführt. Diese ist durch folgende Parameter gekennzeichnet:

- Anzahl der Unterpopulationen und
- Größe der einzelnen Unterpopulationen.

Durch diese beiden Größen ist gleichzeitig die Gesamtzahl der Individuen der Population definiert.

Zusätzlich zu den bisherigen Operatoren Evolutionärer Algorithmen wird durch das regionale Modell der neue Operator **Migration** eingeführt. Der Migrationsoperator wird durch die folgenden Parameter bestimmt:

- Topologie der Unterpopulationen untereinander für die Migration (Migrationstopologie oder -struktur),
- Häufigkeit der Migration (Migrationsintervall, entspricht der Isolationszeit),
- Anteil der Individuen, die pro Migration zwischen den Unterpopulationen ausgetauscht werden (Migrationsrate) und
- Auswahl der Individuen für den Austausch (Migrationsauswahl, Migrationsauslese).

Durch die Wahl dieser Parameter kann beeinflußt werden, wieviele Informationen zwischen den Unterpopulationen ausgetauscht werden und wie schnell sich Informationen in der gesamten Population ausbreiten können. Damit wird beeinflußt, ob sich die Unterpopulationen eher unabhängig voneinander entwickeln oder ob sie sich ähnlich wie eine große Gesamtpopulation verhalten.

Durch eine Unterteilung der Population und einen nicht zu hohen Austausch von Informationen ist es möglich, die Verschiedenartigkeit der Individuen in einer Population länger zu erhalten, als dies mit einer vergleichbaren Gesamtpopulation möglich wäre. In einer einheitlichen (*panmictic*) Population werden die Individuen nach jeder Generation untereinander ähnlicher. Dieser Effekt wird durch die genetische Drift (s. Selektionsvarianz bei der Erläuterung der einzelnen Selektionsverfahren, Abschn. 3.2) hervorgerufen. Dies führt zu einem zunehmenden Nachlassen der Effektivität des Rekombinationsoperators. Im Gegensatz zur einheitlichen Population entwickeln sich bei einer Unterteilung der Population die voneinander getrennten Bereiche mehr oder weniger unabhängig voneinander. Die Individuen werden durch die genetische Drift insgesamt weniger beeinflußt, da unter den Unterpopulationen die Verschiedenartigkeit länger erhalten bleibt. Dies hat außerdem eine bessere Erkundung unterschiedlicher Bereiche des Suchraumes zur Folge.

Ein Äquivalent für das regionale Modell findet sich in der natürlichen Evolution problemlos. So tendieren Arten dazu, sich in Untergruppen oder Unterpopulationen fortzupflanzen. Diese Unterpopulationen sind voneinander geographisch oder durch andere Schranken (kulturell, ökonomisch) lose oder stark getrennt. Nur von Zeit zu Zeit wandern einzelne Individuen von einer Unterpopulation zu einer anderen und pflanzen sich in der neuen Unterpopulation fort. Es ist wesentlich wahrscheinlicher, daß sich Individuen innerhalb ihrer Unterpopulation fortpflanzen, als daß dies zwischen Individuen verschiedener Unterpopulationen geschieht.

#### 4.4.1 Anzahl und Größe der Unterpopulationen

Die Anzahl und die Größe der Unterpopulationen,  $AnzUnderPop$  und  $AnzIndUnderPop$ , und damit die Gesamtzahl der zu verwendenden Individuen, sind stark von der Problemstellung abhängig und können allgemein nicht festgelegt werden. Im Verlauf der Arbeiten mit Evolutionären Algorithmen erwiesen sich einige Richtwerte als gute Startwerte für viele Probleme.

Als Grundlage für die Berechnung dient die Größe des Problems, die hier als Anzahl der Variablen der Zielfunktion definiert ist. Damit ist die wirkliche Anzahl der Problemvariablen gemeint (und nicht etwa die Anzahl der Variablen, die bei der Arbeit mit einer binären Repräsentation des Evolutionären Algorithmus für ein Problem mit reellen oder ganzzahligen Variablen benutzt werden).

Anzahl der Unterpopulationen:

$$AnzUnderPop = \text{floor}(\sqrt{AnzVar}) \quad \begin{aligned} AnzVar: & \text{Anzahl der Variablen} \\ \text{floor:} & \text{runden gegen Null} \end{aligned} \quad (4-5)$$

Anzahl der Individuen pro Unterpopulation:

$$AnzIndUnderPop = 20 + 5 \cdot \text{floor}\left(\frac{AnzVar}{50}\right) \quad (4-6)$$

Die von diesen Überschlagsformeln gelieferten Werte, s. Tabelle 4-1, sind Richtwerte, die eine obere Schranke darstellen. Mit diesen Werten wird eine relativ robuste Arbeit des Evolutionären Algorithmus gewährleistet. Sobald das Problem etwas genauer untersucht, problemspezifisches Wissen eingearbeitet und verschiedene Läufe durchgeführt wurden, können die Anzahl der Unterpopulationen sowie die Individuenanzahl meist etwas niedriger gewählt werden als die von Gl. 4-5 und 4-6 gelieferten Werte.

Wenn das regionale Modell auf einer parallelen Rechnerarchitektur implementiert wird, ist die Anzahl der Unterpopulationen durch die Anzahl der zur Verfügung stehenden Prozessoren vorgegeben bzw. die Anzahl der Unterpopulationen muß ein Vielfaches der Prozessorzahl sein. In Anlehnung an die oben gegebenen Richtwerte kann dann die Anzahl der Individuen pro Unterpopulation gewählt werden.

**Tabelle 4-1.** Anzahl der Unterpopulationen und Individuen je Unterpopulation sowie Gesamtzahl der Individuen in Abhängigkeit der Anzahl der Variablen des Problems

Variablen	Unterpopulationen	Individuen je Unterpopulation	Gesamtzahl Individuen
10	3	20	60
20	4	20	80
40	6	20	120
50	7	25	175
80	8	25	200
100	10	30	300
150	12	35	420
200	14	40	560

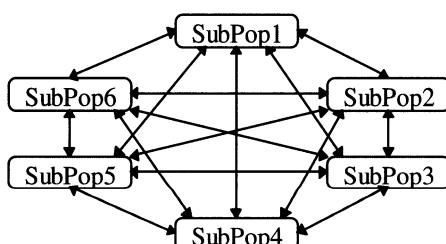
#### 4.4.2 Migrationstopologien

Mit der Migrationstopologie werden die Verbindungen zwischen den Unterpopulationen definiert. Nur auf diesen Verbindungen können Individuen von einer Unterpopulation zu einer anderen wandern. Für die Topologie gibt es viele verschiedene Möglichkeiten.

Migration kann stattfinden:

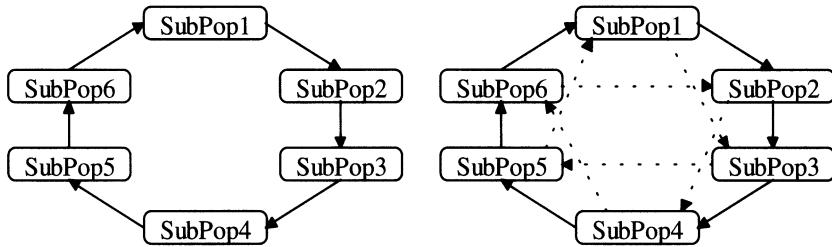
- zwischen allen Unterpopulationen (uneingeschränkte Migration, vollständige Netztopologie), Abb. 4-6,
- in einer Ringtopologie, Abb. 4-7,
- in einer Nachbarschaftstopologie, Abb. 4-8,
- zufällige Auswahl der Unterpopulationen, die Individuen austauschen ([Bel95]).

Die allgemeinste Topologie ist die uneingeschränkte Migration (u.a. [PLG87]), s. Abb. 4-6. In diesem Fall ist jede Unterpopulation mit jeder anderen Unterpopulation verbunden. Daher wird dies auch als vollständige Netztopologie bezeichnet. Die Individuen, die in eine Unterpopulation wandern, können von jeder anderen Unterpopulation kommen. Der Ablauf einer solchen Migration ist in Abb. 4-10 dargestellt.



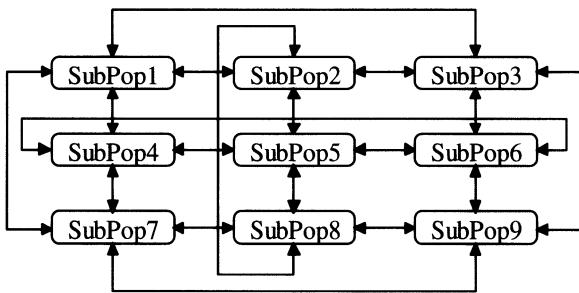
**Abb. 4-6.** Struktur für die uneingeschränkte Migration (vollständige Netztopologie)

Das einfachste Schema ist die Migration in einer Ringtopologie (u.a. [GW91], [VBS91]), s. Abb. 4-7. Die Migranten können hier nur von den in einer Richtung benachbarten Unterpopulationen einwandern. In dem in Abb. 4-7 links gegebenen Beispiel einer eindimensionalen Ringtopologie bedeutet dies, daß Individuen von Unterpopulation 6 zu Unterpopulation 1, Individuen von Unterpopulation 1 zu Unterpopulation 2 usw. wandern.



**Abb. 4-7.** Migration in einer eindimensionalen Ringtopologie; links: Abstand 1, rechts: Abstand 1 und 2

Eine Erweiterung der Ringtopologie ist auf zwei Arten möglich. Einmal kann der Abstand zu möglichen Nachbarn vergrößert werden. Ein Beispiel dafür ist in Abb. 4-7 rechts gezeigt, hier bestehen auch Verbindungen zu Unterpopulationen, die den Abstand 2 haben. Bei einer Vergrößerung des Abstandes bis auf  $\text{AnzUnterPop} - 1$  geht die Ringtopologie in die vollständige Netztopologie über. Weiterhin kann die Ringtopologie in höhere Dimensionen übertragen werden. Bei der folgenden Erläuterung der Nachbarschaftstopologie wird darauf näher eingegangen.

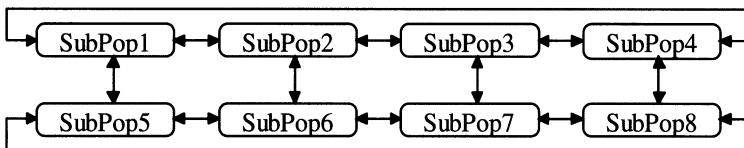


**Abb. 4-8.** Migration in einer Nachbarschaftstopologie (2-D Gitter)

Die Nachbarschaftstopologie ist der Ringtopologie ähnlich. Hierbei erfolgt eine Migration von Individuen von allen Nachbarn einer Unterpopulation aus. Der Abstand kann dabei auch höher als 1 gewählt werden. Eine eindimensionale

Nachbarschaftstopologie geht bei einem Abstand von  $(\text{AnzUnterPop} - 1)/2$  in die vollständige Netztopologie über, für höherdimensionale Nachbarschaftstopologien ist dieser Wert noch kleiner. Abbildung 4-8 zeigt eine zweidimensionale Nachbarschaftstopologie mit Abstand 1. Diese Topologie wird auch als zweidimensionales Gitter bzw. als Torus bezeichnet. Für höhere Dimensionen ergibt sich ein Hyperwürfel (u.a. [Tan87], [Tan89], [CMR91], [VBS91]).

Eine Variante der zweidimensionalen Nachbarschaftstopologie stellt die Leitertopologie dar ([MSB91]), Abb. 4-9. Bei gleicher Anzahl von Unterpopulationen besitzt diese Topologie einen größeren Durchmesser als eine zweidimensionale Gittertopologie, wodurch Informationen zwischen den Unterpopulationen deutlich langsamer ausgetauscht werden.



**Abb. 4-9.** Migration in einer zweidimensionalen Leitertopologie

Bei einer Implementierung des regionalen Modells auf einer parallelen Rechnerarchitektur ist die Topologie der Unterpopulationen meist durch die Topologie des entsprechenden Rechnersystems gegeben. Oftmals ist dies ein zweidimensionales Gitter (2-D Implementierung der Nachbarschaftstopologie) oder ein Hyperwürfel. Auf transputerbasierten Systemen ist dagegen meist nur die maximale Anzahl von Nachbarn vorgegeben, wodurch sehr unterschiedliche Topologien realisiert werden können.

Ein Vergleich der verschiedenen Topologien für die Migration ist nicht ganz einfach. Die entscheidende Frage ist ein Vergleichskriterium, nach dem die Topologien untereinander verglichen werden können. Die Anzahl der Nachbarn einer Unterpopulation im Verhältnis zur Gesamtzahl aller Unterpopulationen, der Durchmesser der Topologie, der Abstand von einer Unterpopulation zur jeweils am weitesten entfernten Unterpopulation und die mögliche Geschwindigkeit des Informationsflusses könnten zusammen ein Kriterium ergeben.

In Unterabschn. 4.3.2 wird eine allgemeinere Beschreibung von Nachbarschaften gegeben. Diese beinhaltet alle in diesem Unterabschnitt aufgeführten Migrationstopologien als Spezialfälle. Die allgemeine Beschreibung der Nachbarschaften ermöglicht die einfache Erweiterung der hier genannten Topologien auf höhere Dimensionen. Außerdem wird dadurch der Vergleich der einzelnen Topologien erleichtert.

#### 4.4.3 Migrationsintervall und Migrationsrate

Migrationsintervall und Migrationsrate steuern den Austausch von Individuen zwischen den Unterpopulationen. Das Migrationsintervall gibt an, wie oft bzw.

wann Migration stattfindet. Die Migrationsrate gibt den Anteil oder die Anzahl der Individuen einer Unterpopulation an, die ausgetauscht werden. Beide Parameter zusammengenommen geben die Menge an Information an, die durch Migration zwischen den Unterpopulationen ausgetauscht werden soll.

Das Migrationsintervall kann als eine fest definierte Zahl von Generationen gegeben sein und definiert damit die Isolationszeit (die Zeit, in der sich die Unterpopulationen isoliert voneinander entwickeln). Der Wert kann zwischen 1 und einer größeren Zahl von Generationen gewählt werden. Bei kleineren Unterpopulationen sollte das Migrationsintervall kleiner sein als bei Verwendung größerer Unterpopulationen (Abhängigkeit des Migrationszeitpunktes von der Anzahl der Individuen in den Unterpopulationen, *AnzIndUnterPop*). Der Zeitpunkt für eine Migration kann aber auch in Abhängigkeit von anderen Ereignissen als der Zahl vergangener Generationen definiert werden. Eine Variante ist die vorzeitig auftretende Konvergenz bei Unterpopulationen. Eine weitere Möglichkeit besteht darin, immer dann eine Migration durchzuführen, wenn in einer Unterpopulation ein besseres Individuum gefunden wurde.<sup>2</sup>

Die Migrationsrate wird als Anteil der Unterpopulation oder als feste Zahl von Individuen pro Unterpopulation angegeben, wobei die erste Angabe über verschiedene Unterpopulationsgrößen vergleichbar ist und deshalb hier verwendet wird. Die Migrationsrate sollte immer nur einen Teil der Unterpopulation umfassen. In verschiedenen Arbeiten mit Evolutionären Algorithmen haben sich folgende Richtwerte als gute Startwerte erwiesen.

Migrationsintervall:

$$\text{Migrationsintervall} = \text{AnzIndUnterPop} \quad [\text{Generationen}] \\ (10 - 40 \text{ Generationen}) \quad (4-7)$$

Migrationsrate:

$$\text{Migrationsrate} = 20\% \cdot \text{AnzIndUnterPop} \quad [\text{Individuen}] \\ (5\% - 30\% \text{ der Individuen einer Unterpopulation}) \quad (4-8)$$

Bei einem sehr kleinen Wert für das Migrationsintervall (< 10 Generationen), sollte die Migrationsrate auch sehr klein sein, da sonst zuviel Information zwischen den Unterpopulationen ausgetauscht wird. Bei einem Migrationsintervall von einer Generation (häufigste Migration) wird meist eine Migrationsrate von einem Individuum (minimale Migrationsrate) gewählt ([PLG87]).

Wenn die Migrationsrate groß und/oder das Migrationsintervall klein gewählt werden, verhält sich das regionale Modell ähnlich dem globalen Modell, da kaum noch eine Isolation zwischen den Unterpopulationen besteht.

---

<sup>2</sup> Es ist immer noch eine offene Frage, zu welchem Zeitpunkt eine Migration stattfinden sollte. Das entscheidende Hindernis ist die Definition eines Kriteriums, das in Abhängigkeit des Zustandes der Unterpopulation/en eine Entscheidung liefert. Die angesprochenen Kriterien sind die am häufigsten verwendeten.

#### 4.4.4 Auswahl der Migranten

Während der Migration findet an zwei Stellen eine Selektion von Individuen statt. Zuerst werden die Individuen ausgewählt, die aus einer Unterpopulation auswandern (Migrationsauswahl). Danach müssen die Individuen ausgewählt werden, die in der Unterpopulation durch die eingewanderten Individuen (Migranten) ersetzt werden sollen (Migrationsauslese).

Für die Auswahl der Individuen gibt es verschiedene Kombinationen:

1. beste – schlechteste: die besten Individuen (fitneßproportionale Auswahl) ersetzen die schlechtesten Individuen – entsprechend ihrer Fitneß werden die besten Individuen einer Unterpopulation für die Migration ausgewählt und ersetzen die schlechtesten Individuen der Unterpopulation, in die sie wandern (u.a. [SWM91], [VBS91]),
2. beste – zufällig: die besten Individuen ersetzen Individuen gleichmäßig verteilt zufällig ([GEATbx]),
3. zufällig – zufällig: gleichmäßig verteilt zufällig ausgewählte Individuen ersetzen Individuen gleichmäßig verteilt zufällig ([CMR91], [GEATbx]) und
4. beste/zufällig – Migranten: Migranten sind die besten Individuen oder zufällig ausgewählte Individuen und es werden die Individuen einer Unterpopulation ersetzt, die selbst als Migranten in eine andere Unterpopulation wandern ([Tan89], [Bel95]).

Wenn eine fitneßproportionale Auswahl verwendet wird, so übt die Migration einen zusätzlichen Selektionsdruck in Richtung der besseren Individuen aus. Bei der zufällig – zufällig erfolgenden Auswahl dagegen findet nur ein Durchmischen der Individuen zwischen den Unterpopulationen statt, es wird kein zusätzlicher Selektionsdruck ausgeübt.

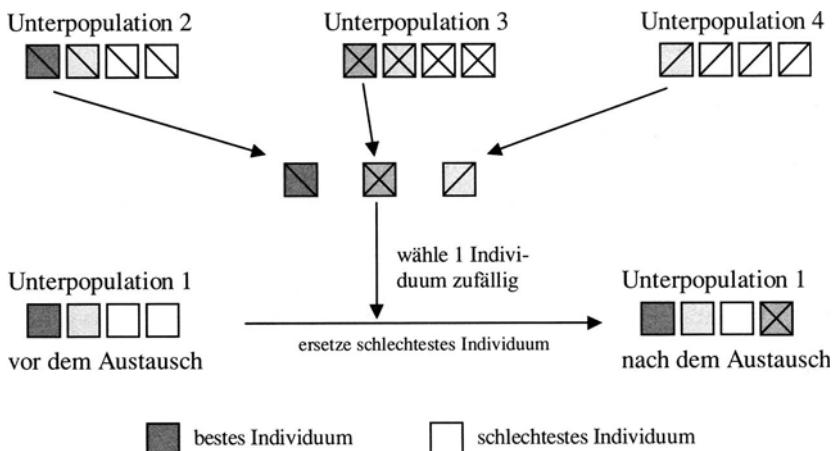
#### 4.4.5 Ablauf der Migration

Abbildung 4-10 zeigt den Ablauf der Migration von Individuen zwischen Unterpopulationen. In diesem Beispiel werden vier Unterpopulationen verwendet, zwischen denen eine uneingeschränkte Migration (vollständige Netztopologie) stattfindet. Die Auswahl der Migranten erfolgt fitneßbasiert mit einer Migrationsrate von einem Individuum pro Unterpopulation. Das Beispiel zeigt, wie Individuen in die Unterpopulation 1 wandern.

Zu Beginn werden die Unterpopulationen ausgewählt, die mit Unterpopulation 1 durch die verwendete Migrationstopologie verbunden sind. Durch die verwendete vollständige Netztopologie sind das in diesem Falle alle anderen Unterpopulationen 2, 3 und 4. Aus diesen Unterpopulationen werden jeweils entsprechend der Selektionsmethode (fitneßbasiert – Auswahl der besten Individuen) und der Migrationsrate (ein Individuum) Individuen als potentielle Migranten ausgewählt, im Beispiel das jeweils beste Individuum dieser Unterpopulationen. Diese Individuen bilden einen Pool, aus dem gleichmäßig zufällig das Individuum ausgewählt wird, das in Unterpopulation 1 eingefügt wird und dabei das schlechteste Individuum in Unterpopulation 1 ersetzt. Dieser Ablauf wird für jede

Unterpopulation durchgeführt. Durch dieses Verfahren ist gleichzeitig sichergestellt, daß keine Unterpopulation Individuen von sich selbst erhält.

Bei einer anderen Topologie für die Migration ändern sich nur die Unterpopulationen, aus denen die Individuen für die Migration in eine bestimmte Unterpopulation ausgewählt werden. Der Ablauf ist ansonsten identisch.



**Abb. 4-10.** Ablauf der Migration von Individuen zwischen Unterpopulationen

#### 4.4.6 Analyse des regionalen Modells

Die parallele Implementierung des regionalen Modells bringt nicht nur eine Beschleunigung der Berechnungszeiten. Es zeigt sich, daß bei einer Vielzahl untersuchter Zielfunktionen weniger Zielfunktionsberechnungen zum Erreichen eines gleichwertigen Ergebnisses benötigt werden, als bei Verwendung einer Gesamtpopulation mit entsprechend vielen Individuen. Gleichlautende Ergebnisse werden in vielen Untersuchungen berichtet, u.a. [Bel95], [GW91], [Loh91], [MSB91], [Rud91], [SWM91], [Swm96], [Tan89], [VBS91], [VSB92]. Damit ergibt sich, daß die Verwendung des regionalen Modells auch bei einer sequentiellen Implementierung (Computer mit einem Prozessor) Vorteile bringt. Der Evolutionäre Algorithmus findet öfter eine Lösung (robuster) oder die Lösung wird mit weniger Zielfunktionsberechnungen ermittelt (schneller).

MÜHLENBEIN stellt fest:

Unterpopulationen, die für eine gewisse Zeit isoliert voneinander sind, halten die Verschiedenartigkeit der Population hoch. Nach der Migration von Individuen werden neue vielversprechende Bereiche durch Rekombination entdeckt. ([Müh91], S.323)

Wenn die von Individuen in unterschiedlichen Unterpopulationen gefundenen Teillösungen in ihrer Kombination (durch Rekombination) eine bessere Lösung ergeben, so sollte die Anwendung des regionalen Modells deutliche Vorteile ge-

genüber dem globalen Modell (oder einem einfachen Evolutionären Algorithmus) bringen. Erzeugt die Kombination der Teillösungen dagegen eine schlechtere Lösung, so müßte eine einheitliche Gesamtpopulation Vorteile haben [SWM91].

Aus diesen beiden Anmerkungen sind bereits die Einsatzgebiete des regionalen Modells vorgezeichnet. Die Unterpopulationen arbeiten meist (besonders zu Beginn eines Laufs) in unterschiedlichen Bereichen des Suchraumes. Dabei werden unterschiedliche Teillösungen gefunden. Wenn sich aus den Teillösungen der verschiedenen Unterpopulationen bessere Lösungen zusammensetzen lassen, so ist die Verwendung des regionalen Modells von Vorteil. Viele der oft genutzten Testfunktionen zur Demonstration der Vorteile des Einsatzes des regionalen Modells sind meist so konstruiert, daß sich die Gesamtlösung durch Kombination von Einzellösungen (separierbare Funktionen) ermitteln läßt (z.B. RASTRIGIN's Funktion, SCHWEFEL's Funktion, s. [GEATbx]).

Finden dagegen alle Unterpopulationen dieselbe Lösung bzw. benutzen denselben Weg durch den Suchraum, so arbeitet eine Gesamtpopulation effektiver. Wenn die Teillösungen der Unterpopulationen in ihrer Kombination schlechtere Lösungen ergeben, so ist die Migration von Individuen im regionalen Modell von Nachteil. In diesem Falle verbliebe nur der Vorteil der Suche in unterschiedlichen Bereichen des Suchraumes und das langsamere Absinken der Diversität der Individuen in der Gesamtpopulation.

## 4.5 Anwendung verschiedener Strategien

Bei einer in mehrere Unterpopulationen geteilten Population (regionales Modell) sind die einzelnen Unterpopulationen unter sich in ihrer Parametrisierung identisch und gehen daher bei der Suche nach derselben Strategie vor. Von der Arbeit mit unterschiedlichen Evolutionären Algorithmen ist bekannt, daß die Wahl der Operatoren und deren Parameter einen großen Einfluß auf den Verlauf und den Ausgang der Optimierung hat.

Eine logische Erweiterung des regionalen Modells eines Evolutionären Algorithmus ist die Verwendung unterschiedlicher Strategien für einige oder alle Unterpopulationen (engl. *multiple strategies*). Jede Unterpopulation verwendet einen eigenen Satz von Parametern, der sich von denen der anderen Unterpopulationen in einem, in mehreren oder in allen Parametern unterscheiden kann. Je nach Anwendung der zur Verfügung stehenden Operatoren bzw. deren Parametrisierung kann sich das Suchverhalten der einzelnen Unterpopulationen stark oder wenig voneinander unterscheiden.

Durch die Verwendung des regionalen Modells bleiben dessen Eigenschaften erhalten. Die Unterpopulationen suchen weiterhin parallel zueinander. Von Zeit zu Zeit findet ein Austausch von Individuen zwischen den Unterpopulationen (Migration) statt. Dies führt zu einer Propagierung neuer und guter Individuen und damit zu einer Fokussierung der Suche auf vielversprechende Regionen des Suchbereiches.

Für die Anwendung verschiedener Strategien pro Unterpopulation gibt es zwei Ansatzpunkte:

1. Abhängig vom Ort im Suchraum bzw. vom Fortschritt der Optimierung sind unterschiedliche Strategien am günstigsten.
2. Die für ein spezielles Problem beste/schnellste/robuste Suchstrategie ist nicht bekannt und kann auf diese Weise ermittelt werden.

Oftmals sind aus vorangegangenen Experimenten mit ähnlichen Optimierungsproblemen Erkenntnisse vorhanden, mit welchen Operatoren bzw. Parametern ein Problem gut zu lösen ist. Da diese Entscheidung nur in den seltensten Fällen eindeutig ist, bietet sich eine Anwendung verschiedener Strategien an. In einem ersten Lösungsansatz werden die aus Sicht des Anwenders vielversprechenden Operatoren und Parameter angewandt. Mittels einer einfachen Auswertung der Konvergenz der einzelnen Unterpopulationen können recht schnell die Operatoren und Parametereinstellungen verworfen werden, die keinen großen Beitrag zum Fortschritt der Optimierung geben.

#### **4.5.1 Berechnung der Ordnung der Unterpopulationen**

Für die Auswertung des Einsatzes unterschiedlicher Strategien ist es notwendig, ein Maß zu finden, das es ermöglicht, den Erfolg der einzelnen Strategien zu bewerten. Als Maß für den Erfolg einer Unterpopulation wird die Position einer Unterpopulation in einer Ordnung der Unterpopulationen verwendet. Dabei ist eine niedrige Position besser als eine hohe Position. Daraus kann abgelesen werden, wie gut eine Unterpopulation zu einem bestimmten Zeitpunkt im Vergleich zu den anderen Unterpopulationen ist. Damit ist eine einfache Bewertung der Anwendung verschiedener Strategien möglich.

Zur Bildung der Ordnung der Unterpopulationen müssen die Unterpopulationen untereinander nach einem einheitlichen Kriterium sortiert werden. Da eine Unterpopulation aus einer Anzahl von Individuen besteht, ergibt sich die Güte einer Unterpopulation aus den Eigenschaften ihrer Individuen.

Das Verfahren zur Bildung der Ordnung der Unterpopulationen findet nach folgendem Schema statt:

1. Von jeder Unterpopulation wird eine Anzahl der besten Individuen verwendet. (Der Anteil einer Unterpopulation, der zur Bewertung dieser Unterpopulation verwendet wird, kann nur das beste Individuum, eine Anzahl von Individuen der Unterpopulation oder alle Individuen der Unterpopulation umfassen.)
2. Zwischen diesen Individuen findet ein Ranking (Reihenfolgebildung) zur Bewertung statt. Grundlage für die Bewertung der einzelnen Individuen ist deren Güte (deren Zielfunktionswert). Die Bewertung der Individuen entspricht den Verfahren zur Fitneßzuweisung aus Abschn. 3.1, ab S.16. Jedes der dort erläuterten Verfahren kann verwendet werden. Alle angeprochenen Probleme bzw. Vorteile der einzelnen Verfahren treffen auch hier zu. Empfohlen wird die Verwendung der linearen reihenfolgebasierten Fitneßzuweisung (lineares Ranking). Dadurch wird jedem Individuum

- eine Bewertung (Fitneßwert) im Vergleich zu allen anderen Individuen gegeben.
3. Aus der Bewertung der Individuen einer Unterpopulation wird eine Bewertung jeder Unterpopulation gebildet.
  4. Durch Sortieren der Bewertung der Unterpopulationen ergibt sich die Ordnung der Unterpopulationen.

Dieser Vorgang wird im folgenden an drei speziellen Varianten genauer erläutert.

Bewertung durch alle Individuen einer Unterpopulation:

1. Von jeder Unterpopulation werden alle Individuen verwendet.
2. Alle Individuen werden durch ein lineares Ranking bewertet.
3. Aus den Fitneßwerten der Individuen einer jeden Unterpopulation wird der Mittelwert berechnet und bildet die Bewertung der Unterpopulation.
4. Durch Sortieren dieser Bewertung wird die Ordnung der Unterpopulationen gebildet.

Bewertung durch das beste Individuum jeder Unterpopulation:

1. Von jeder Unterpopulation wird nur das beste Individuum verwendet.
2. Diese Individuen werden sortiert und die Ordnung der Individuen ergibt die Ordnung der Unterpopulationen.

Bewertung durch das beste Individuum der gesamten Population:

1. Es wird nur das beste Individuum der gesamten Population in Betracht gezogen.
2. Die Unterpopulation, die dieses Individuum enthält, ist die beste Unterpopulation. Alle anderen Unterpopulationen sind schlechter, es wird zwischen ihnen keine Unterscheidung vorgenommen.

Die zweite und dritte Variante sind Spezialfälle der ersten Variante, die sich durch Beschränkung der Anzahl der Individuen, die zur Bewertung verwendet werden, ergeben.

Die Ordnung der Unterpopulationen gibt ein momentanes Abbild der Unterpopulationen. In Abhängigkeit der verwendeten Operatoren und Parameter kann die Ordnung innerhalb einiger Generationen stark schwanken. Dies ist im besonderen zu beobachten, wenn die Strategien der Unterpopulationen ähnlich erfolgreich sind. Für die Darstellung und Bewertung ist es daher von Vorteil, wenn die Ordnung der Unterpopulationen einer Gewichtung unterzogen wird. Dadurch wird aus dem *Rang* jeder Unterpopulation ein *Positionswert* gebildet, der einen gewichteten Mittelwert des Rangs der Unterpopulation in den zurückliegenden Generationen darstellt.

$$\text{Positionswert}_{\text{Generation}} = 0,9 \cdot \text{Positionswert}_{\text{Generation}-1} + 0,1 \cdot \text{Rang}_{\text{Generation}} \quad (4-9)$$

Dieser *Positionswert* wird für die folgenden Darstellungen verwendet. Je kleiner der *Positionswert* ist, um so größer ist der Erfolg einer Unterpopulation. Wenn sich der Rang einer Unterpopulation lange nicht verändert, wird der *Positionswert* gleich dem *Rang*.

### 4.5.2 Beispiel der Anwendung verschiedener Strategien

Die Bewertung des Erfolgs einer Strategie hängt entscheidend von der verwendeten Methode zur Bestimmung der Ordnung der Unterpopulationen ab. Die Verwendung des besten Individuums der gesamten Population ist die einfachste Methode. Dafür liefert sie nur eine Aussage darüber, welche Unterpopulation die beste ist; es findet keinerlei Differenzierung zwischen den anderen Unterpopulationen statt. Eine Ordnung der Unterpopulationen kann erst durch Verwendung des besten Individuums jeder Unterpopulation ermittelt werden. Damit ist eine differenziertere Bewertung des Erfolgs der einzelnen Unterpopulationen möglich. Allerdings wird durch die ausschließliche Verwendung des jeweils besten Individuums noch immer eine beschränkte Bewertung des Erfolgs einer Unterpopulation vorgenommen.

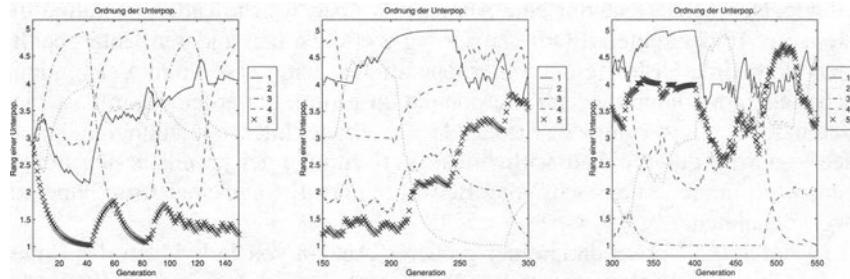
Erst durch die Verwendung einer größeren Anzahl von Individuen der Unterpopulationen kann eine umfassendere Bewertung des Erfolgs einer Unterpopulation vorgenommen werden. Durch die größere Anzahl von Individuen wird die Verteilung der Gütwerte der Individuen mit in die Bewertung einbezogen. Damit kann eine Unterpopulation, die nicht das beste Individuum enthält, dafür aber fast nur gute Individuen, besser bewertet werden, als eine Unterpopulation mit wenigen sehr guten und vielen schlechten Individuen.

Die Verwendung der Verfahren soll an einem Beispiel illustriert werden, bei dem zwei Unterpopulationen miteinander verglichen werden sollen. Die erste Unterpopulation hat ein sehr gutes Individuum, die anderen Individuen sind schlecht. Die zweite Unterpopulation hat viele gute Individuen und nur wenige schlechte Individuen. Welche dieser beiden Unterpopulationen ist besser? Wenn die erste Unterpopulation mit dem sehr guten Individuum als beste Unterpopulation bewertet werden soll, dann kann eine der einfachen Methoden verwendet werden, die nur das beste Individuum/die besten Individuen zur Bewertung benutzen. Soll dagegen ein umfassenderes Abbild in der Ordnung der Unterpopulationen zum Ausdruck kommen, so müssen auch mehrere Individuen in die Bewertung einbezogen werden. Der Mittelwert der Fitneß der Individuen einer Unterpopulation ist eine solche Möglichkeit.

Abbildung 4-11 zeigt ein Beispiel des Verlaufs und der Ergebnisse bei der Anwendung verschiedener Strategien. Die Optimierung (hier: DEJONG's Funktion 1 oder Hypersphäre) wurde mit 5 Unterpopulationen durchgeführt. Jede der Unterpopulationen verwendete eine andere Strategie, die sich in der Größe der verwendeten Mutationsschritte unterschieden (1: große Mutationsschritte; 2: mittlere Mutationsschritte; 3: kleine Mutationsschritte; 4: winzige Mutationsschritte; 5: mittlere und kleine Mutationsschritte zusammen). Je kleiner die Mutationsschritte sind, um so lokaler ist die Suche dieser Strategie. Alle Strategien verwendeten denselben Rekombinationsoperator, alle weiteren Parameter waren für alle Strategien identisch (z.B. *generation gap*: 0,9). Alle 40 Generationen fand eine Migration zwischen den Unterpopulationen statt. Dabei wanderten die besten Individuen in einer vollständigen Netzstruktur.

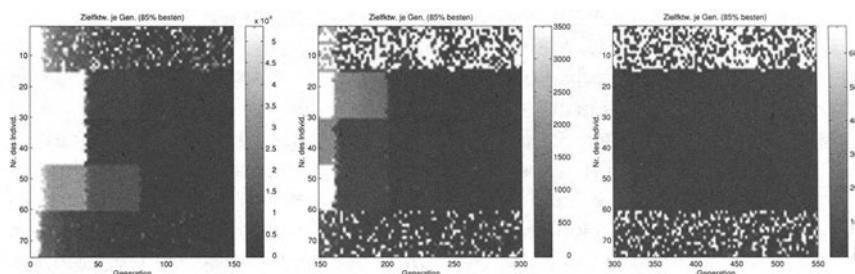
Die linke Grafik, die den Beginn des Laufs zeigt, lässt erkennen, daß am Anfang Strategie 5 am erfolgreichsten ist. Dies ändert sich erst ab Generation 200, wenn Strategie 2 besser wird und ab Generation 220 die Spitzenposition übernimmt, s. mittlere Grafik. Ab Generation 300 wird Strategie 3 am besten und ab

Generation 500 Strategie 4, s. rechte Grafik. Strategie 1 zeigt während der ersten 50 Generationen ein gutes Verhalten, s. linke Grafik, fällt dann aber zurück und kann nie wieder erfolgreich sein.



**Abb. 4-11.** Anwendung verschiedener Strategien, Ordnung der Unterpopulationen; links: Beginn des Laufs, Mitte: Mitte des Laufs, rechts: Ende des Laufs

Bei Kenntnis der Beispielfunktion ist dieses Verhalten nicht verwunderlich. Große Mutationsschritte sind besonders zu Beginn erfolgreich, danach führen große Mutationsschritte nur noch zu Verschlechterungen. Entsprechendes gilt für die Strategien mit immer kleiner werdenden Mutationsschritten. Wenn größere Schrittweiten noch Fortschritte bringen, sind kleine Schritte zu langsam. Damit hat in diesem Beispiel jede der ersten vier Strategien ihren speziellen Einsatzbereich bzw. besten Zeitpunkt. Eine Zwischenstellung nimmt Strategie 5 mit den mittleren und kleinen Schritten ein. Es ist eindeutig aus der linken Grafik zu entnehmen, daß diese Strategie am Anfang die Beste ist; ab und zu größere Schritte und meist kleinere Schritte sind eine gute Strategie. Ab Generation 200 ist dann aber eine der spezialisierten Strategien besser, da Strategie 5 nur noch selten gute Individuen und meist schlechte Individuen produziert.



**Abb. 4-12.** Anwendung verschiedener Strategien, Zielfunktionswerte aller Individuen; links: Beginn des Laufs, Mitte: Mitte des Laufs, rechts: Ende des Laufs (gleicher Lauf wie in Abb. 4-11)

Ein Blick auf Abb. 4-12 unterstreicht diese Analyse. Strategie 5 (Unterpopulation 5 mit den Individuen 61-75) produziert von Beginn an in jeder Generation deutlich besser werdende Individuen. Bis Generation 50 kann dies auch Strategie 1 (Individuen 1-15). Danach produziert Strategie 1 nur noch schlechte Individuen, die guten Individuen kamen durch Migration in Unterpopulation 1. Bis Generation 200 hat Strategie 5 immer den größten Fortschritt zu verzeichnen. Erst dann wird Strategie 2 (Individuen 16-30) besser, s. mittlere Grafik.

In allen unteren Grafiken ist deutlich der Effekt der Migration alle 40 Generationen zu erkennen. Neue gute Individuen von anderen Unterpopulationen werden in die Unterpopulationen eingefügt. Die Information dieser guten Individuen kann sich sehr schnell in den Unterpopulationen ausbreiten.

#### 4.5.3 Analyse der Anwendung verschiedener Strategien

Das Beispiel im vorigen Unterabschnitt verdeutlicht die Möglichkeiten, die sich aus der Anwendung verschiedener Strategien und der anschließenden grafischen Auswertung ergeben.

Zum einen ist die Anwendung verschiedener Strategien einfacher und meist schneller als die Durchführung unabhängiger Experimente mit jeweils eigenen Parametereinstellungen. Die Ergebnisse werden in einer kompakteren Form dargeboten und sind leicht zu überschauen.

Zum anderen ist die gleichzeitige Anwendung verschiedener Strategien in einem Lauf von großem Vorteil, da die für sich allein nicht sehr erfolgreichen Strategien in ihrer verbundenen Anwendung bessere Ergebnisse bringen als jede Strategie für sich. Es kommt z.B. recht häufig vor, daß zu Beginn eines Laufs eine andere Suchstrategie erfolgreich ist als in der Mitte oder am Ende eines Laufs (grobe Suche zu Beginn gegenüber feiner Suche am Ende, s. Beispiel in Abb. 4-11 und 4-12). Die Erzielung dieses Effekts wurde bisher durch die Adaption von Parametern der Suchstrategie versucht, was aber meist nur bei einfachen bzw. bekannten Optimierungsproblemen mit vertretbarem Aufwand zu erreichen ist.

Die Anwendung verschiedener Strategien ermöglicht damit eine höhere Stufe des Einsatzes Evolutionärer Algorithmen. Aus den angesprochenen Vorteilen sind dies im besonderen zwei Bereiche:

- Test einer Anzahl von Parametereinstellungen zur Ermittlung guter Strategien und zum gleichzeitigen Ausschließen erfolgloser Strategien,
- gleichzeitige Anwendung verschiedener Strategien, wobei jede der Strategien zu einem bestimmten Zeitpunkt eines Laufs besonders erfolgreich ist bzw. die Strategien sich untereinander ergänzen.

Ein Vorbild für die Anwendung verschiedener Strategien in der natürlichen Evolution kann leicht gefunden werden. So ist es viel wahrscheinlicher, daß sich voneinander getrennt entwickelnde Unterpopulationen auch verschiedene Strategien verwenden. Innerhalb einer Unterpopulation ist es dagegen wahrscheinlich, daß alle Individuen eine (fast) identische Strategie benutzen.

Die Anwendung verschiedener Suchstrategien für jede Unterpopulation wird in einigen der frühen Berichte zu parallelen Evolutionären Algorithmen erwähnt. TANESE berichtet in [Tan87] von der Verwendung unterschiedlicher Mutations- und Rekombinationsraten für einige oder alle Unterpopulationen. Die Ergebnisse der Experimente weisen darauf hin, daß das verwendete regionale Modell mit unterschiedlichen Parametern für die einzelnen Unterpopulationen robuster war, als es bei der Verwendung identischer Parameter der Fall war und die besten Werte für diese Parameter nicht bekannt sind.

## 4.6 Konkurrierende Unterpopulationen

Eine logische Erweiterung des regionalen Modells mit unterschiedlichen Strategien für die einzelnen Unterpopulationen ist das Prinzip der miteinander konkurrierenden Unterpopulationen (engl. *competing subpopulations*). Bei der Anwendung unterschiedlicher Strategien innerhalb des regionalen Modells blieb die Größe jeder Unterpopulation (Anzahl der Individuen in einer Unterpopulation) während eines Laufs konstant. Auch wenn eine Strategie nicht erfolgreich ist, standen dieser Strategie weiterhin dieselben Ressourcen zur Verfügung.

Von dieser Konstanz der Ressourcen wird bei der Anwendung des Prinzips miteinander konkurrierender Unterpopulationen abgegangen. Statt dessen wird die Größe jeder Unterpopulation davon abhängig gemacht, ob die von ihr angewandte Strategie im Moment erfolgreich ist. Eine erfolgreiche Unterpopulation bekommt mehr Ressourcen zur Verfügung gestellt, eine erfolglose Unterpopulation muß Ressourcen an andere Unterpopulationen abgeben.

Eine Entsprechung dieses Prinzips in der natürlichen Evolution liegt auf der Hand. Eine Unterpopulation, die erfolgreich ist, hat mehr Ressourcen zur Verfügung und kann sich dadurch meist besser fortpflanzen und entwickeln. Dieses Prinzip ist eines der grundlegenden in der Natur und läßt sich in den verschiedensten Bereichen beobachten.

Immer, wenn ein Wettbewerb zwischen konkurrierenden Unterpopulationen durchgeführt wird, müssen die folgenden Schritte durchlaufen werden:

- Berechnung des Erfolgs der Unterpopulationen,
- Berechnung der Aufteilung der Ressourcen,
- Durchführung der Verteilung der Ressourcen.

Die Anwendung der miteinander konkurrierenden Unterpopulationen ermöglicht eine effektive Verteilung der Ressourcen. Die Unterpopulationen, die am erfolgreichsten sind, erhalten mehr Ressourcen (Individuen) und damit mehr Rechenressourcen. Durch eine geeignete Adaption werden diese Ressourcen effektiv umverteilt, sobald es eine Verschiebung innerhalb des Erfolges der Unterpopulationen gibt.

Durch eine geeignete Wahl der Parameter können verschiedene Eigenschaften für das Verhalten der miteinander konkurrierenden Unterpopulationen definiert werden und damit die Stärke der Konkurrenz bzw. Unterdrückung erfolgloser Unterpopulationen durch erfolgreiche Unterpopulationen beeinflußt werden.

### 4.6.1 Erfolg der Unterpopulationen

Die Berechnung des Erfolgs der Unterpopulationen wurde bereits im Abschnitt zur Anwendung verschiedener Strategien, Abschn. 4.5, behandelt. Die dortigen Ergebnisse werden hier verwendet. Der Erfolg der Unterpopulationen wird in einer Ordnung der Unterpopulationen ausgedrückt, die wie in Unterabschn. 4.5.1 ermittelt wird. Um ein zu starkes Schwanken der Ordnung der Unterpopulationen zu verhindern, wird für den Erfolg der Unterpopulationen dieselbe Gewichtung (Gl. 4-9) angewendet, wie für die Darstellung der Ordnung der Unterpopulationen.

### 4.6.2 Aufteilung der Ressourcen

Bei jedem Wettbewerb zwischen den Unterpopulationen werden die zur Verfügung stehenden Ressourcen neu aufgeteilt. Dies kann einmal durch die Bestimmung von erfolgreichen und erfolglosen Unterpopulationen geschehen, andererseits aber auch durch eine gewichtete Aufteilung der Ressourcen.

Für die Bestimmung erfolgreicher und erfolgloser Unterpopulationen dient die gewichtete Ordnung der Unterpopulationen als Grundlage. Dadurch wird festgelegt, welche Unterpopulationen zusätzliche Ressourcen bekommen und welche Unterpopulationen Ressourcen abgeben müssen.

Die einfachste Variante dafür ist, daß nur die beste Unterpopulation erfolgreich ist und Ressourcen erhält, alle anderen Unterpopulationen müssen Ressourcen abgeben. Dies würde während eines Laufs dazu führen, daß immer nur eine Unterpopulation viele Individuen enthält, alle anderen Unterpopulationen haben die minimale Anzahl an Individuen. Zu einer Koexistenz von zwei oder mehr Unterpopulationen kann es nicht kommen.

Etwas umfassender ist der Ansatz, daß ein Teil der Unterpopulationen als erfolgreich eingestuft wird, der Rest als erfolglos. Zum Beispiel können 25% der besten Unterpopulationen als erfolgreich eingestuft werden (*Erfolgsrate* = 0,25) und erhalten zusätzliche Ressourcen, die anderen 75% müssen Ressourcen abgeben. Dadurch können neben der besten Strategie auch solche Strategien mit einem größeren Teil der Ressourcen arbeiten, die zwar gut sind, aber nicht führend.

Bei einer genaueren Betrachtung dieses Prozesses der Aufteilung von Ressourcen finden sich Gemeinsamkeiten zur Fitneßzuweisung, s. Unterabschn. 3.1, ab S.16. Bei der Fitneßzuweisung geht es darum, jedem Individuum aus seinem Zielfunktionswert, im Vergleich zu den Zielfunktionswerten der anderen Individuen des Selektionspools, eine Fortpflanzungswahrscheinlichkeit und damit die Anzahl möglicher Nachkommen zuzuweisen. Die Güte eines Individuums wird umgerechnet in die Ressourcen, die dem Individuum zur Produktion von Nachkommen zur Verfügung stehen.

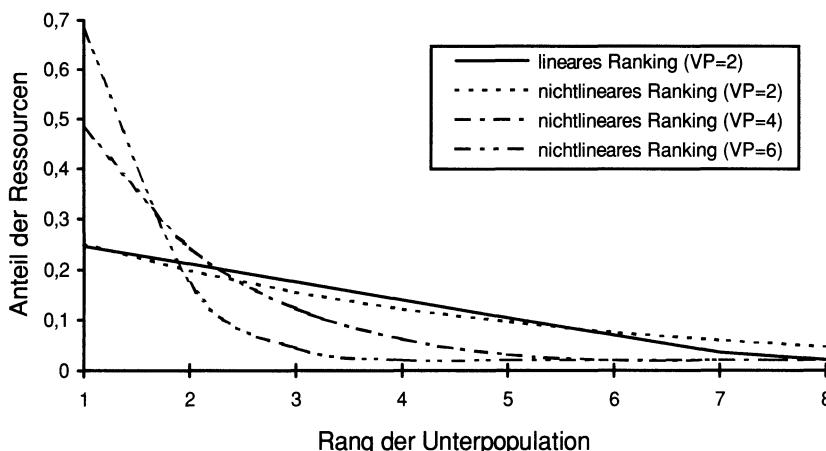
Genau um denselben Sachverhalt geht es hier, nur auf einer höheren Ebene. Die zur Verfügung stehenden Ressourcen sollen auf die Unterpopulationen aufgeteilt werden. Die Erkenntnisse aus der Fitneßzuweisung können direkt auf die Verteilung der Ressourcen bei konkurrierenden Unterpopulationen angewendet werden und führen zur gewichteten Aufteilung der Ressourcen. Bei der Fitneß-

zuweisung wurde gezeigt, daß Verfahren auf der Basis der Reihenfolge der Individuen (*ranking*) am robustesten arbeiten. Die in Unterabschn. 4.5.1 berechnete gewichtete Ordnung der Unterpopulationen, Gl. 4-9, dient deshalb als Grundlage für die Verteilung der Ressourcen.

Im nächsten Schritt wird jeder Unterpopulation auf der Grundlage ihres Ranges ein Anteil an den insgesamt zur Verfügung stehenden Ressourcen zugewiesen. Aus dem Bereich der Fitneßzuweisung sind die beiden Verfahren lineares Ranking und nichtlineares Ranking bekannt. Parameter dieser Verfahren ist der Selektionsdruck  $SP$ . In Analogie dazu wird hier der Parameter des Verteilungsdrucks  $VP$  definiert. Dieser bestimmt, wie die Ressourcen verteilt werden. Für einen niedrigen Verteilungsdruck ist der Unterschied im Anteil der Ressourcen jeder Unterpopulation nur gering. Für einen hohen Verteilungsdruck, besonders bei Anwendung des nichtlinearen Ranking, hat die beste Unterpopulation einen sehr hohen Anteil an den Ressourcen, für die anderen Unterpopulationen ist der Anteil deutlich geringer. Die weiteren guten Unterpopulationen erhalten aber trotzdem einen höheren Anteil an Ressourcen als die schlechten Unterpopulationen.

In Abb. 4-13 ist die Verteilung der Ressourcen auf acht Unterpopulationen für unterschiedliche Werte des Verteilungsdruckes dargestellt. Dabei wurde ein Minimum an Ressourcen (Unterpopulationsminimum, hier 1%), das jeder Unterpopulation zum Überleben zur Verfügung stehen muß, berücksichtigt.

Mit einem linearen Ranking ist nur ein maximaler Verteilungsdruck von 2 einstellbar. Da die beste Unterpopulation meist stärker bevorzugt werden soll, muß ein nichtlineares Ranking angewendet werden. Damit können Werte für den Verteilungsdruck bis zu *Anzahl der Unterpopulationen*-2 verwendet werden.



**Abb. 4-13.** Verteilung der Ressourcen auf die Unterpopulationen für lineares und nichtlineares Ranking und verschiedene Werte des Verteilungsdrucks

### 4.6.3 Umsetzung der Verteilung der Ressourcen

Für die Umsetzung der berechneten Aufteilung der Ressourcen werden die folgenden Verfahren und Parameter benutzt bzw. müssen definiert werden:

- Ressourcenverbrauch durch die Individuen,
- Häufigkeit, mit der ein Wettbewerb um Ressourcen durchgeführt wird (Konkurrenzintervall),
- maximaler Anteil an Ressourcen (Konkurrenzrate), der von erfolglosen Unterpopulationen abgegeben werden muß,
- Art der Auswahl der Ressourcen (Konkurrenzauswahl), die von erfolglosen Unterpopulationen abgegeben werden müssen,
- minimale Größe einer erfolglosen Unterpopulation (Unterpopulationsminimum).

#### **Ressourcenverbrauch**

Bisher wurde nur von den Ressourcen gesprochen. Es fehlte aber jede Umsetzung auf die vom Evolutionären Algorithmus benutzte Größe der Individuen. Diese wird durch den Ressourcenverbrauch definiert, der angibt, wieviele Ressourcen jedes Individuum benötigt.

Die einfachste Variante ist, daß jede Ressourceneinheit einem Individuum entspricht. Damit wird angenommen, daß die Individuen aller Strategien denselben Ressourcenverbrauch haben.

Es ist jedoch viel wahrscheinlicher, daß Individuen verschiedener Strategien einen unterschiedlichen Ressourcenverbrauch haben. Diese Annahme ermöglicht eine deutlich realistischere Modellierung der Konkurrenz zwischen Unterpopulationen bzw. verschiedenen Strategien. Eine Strategie kann z.B. mit 10 Ressourceneinheiten 10 Individuen versorgen, eine zweite Strategie nur 2 Individuen, eine dritte sogar 100 Individuen.

Der Parameter des Ressourcenverbrauchs bestimmt damit umgekehrt proportional, wie stark eine Unterpopulation wächst (neue Individuen bekommt), wenn sie zusätzliche Ressourcen zugewiesen bekommt. Gleichzeitig kann über den Ressourcenverbrauch einer Strategie gesteuert werden, ob eine Strategie mit einer kleinen Individuenzahl (hoher Ressourcenverbrauch) oder eher mit einer hohen Individuenzahl (niedriger Ressourcenverbrauch) arbeitet.

#### **Konkurrenzintervall und Konkurrenzrate**

Das Konkurrenzintervall gibt an, zu welchen Zeitpunkten ein Wettbewerb zwischen den Unterpopulationen und damit eine Umverteilung der Ressourcen zwischen den Unterpopulationen stattfindet. Damit wird definiert, wie lange die Unterpopulationen ohne Veränderung der ihnen zur Verfügung stehenden Ressourcen existieren bzw. wie lange sie Zeit haben, bevor sie im Vergleich zu den anderen Unterpopulationen ihre Ergebnisse zeigen müssen.

Das Konkurrenzintervall kann als eine fest definierte Zahl von Generationen gegeben sein. Der Wert kann zwischen 1 und einer größeren Zahl von Genera-

tionen gewählt werden. Der Zeitpunkt für einen Wettbewerb kann aber auch in Abhängigkeit von anderen Dingen als der Anzahl vergangener Generationen definiert werden. Eine Möglichkeit besteht zum Beispiel darin, immer dann einen Wettbewerb durchzuführen, wenn in einer Unterpopulation ein deutlicher Fortschritt zu verzeichnen war<sup>3</sup>.

Die Unterpopulationen, die bei einem Wettbewerb als erfolglos eingestuft wurden, müssen Ressourcen abgeben. Dasselbe gilt für Unterpopulationen, deren bisheriger Anteil an den Gesamtressourcen höher ist, als er sich aus der neuen Verteilung der Ressourcen ergibt. Der maximale Umfang an Ressourcen, der in einem Wettbewerb abgegeben werden kann, wird durch die Konkurrenzrate festgelegt.

Die Konkurrenzrate wird als Anteil der Ressourcen der Unterpopulation, und nicht als fester Wert, angegeben. Dadurch wird gewährleistet, daß Unterpopulationen mit wenigen Ressourcen im Vergleich zu Unterpopulationen mit vielen Ressourcen einen absolut gesehen kleineren Betrag abgeben müssen. Die Konkurrenzrate sollte immer nur einen Teil der Ressourcen einer Unterpopulation umfassen.

Im folgenden werden Richtwerte für Konkurrenzintervall und Konkurrenzrate aufgeführt, die sich als gute Startwerte erwiesen haben.

Konkurrenzintervall (abhängig von der durchschnittlichen oder anfänglichen Anzahl der Individuen pro Unterpopulation):

$$\text{Konkurrenzintervall} = 20\% \cdot \text{AnzIndUnderPop} \quad [\text{Generationen}] \quad (4-10)$$

(10% – 50% ⇒ 4 – 20 Generationen)

Konkurrenzrate:

$$\text{Konkurrenzrate} = 10\% \quad [\text{Ressourcen der Unterpopulation}] \quad (4-11)$$

(5% – 20% der Ressourcen der Unterpopulation)

Konkurrenzintervall und Konkurrenzrate zusammen ergeben die Geschwindigkeit, mit der eine Umverteilung der Ressourcen zwischen den Unterpopulationen stattfindet.

Aus den oben genannten Richtwerten ergibt sich eine Umverteilungsgeschwindigkeit von:

$$\begin{aligned} \text{UmverteilungGeschwindigkeit} &= \frac{\text{Konkurrenzrate}}{\text{Konkurrenzintervall}} \\ &= 0.5\% – 4\% \quad \left[ \frac{\% \text{ der Unterpopulation}}{\text{Generation}} \right] \end{aligned} \quad (4-12)$$

<sup>3</sup> Die Definition eines „deutlichen Fortschritts“ ist eine problemspezifische Entscheidung und kann nicht allgemein beantwortet werden. Eine Möglichkeit ist, daß ein besseres Individuum in einer Unterpopulation gefunden wurde. Bei vernünftiger Einstellung der Parameter und der Problemdefinition ist dieses Kriterium aber zu einschränkend, da in fast jeder Generation etwas besseres gefunden wird. Erst bei einem speziellen Problem läßt sich ein deutlicher Fortschritt eindeutig und sinnvoll definieren (Größe der Veränderung des Zielfunktionswertes oder prozentuale Verbesserung gegenüber einem bestimmten Richtwert).

Bei einem sehr kleinen Wert für das Konkurrenzintervall (< 6 Generationen) sollte die Konkurrenzrate auch sehr klein sein, da sonst eine zu schnelle Umverteilung der Ressourcen zwischen den Unterpopulationen stattfindet. Ist die Konkurrenzrate höher gewählt (> 10%), sollte auch das Konkurrenzintervall größer gewählt werden. Die Umverteilungsgeschwindigkeit ermöglicht einen einfachen Vergleich verschiedener Werte für beide Parameter.

### **Konkurrenzauswahl**

Die Konkurrenzauswahl gibt an, wie die Individuen ausgewählt werden, die von einer Unterpopulation abgegeben werden müssen.

Für die Auswahl der Individuen gibt es verschiedene Möglichkeiten:

- die schlechtesten Individuen,
- gleichmäßig verteilt zufällig ausgewählte Individuen und
- die besten Individuen.

Für die Auswahl der schlechtesten Individuen spricht, daß dadurch die erfolglosen Unterpopulationen nicht noch zusätzliche Nachteile im Wettbewerb haben. Bei der Abgabe der besten Individuen würden die erfolglosen Unterpopulationen noch weiter benachteiligt, als dies durch die Verringerung der Individuenzahl ohnehin schon geschieht. Von der Wertigkeit genau dazwischen liegt eine zufällig verteilte Auswahl der Individuen.

Die Auswahl der besten oder schlechtesten Individuen kann unter Verwendung eines der vorgestellten Selektionsverfahren aus Abschn. 3.2 erfolgen.

### **Unterpopulationsminimum**

Um zu verhindern, daß eine erfolglose Unterpopulation vollständig verschwindet, muß eine minimale Unterpopulationsgröße bzw. ein minimaler Anteil an Ressourcen festgelegt werden, der jeder Unterpopulation immer erhalten bleibt. Nur bis zum Erreichen dieses Minimums können Ressourcen abgegeben werden, danach bleibt ihre Größe konstant (bis diese Unterpopulation vielleicht wieder einmal erfolgreich wird).

Das Unterpopulationsminimum kann angegeben werden als:

- feste Anzahl von Individuen,
- Anteil der durchschnittlichen Größe der Unterpopulationen oder
- Anteil der insgesamt zur Verfügung stehenden Ressourcen.

Die Angabe als Anteil der Unterpopulationsgröße hat, gegenüber einer festen Individuenzahl, den Vorteil der besseren Vergleichbarkeit zwischen unterschiedlich großen Unterpopulationen. Oftmals ist aber bekannt, wie groß eine Unterpopulation mindestens sein muß, um noch arbeiten zu können. Die flexibelste Angabe ist die Definition als Anteil der insgesamt zur Verfügung stehenden Ressourcen.

$$\text{Unterpopulationsminimum} = \begin{cases} 6 \quad (4-10) \quad [\text{Individuen}] \\ 20\% \quad (10\%-30\%) \quad [\% \text{ der Unterpopulationsgröße}] \\ 4\% \quad (1\%-10\%) \quad [\% \text{ der Gesamtressourcen}] \end{cases} \quad (4-13)$$

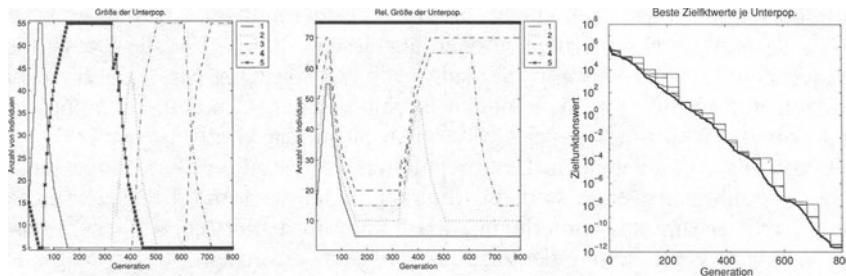
In Gl. 4-13 werden Richtwerte für das Unterpopulationsminimum aufgeführt, die sich als sinnvolle Werte erwiesen haben. Wenn das Unterpopulationsminimum sehr klein gesetzt wird (4-6 Individuen), ist die Funktion der meisten Evolutionären Algorithmen in einer Unterpopulation mit so wenigen Individuen kaum noch gewährleistet. Ein Erfolg dieser sehr kleinen Unterpopulation gegenüber anderen größeren Unterpopulationen tritt dann nur noch ein, wenn die anderen Unterpopulationen mit ihren Strategien deutlich schlechter sind.

Bei einem Unterpopulationsminimum von 10% der Gesamtressourcen, 5 Unterpopulationen und davon einer erfolgreichen Unterpopulation erhält die erfolgreiche Unterpopulation nach einigen Wettbewerben 60% der Gesamtressourcen. Die erfolglosen 4 Unterpopulationen dagegen haben jede nur 10% der Gesamtressourcen zur Verfügung. Dies entspricht einem Verhältnis von 6:1 (erfolgreiche zu erfolgloser Unterpopulation) bzw. 3:2 (erfolgreiche Unterpopulation zu allen erfolglosen Unterpopulationen). Bei einer weiteren Verringerung des Unterpopulationsminimums würde sich dieses Verhältnis weiter verschieben. Wenn eine gewichtete Verteilung der Ressourcen auf die Unterpopulationen gewählt wird, kann das Unterpopulationsminimum, im Vergleich zur Verwendung einer einzigen erfolgreichen Unterpopulation, kleiner gewählt werden. Bei einer gewichteten Verteilung erreichen nur die wirklich schlechtesten Unterpopulationen das Unterpopulationsminimum, die anderen bekommen einen höheren Anteil der Ressourcen zur Verfügung gestellt.

#### **4.6.4 Beispiel des Einsatzes konkurrierender Unterpopulationen**

In diesem Unterabschnitt soll der Einsatz konkurrierender Unterpopulationen demonstriert werden. Das Beispiel benutzt dieselbe Testfunktion, die bei der Demonstration des Einsatzes unterschiedlicher Strategien verwendet wurde (DEJONG's Funktion 1 oder Hypersphäre). Die Optimierung wurde mit denselben Parametern und Strategien durchgeführt. Jede der 5 Unterpopulationen verwendete eine andere Strategie, die sich in der Größe der verwendeten Mutationsschritte unterschieden (1: große Mutationsschritte; 2: mittlere Mutationsschritte; 3: kleine Mutationsschritte; 4: winzige Mutationsschritte; 5: große und mittlere Mutationsschritte zusammen). Je kleiner die Mutationsschritte sind, um so lokaler ist die Suche dieser Strategie. Alle Strategien verwendeten denselben Rekombinationsoperator, alle weiteren Parameter waren für alle Strategien identisch (z.B. *generation gap* = 0,9). Zu Beginn des Laufs hatten alle Unterpopulationen eine Größe von 15 Individuen. Alle 40 Generationen fand Migration zwischen den Unterpopulationen statt. Dabei wanderten die besten Individuen in einer vollständigen Netzstruktur.

Die Parameter der Konkurrenz waren wie folgt eingestellt: nur die beste Unterpopulation erhält Ressourcen, Ressourcenverbrauch für alle Individuen 1, Konkurrenzintervall von 4 Generationen, die schlechtesten Individuen einer Unterpopulation werden abgegeben, Unterpopulationsminimum von 5 Individuen. Dieses Beispiel stellt damit eine der einfachsten Varianten konkurrierender Unterpopulation dar, die gleichzeitig gut zu überblicken ist. Der Verlauf und die Ergebnisse eines solchen Optimierungslaufs sind in den Abbildungen 4-14 und 4-15 dargestellt.



**Abb. 4-14.** Einsatz konkurrierender Unterpopulationen; links: Größe der Unterpopulationen, Mitte: relative Größe der Unterpopulationen, rechts: bester Zielfunktionswert je Unterpopulation

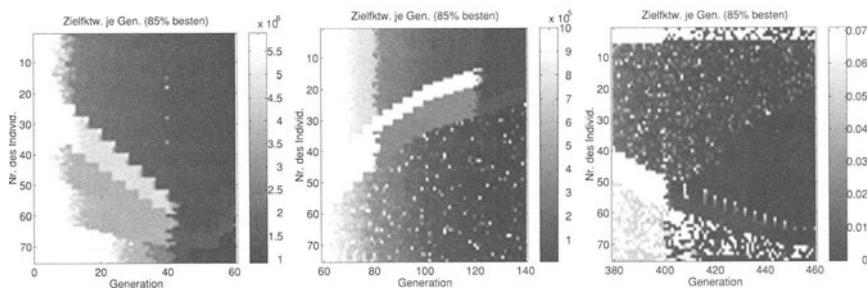
Einen guten Überblick über den Verlauf der Konkurrenz zwischen den Unterpopulationen geben die linke und mittlere Grafik in Abb. 4-14. Die linke Grafik zeigt die absolute Größe der Unterpopulationen, die mittlere Grafik die relative Größe der einzelnen Unterpopulationen im Verlauf der Optimierung. Bei der mittleren Grafik ergibt sich die Anzahl der Individuen einer Unterpopulation aus der Differenz zwischen zwei Linien. Im Gegensatz zur Darstellung der Größe der Unterpopulationen kann aus dieser Darstellung auch ohne Legende die Zuordnung der Größe einer Unterpopulation zur Nummer der Unterpopulation erfolgen.

Zu Beginn des Laufs steigt die Größe von Unterpopulation 1 (große Mutationsschritte) kurz an. Etwa ab Generation 50 wächst Unterpopulation 5 (große und mittlere Mutationsschritte). Erst nach Generation 320 wird Unterpopulation 2 besser (mittlere Mutationsschritte), ab Generation 400 erhält Unterpopulation 3 (kleine Mutationsschritte) mehr Individuen. Die letzten knapp 200 Generationen ist Unterpopulation 4 (winzige Mutationsschritte) am erfolgreichsten und hat die meisten Individuen. Die erfolglosen Unterpopulationen verfügen zu jedem Zeitpunkt nur über einen geringen Anteil der Individuen der Gesamtpopulation.

In Abb. 4-14 rechts ist der Verlauf der besten Zielfunktionswerte für jede Unterpopulation dargestellt. Die Grafik zeigt, daß zu jedem Zeitpunkt einige der Unterpopulationen keine Verbesserung erreichen. Erst nach einer Migration verbessert sich der Wert des besten Zielfunktionswertes dieser Unterpopulationen. Im Verlauf der Optimierung sind es aber immer unterschiedliche Unterpopulationen, die keine Verbesserung mit ihrer Strategie erreichen bzw. aus einer erfolglosen Unterpopulation wird im Verlauf der Optimierung eine erfolgreiche Unterpopulation und umgekehrt. Die Übergänge von einer Strategie zur nächsten sind gut zu erkennen (um Generation 440 und Generation 600). Die sich verlangsamt Konvergenz kann durch die neue Strategie wieder beschleunigt werden. Über den gesamten Lauf gesehen wird eine gleichmäßige Konvergenzgeschwindigkeit erreicht.

Einen tieferen Einblick in den Verlauf der Optimierung zeigt Abb. 4-15. Alle Grafiken zeigen die Zielfunktionswerte aller Individuen der Population über mehrere Generationen (die Individuen mit den niedrigsten Nummern bilden Unterpopulation 1; die Individuen mit den höchsten Nummern Unterpopulation 5; dazwischen liegen die anderen Unterpopulationen). Die Trennung zwischen den

Unterpopulationen ist deutlich zu erkennen. Dadurch kann auch indirekt die Größe der einzelnen Unterpopulationen abgelesen werden. Für die Darstellung wurden Zeiträume ausgewählt, die den Übergang von einer Strategie zur nächsten zeigen. Die linke Grafik zeigt den Beginn des Laufs, in dem Unterpopulation 1 erfolgreich war und in der Größe anwuchs. In der Grafik ist der Grund für diesen Erfolg zu erkennen: in Unterpopulation 1 findet die größte Verbesserung in den Zielfunktionswerten statt. In Unterpopulation 2, 3 und 4 dagegen gibt es kaum Verbesserung, nur Unterpopulation 5 kann in den ersten 40 Generationen eine sichtbare Verbesserung der Zielfunktionswerte erreichen. Mit der Migration in Generation 40 erfolgt eine Angleichung der Zielfunktionswerte zwischen den Unterpopulationen, ein Effekt, der auch in Abb. 4-14 Mitte abzulesen ist.



**Abb. 4-15.** Einsatz konkurrierender Unterpopulationen – Zielfunktionswerte aller Individuen; links: Beginn des Laufs, Mitte: Mitte des Laufs, rechts: Ende des Laufs

Nach der Migration in Generation 40 verändert sich das Bild. Ab Generation 60 (Abb. 4-15 Mitte) steigt die Größe von Unterpopulation 5 an. Die Zielfunktionswerte dieser Unterpopulation zeigen die stärkste Verbesserung. Unterpopulation 1 kann zwar auch noch die Zielfunktionswerte verbessern, allerdings nicht so stark wie Unterpopulation 5. Die Unterpopulationen 2, 3 und 4 zeigen weiterhin keine Verbesserung der Zielfunktionswerte, außer der Angleichung durch die Migration in Generation 80.

In Abb. 4-15 rechts ist ein weiterer Übergang zwischen den Strategien dargestellt. Unterpopulation 2 wächst noch bis Generation 400. Danach wird Unterpopulation 3 am erfolgreichsten. Dies ist an der gleichmäßigen Verbesserung der Zielfunktionswerte zu erkennen, die ab Generation 420 mit einer Zunahme der Unterpopulationsgröße verbunden ist. Unterpopulation 2 kann zwar auch gute Zielfunktionswerte erreichen, daneben sind in Unterpopulation 2 aber auch viele deutlich schlechtere Individuen enthalten.

Die Auswertung eines Laufs mit Konkurrenz zwischen Unterpopulationen mit verschiedenen Strategien liefert dem Anwender Informationen über die Verteilung der Ressourcen, wann welche Strategie erfolgreich ist und wie sich die Zielfunktionswerte verändern. Durch die Kombination der hier vorgestellten 3 Visualisierungsmethoden (Abb. 4-14 links, Abb. 4-14, Mitte und rechts sowie Abb. 4-15) lassen sich diese Informationen einerseits leicht erfassen, andererseits kann durch Auswahl der Methoden die Menge bzw. Tiefe an Information be-

stimmt werden, die zur Verfügung steht bzw. verwendet werden soll. Die Größe der Unterpopulationen zeigt direkt die Verteilung der Ressourcen und gibt damit ein Abbild des Erfolgs der Unterpopulationen. Die Darstellung des besten Zielfunktionswertes je Unterpopulation gibt eine erste direkte Maßzahl für den Erfolg einer Unterpopulation. Mit der Darstellung aller Zielfunktionswerte werden alle Daten veranschaulicht, die für die Berechnung des Erfolgs der Unterpopulationen verwendet werden. Diese Grafik (Abb. 4-15) gibt den tiefsten Einblick in den Zustand der Unterpopulationen und lässt detaillierte Rückschlüsse über den Erfolg einer Suchstrategie zu.

Ein weiteres Beispiel des Einsatzes konkurrierender Unterpopulationen wird in Abschn. 8.4.5, ab S.232, bei der Optimierung der Parameter eines Reglers gezeigt.

#### 4.6.5 Analyse der konkurrierenden Unterpopulationen

Der Einsatz miteinander konkurrierender Unterpopulationen hat zwei Vorteile gegenüber der Anwendung eines normalen regionalen Modells: erstens werden zur gleichen Zeit verschiedene Strategien angewendet und zweitens wird denjenigen Strategien, die im Moment die besten Ergebnisse erzielen, ein größerer Teil der Ressourcen zur Verfügung gestellt. Auf diesem Weg findet eine effektive Verteilung der Ressourcen auf die vom Anwender vorgegebenen Strategien während des gesamten Laufs statt. Es ist nicht notwendig, für die Verteilung der Ressourcen einen Plan von außen vorzugeben bzw. die Veränderung von Strategieparametern zu Beginn eines Laufs festzulegen.

Die Vorteile konkurrierender Unterpopulationen eröffnen in der praktischen Anwendung in mehreren Bereichen ein neues bzw. verändertes Vorgehen. Bei der Bearbeitung eines Problems mit einem Evolutionären Algorithmus muß man sich nicht auf einen Satz von Parametern beschränken, sondern es können mehrere gleichzeitig angewendet werden. In der Auswertung dieser Läufe kann ermittelt werden, welche Strategien keinen oder einen sehr geringen Beitrag zur Lösung des Problems beitragen. Diese Information liefert oftmals neue Erkenntnisse über den Typ des Problems und damit ein erweitertes Verständnis. Dasselbe gilt für die Informationen, die aus dem Erfolg einer Strategie gezogen werden können. Bei der weiteren Bearbeitung dieses Problems können die erfolglosen Strategien durch neue Strategien ersetzt werden, die entweder auf den bisher erfolgreichen Strategien beruhen oder auf Grund der neuen Informationen über das Problem als vielversprechend eingeschätzt werden. Innerhalb einer vernünftigen Anzahl von Läufen läßt sich damit aus einem Repertoire vorhandener Strategien die für ein konkretes Problem erfolgreichste Strategie bzw. erfolgreiche Kombination von Strategien ermitteln.

Bisher wurden erst wenige Arbeiten zur Beschreibung und Anwendung konkurrierender Unterpopulationen veröffentlicht.

In [SVM94] wird eine einfache Variante konkurrierender Unterpopulationen zur Strategieanpassung vorgestellt. Es wird ein Ressourcenverbrauch von 1, ein Konkurrenzintervall von 4 Generationen, eine Konkurrenzrate von 1/8 und ein Unterpopulationsminimum von 4 Individuen benutzt. Eine Ordnung der Unterpopulationen wurde nicht berechnet, sondern es wurde nur die beste Unterpopu-

lation ermittelt. Alle Unterpopulationen außer der besten mußten Individuen abgeben, die Konkurrenzauswahl erfolgte zufällig und die beste Unterpopulation erhielt alle freien Ressourcen. Eine Migration fand alle 16 Generationen statt. Während einer Migration wanderte das global beste Individuum in alle anderen Unterpopulationen, die beste Unterpopulation erhielt kein neues Individuum. Andere Werte für die Parameter der miteinander konkurrierenden Unterpopulationen bzw. deren Ergebnisse wurden nicht vorgestellt. SCHLIERKAMP-VOOSEN und MÜHLENBEIN zeigen die Anwendung dieser einfachen Variante konkurrierender Unterpopulationen zur Optimierung bekannter Standardtestfunktionen (Hypersphäre, ROSENBROCK's Funktion, GRIEWANGK's Funktion).

In [SVM96] wird das Konzept konkurrierender Unterpopulationen zur Anpassung der Individuenzahl der Gesamtpopulation erweitert (*extended competition model*). Der zusätzliche Parameter des Ressourcenverbrauchs, *consumption factor*, wurde eingeführt. Dadurch konnte die relative Größe der Unterpopulationen gesteuert werden. Diese Erweiterung des Konzepts konkurrierender Unterpopulationen zeigt ihre Vorteile, wenn Strategien zum Einsatz kommen, die am effektivsten mit stark unterschiedlichen Populationsgrößen arbeiten. Als Beispiel wurde die Konkurrenz zwischen einer Unterpopulation, deren viele Individuen (geringer Ressourcenverbrauch) nur rekombiniert werden und einer Unterpopulation, deren wenige Individuen (hoher Ressourcenverbrauch) nur mutiert werden, vorgestellt. Wie nicht anders zu erwarten, war zu Beginn des Laufs die Strategie mit der Rekombination am erfolgreichsten. Im weiteren Verlauf wurde die Mutation erfolgreicher. Als Folge sank die Gesamtzahl der Individuen im Verlauf der Optimierung deutlich ab.

Die von SCHLIERKAMP-VOOSEN und MÜHLENBEIN vorgestellte Variante konkurrierender Unterpopulationen stellt die einfachste Variante dar. Die größte Vereinfachung betrifft die Aufteilung der Ressourcen, die nur vom besten Individuum der gesamten Population abhängt. Dies hat zur Folge, daß eine Unterpopulation immer erfolglos ist, wenn in ihr nicht das global beste Individuum enthalten ist. Mit dieser Methode wird immer nur eine Strategie bevorzugt, alle anderen werden benachteiligt. Dies führt während eines Laufs, wenn die bisher beste Strategie in ihrem Suchverhalten nicht mehr so gut wie die nächstbeste ist, zu einer uneffektiven Verteilung der Ressourcen aus folgendem Grund: Wenn eine Unterpopulation mit mehr Ressourcen/Individuen arbeitet, ist die Wahrscheinlichkeit für das Finden besserer Lösungen größer. Da die beste Unterpopulation nur auf Grund des besten Individuums bestimmt wird, bleibt diese Unterpopulation solange beste Unterpopulation, bis ihre Suchstrategie wesentlich schlechter als die einer anderen Unterpopulation ist (der Nachteil der schlechteren Strategie kann durch die deutlich größeren zur Verfügung stehenden Ressourcen ausgeglichen werden). Zu einem Zeitpunkt, an dem längst eine andere Strategie erfolgreicher wäre, wenn beide Strategien mit denselben Ressourcen/derselben Individuenzahl arbeiten würden, kann sich die vorher bessere Unterpopulation noch behaupten.

Lösungen für dieses Problem wurden in diesem Abschnitt aufgezeigt. Die Einbeziehung der besten Individuen aller Unterpopulationen bzw. aller Individuen der Population ermöglicht die Bestimmung einer gewichteten Ordnung der Unterpopulationen, die eine besser abgestufte Aussage zum Erfolg der Unterpopulationen macht. Durch die zusätzliche gewichtete Aufteilung der Ressourcen auf

alle Unterpopulationen wird eine Verteilung der Ressourcen erreicht, die nicht nur der besten Unterpopulation (bzw. der Unterpopulation mit dem besten Individuum) zugute kommt, sondern mehreren guten Unterpopulationen. Insbesondere beim Einsatz ähnlicher bzw. sich ergänzender Strategien führt dies zu einer „gerechteren“ Verteilung der Ressourcen.

Von der Anwendung des Prinzips konkurrierender Unterpopulationen auf ein reales Problem, die automatische Generierung von Testsequenzen für digitale Schaltungen, wird in [CPR96] berichtet. Mehrere Unterpopulationen, die alle mit demselben Parametersatz arbeiten, werden jeweils zu einer Gruppe zusammengefaßt. Die Ressourcen werden nicht durch Verlagerung von Individuen zwischen erfolglosen und erfolgreichen Gruppen verteilt, sondern eine erfolglose Unterpopulation erhält die Parameter einer erfolgreichen Unterpopulation und wechselt dadurch ihre Gruppenzugehörigkeit. Dadurch ist zwar keine so feine Verteilung der Ressourcen möglich, jedoch ist dieser Ansatz einfacher in der Anwendung auf parallelen Rechnerarchitekturen. Hinweise auf die verwendeten Parameter und deren Einstellung werden nicht gegeben.

## 4.7 Zusammenfassung

In diesem Kapitel wurden Populationsmodelle für die Anwendung bei Evolutio- nären Algorithmen vorgestellt, erläutert und analysiert. Besonderes Augenmerk wurde auf die ausführliche Darstellung der verschiedenen Parameter der Popula- tionsmodelle gelegt. Durch diese umfassende Beschreibung ist ein genaues Ver- ständnis der Populationsmodelle möglich. Außerdem kann durch die Erläuterung der Auswirkungen der einzelnen Parameter sowie deren Wechselwirkungen das Verhalten des Gesamtsystems besser eingeschätzt werden. Für alle Parameter wurden Empfehlungen für die praktische Anwendung gegeben.

Die ausführliche Erläuterung der Populationsmodelle erlaubt die einfache Ein- ordnung „neuer Populationsmodelle“ in die hier verwendete Klassifikation.

Auf der Grundlage dieser Ausführungen konnten zwei Erweiterungen des re- gionalen Modells eingeführt werden, die bisher noch nicht bzw. nur in einfachen Varianten berichtet wurden.

Die Anwendung verschiedener Strategien erlaubt die gleichzeitige Verwen- dung verschiedenster Parametereinstellungen. Dies ist in der Durchführung schneller und einfacher als mehrere unabhängige Experimente. Zusätzlich wer- den die Ergebnisse in einer kompakteren und besser zu überschauenden Weise dargeboten. Sehr einfach können erfolgreiche und erfolglose Strategien erkannt bzw. der unterschiedliche Erfolg einzelner Strategien während eines Laufs abge- lesen werden. Außerdem eröffnet die gleichzeitige Anwendung verschiedener Strategien die Möglichkeit, daß sich die Strategien innerhalb eines Laufs ergän- zen und dadurch in der gleichzeitigen Anwendung bessere Ergebnisse erreicht werden, als wenn die Strategien getrennt verwendet würden. Erst der Erfolg einer Strategie zu Beginn eines Laufs verhilft einer anderen Strategie im weiteren Verlauf zu ihrer erfolgreichen Anwendung.

Der Einsatz konkurrierender Unterpopulationen beruht auf der Anwendung verschiedener Strategien, geht aber noch eine Stufe weiter. Die verschiedenen

Strategien werden nicht nur gleichzeitig angewendet, sondern die erfolgreichen Strategien erhalten mehr Ressourcen zur Verfügung, als die weniger erfolgreichen. Dadurch kommt es zu einer dynamischen Verteilung der Ressourcen auf die Strategien, die zum jeweiligen Zeitpunkt den größten Erfolg hatten.

Diese Verteilung der Ressourcen führt indirekt zu einer Anpassung der Strategieparameter während eines Laufs. Der Vorteil des Einsatzes konkurrierender Unterpopulationen gegenüber bisherigen Methoden zur Anpassung der Strategieparameter besteht darin, daß die Anpassung der Parameter nicht von außen durch den Benutzer vorgegeben werden muß, sondern in Abhängigkeit des Zustandes der Unterpopulationen geschieht. Damit eröffnen sich neue Möglichkeiten der Arbeit mit Evolutionären Algorithmen.

Die Beschränkung auf einen Satz von Parametern ist nicht mehr gegeben und die Suche nach guten Sätzen von Parametern (Strategien) für die Lösung eines Problems wird deutlich vereinfacht. Neben dem Test verschiedener Strategien können gleichzeitig mehrere Strategien verwendet werden. Durch den Einsatz konkurrierender Unterpopulationen erhalten diejenigen Strategien wenig Ressourcen, die nicht sehr gut für die Lösung des Problems geeignet sind. Dadurch wird nur ein kleiner Teil der Ressourcen für den Test dieser erfolglosen Strategien verwendet, wodurch der Test zusätzlicher erfolgversprechender Strategien auch bei sehr aufwendigen Problemen ermöglicht wird.

Die beiden hier vorgestellten Erweiterungen, die Anwendung verschiedener Strategien und der Einsatz konkurrierender Unterpopulationen, sind sehr leistungsstark. Besonders in der praktischen Anwendung haben sie eine hohe Bedeutung. Beide Erweiterungen sind ein weiterer Schritt zur Entwicklung leistungsfähiger Evolutionärer Algorithmen, die besser auf die Lösung großer und schwieriger Probleme ausgerichtet sind.

## 5 Visualisierung und Optimierung

Evolutionäre Algorithmen erreichen ein komplexes Verhalten durch eine Anzahl von Verfahren, die in ihren algorithmischen Grundlagen einfach strukturiert sind. Dabei werden in einem sich immer wiederholenden Ablauf große Mengen an Daten produziert. Die in diesen Verfahren enthaltenen Operationen und die produzierten Daten geben dem Betrachter keine oder nur geringe Einsicht in das Verhalten, die Funktion bzw. die Auswirkungen des Evolutionären Algorithmus, sie tragen für sich genommen nur wenig zum Verständnis bei. Erst durch eine Auswahl und Zusammenfassung der Daten und eine Abstraktion der Einzelschritte ist es möglich, das Verhalten der Algorithmen zu verstehen bzw. zu vergleichen und die Ergebnisse ihrer Anwendung zu interpretieren. Die vielen kleinen Einzelteile, die jedes für sich wenig bis nichts bedeuten, müssen zu einem großen Ganzen zusammengefügt werden, um ein aussagefähiges und interpretierbares Gebilde zu entwickeln. Eines der leistungsfähigsten Werkzeuge dafür ist die Visualisierung dieser Daten.

Die Visualisierung lässt sich in zwei große Bereiche unterteilen: *Programm-Visualisierung* und *Algorithmen-Visualisierung* ([Col95]). Bei der *Programm-Visualisierung* werden die grundlegenden Bausteine eines Programms, der Programmcode oder die Programmdaten dargestellt. Bei der *Algorithmen-Visualisierung* geht es um die Darstellung des Programmalgorithmus – seiner grundlegenden Funktion. Die im folgenden vorgestellten Methoden und Verfahren können dem Bereich der *Algorithmen-Visualisierung* zugeordnet werden.

Für ein Verständnis der Funktion und des Effekts Evolutionärer Algorithmen ist es für den Benutzer wichtig, Werkzeuge in die Hand zu bekommen, die schnell und effektiv alle verfügbaren und berechenbaren Informationen vor ihm ausbreiten und erklären bzw. sichtbar machen. Im weiteren sollen die Daten und Werte ausgesucht, untersucht und dargestellt werden, die einen guten Überblick über die Funktion und Arbeitsweise des Evolutionären Algorithmus geben. Gleichzeitig sollen dadurch mit vertretbarem Aufwand Aussagen über bestimmte Meßwerte oder Eigenschaften eines Individuums, einer Generation, eines Laufs oder einer Konfiguration gemacht werden. Die dabei vorgestellten Methoden und Verfahren können relativ einfach auf eine konkrete Anwendung übertragen und eingesetzt werden.

Viele der praktischen Anwendungen Evolutionärer Algorithmen benutzen bei der Visualisierung nur die Darstellung des Verlaufs der Zielfunktionswerte vom besten und „mittleren“ Individuum über den Verlauf der Generationen, meist als Fitneßkurve bezeichnet. Dies ist jedoch nur ein kleiner Ausschnitt aus der großen Datenvielfalt, die im Verlauf der Berechnungen zur Verfügung stehen. Zudem erhält man mit dieser Darstellung kaum einen Einblick in die Arbeitsweise des Evolutionären Algorithmus.

In diesem Kapitel werden Methoden und Techniken zur übersichtlichen und effektiven visuellen Darstellung des Verlaufs und der Ergebnisse eines Evolutionären Algorithmus sowie zum Vergleich verschiedener Evolutionärer Algorithmen untereinander genannt, systematisiert und erläutert. Weiterhin werden Möglichkeiten der Visualisierung von Eigenschaften der Zielfunktion untersucht. Besonderer Wert wird auf eine umfassende Darstellung verschiedener Techniken für die unterschiedlichen Anwendungsfälle und deren Bewertung für die praktische Anwendung gelegt. Aus der Literatur bekannte Methoden und Systematisierungen werden untersucht, in ihrer Anwendbarkeit bewertet und in das entwickelte Schema eingegordnet.

Abschnitt 5.1 nimmt eine Klassifizierung von Visualisierungstechniken vor, die in den nachfolgenden Abschnitten in ihrer Anwendung und Darstellung detailliert behandelt werden. Generationsübergreifende Visualisierungsmöglichkeiten werden in Abschn. 5.2 vorgestellt. Abschnitt 5.3 zeigt Verfahren, die ein Abbild der aktuellen Population geben. Ein Verfahren zur Visualisierung hochdimensionaler Daten und deren Anwendung auf Evolutionäre Algorithmen wird in Abschn. 5.4 untersucht. Die Visualisierung allgemeiner Eigenschaften der Zielfunktion zeigt Abschn. 5.5. Auf Fragen der Protokollierung von Daten und Ergebnissen, die mit der Visualisierung in engem Zusammenhang stehen, wird in Abschn. 5.6 eingegangen.

## **5.1 Systematisierung der Visualisierung von EA**

Eine Klassifizierung der Visualisierung Evolutionärer Algorithmen lässt sich auf verschiedenen Ebenen vornehmen. Im folgenden wird eine Unterteilung nach mehreren Kriterien vorgenommen, die in ihrer Kombination eine feine Systematisierung erlauben. Wenn Methoden von der Repräsentation der Werte abhängig sind, wird dies bei der ausführlichen Erläuterung der einzelnen Techniken in den folgenden Abschnitten erwähnt. Wenn möglich, wird für die gebräuchlichsten Repräsentationen eine praktische Realisierungsmöglichkeit aufgezeigt.

Zu Beginn erfolgt eine Aufstellung aller Kriterien, die für die weitere Systematisierung in Betracht gezogen werden müssen bzw. als Kriterien für eine Unterteilung herangezogen werden können. Aus der Kombination der einzelnen Unterpunkte der Kriterien ergeben sich im weiteren die zu charakterisierenden Bereiche. Für jeden dieser Bereiche können Techniken der Visualisierung abgeleitet bzw. entwickelt werden.

### **5.1.1 Kriterien der Systematisierung**

Eine erste Unterteilung wird entsprechend dem Zeitintervall der verwendeten Daten vorgenommen:

- einzelne Individuen (individuelle Daten und Eigenschaften),
- eine/aktuelle Generation (lokale Daten und Eigenschaften),
- alle Generationen eines Laufs (globale Daten und Eigenschaften) und
- mehrere Läufe (Vergleich globaler Daten und Eigenschaften).

Die zweite Unterteilung bezieht sich auf die Anzahl der Individuen, für die gleichzeitig Werte dargestellt werden und dient einer gezielten Verfeinerung der ersten Unterteilung:

- einzelne Individuen,
- alle Individuen einer (Unter-)Population und
- Individuen mehrerer (Unter-)Populationen.

Eine dritte Unterteilung kann danach vorgenommen werden, ob:

- der Verlauf / die Arbeit eines Evolutionären Algorithmus dargestellt oder
- ob Einblick in verschiedene Eigenschaften der Zielfunktion gegeben werden soll. (Diese sind im engeren Sinne nicht speziell auf Evolutionäre Algorithmen abgestimmt, werden aber bei der Arbeit mit Evolutionären Algorithmen oft benötigt. Erst mit etwas Wissen über die Zielfunktion kann oftmals der richtige bzw. ein gut geeigneter Evolutionärer Algorithmus ausgewählt werden.)

Zusätzlich spielt die Repräsentation der Variablen des Problems eine wichtige Rolle. Je nach Typ der Variablen (Repräsentation bzw. Kodierung) müssen bei einigen Visualisierungstechniken verschiedene Darstellungsformen angewendet werden. Manche Varianten sind nur für einzelne dieser Repräsentationen berechenbar bzw. sinnvoll. Deutlich komplizierter wird eine Visualisierung von Werten, wenn diese nicht als ein- oder mehrdimensionaler Vektor, sondern als komplexe Struktur (Baum o.ä.) vorliegen. Im folgenden wird davon ausgegangen, daß die Werte in einer Matrixstruktur vorliegen.

### 5.1.2 Daten zur Visualisierung

Für die Visualisierung werden zwei Arten von Daten verwendet:

- direkte Werte der Individuen bzw. Populationen (mit diesen arbeitet der Evolutionäre Algorithmus bzw. liefert sie als Ergebnis) und
- abgeleitete Werte für Individuen bzw. Populationen (diese werden aus direkten Werten berechnet und charakterisieren Eigenschaften /Zustände).

**Tabelle 5-1.** Arten der Daten, die zur Visualisierung verwendet werden

direkte Werte	abgeleitete Werte
<ul style="list-style-type: none"> <li>• Variablen der Individuen,</li> <li>• Zielfunktionswerte (Güte) der Individuen,</li> <li>• Anzahl der Individuen der Population (Unterpopulationen),</li> <li>• Anzahl der Generationen,</li> <li>• Parameter des Evolutionären Algorithmus,</li> <li>• erfolgreicher Abschluß eines Laufs</li> </ul>	<ul style="list-style-type: none"> <li>• Veränderung der Zielfunktionswerte (Ableitung, Differenz),</li> <li>• Verteilung oder Standardabweichung der Zielfunktionswerte,</li> <li>• Abstände der Individuen voneinander (Distanz),</li> <li>• Verteilung der Abstände/Distanzen zwischen den Individuen</li> </ul>

### 5.1.3 Schema der Systematisierung

Mit den aufgestellten Kriterien und den zur Verfügung stehenden Daten zur Visualisierung kann ein Schema aufgestellt werden, das eine Zuordnung der entsprechenden Werte bzw. Verfahren vornimmt. In Tabelle 5-2 ist dieses Schema vorgestellt.

Eine systematische Darstellung der Visualisierungsmöglichkeiten Evolutionärer Algorithmen ist bisher nur von SCHWEHM [Swm96] bekannt. Er unterscheidet nach der Detailliertheit der Darstellung ([Swm96], S.92):

- Gesamtverhalten einer Konfiguration:
  - Vergleich verschiedener Konfigurationen bzw. Läufe,
- globales Verhalten der Population:
  - Eigenschaften der gesamten Population über mehrere Generationen,
  - globale Maße; Konvergenz und Populationsstatistik,
- lokales Verhalten der Population:
  - Eigenschaften lokaler Nachbarschaften,
  - lokale Maße,
- spezielle Eigenschaften einzelner Individuen:
  - Genotyp und Phänotyp einzelner Individuen.

Diese Systematisierung ist im Ansatz gut und entspricht der auf S.116 vorgenommenen Unterteilung nach dem Zeitintervall der verwendeten Daten. Die Begriffe dieser Systematisierung werden deshalb dort mit aufgeführt und im folgenden an den entsprechenden Stellen zusätzlich verwendet. In der alleinigen Anwendung erweist sich diese Systematisierung allerdings als zu grob. Außerdem betrachtet SCHWEHM nur Verfahren für die binäre Repräsentation von Variablen. Einige der dort untersuchten Verfahren lassen sich nicht auf andere Repräsentationen anwenden.

Von COLLINS ([Col93], [Col97a], [Col97b]) und ROUTEN ([RC93], [Rou94]) werden verschiedene Möglichkeiten zur Visualisierung Evolutionärer Algorithmen vorgestellt. Allerdings wird keine Systematisierung der Methoden versucht. ROUTEN und COLLINS nennen eine Anzahl von Aufgaben, welche die Visualisierung erfüllen sollte ([RC93], S.278):

- Darstellung, ob die Fitneß (gemeint sind die Güte bzw. der Zielfunktionswert) der Population (noch) steigt und wie stark sich die Fitneß ändert,
- Auswirkungen der Einstellungen der Parameter verstehen,
- Verschiedenartigkeit (*diversity*) innerhalb der Population erkennen,
- welche Allele innerhalb eines Individuums werden bevorzugt und an welchen Stellen liegen diese Allele,
- Auftreten und Auswirkungen des Mutationsoperators erkennen und
- werden zwei oder mehrere mögliche Lösungen vom Evolutionären Algorithmus bevorzugt (deutet auf multimodale Zielfunktion hin)?

Alle diese Aufgaben (und einige weitere) lassen sich durch die in den folgenden Abschnitten vorgestellten Verfahren erfüllen. Eine eindeutige Zuordnung eines Verfahrens zu einer Aufgabe ist aber nicht möglich. Viele Verfahren liefern Informationen, die zur Erfüllung mehrerer dieser Aufgaben beitragen können. Bei der Erläuterung jedes Verfahrens werden die Informationen genannt, die aus der

Darstellung gewonnen werden können. Dabei erfolgt auch eine Bewertung der von COLLINS und/bzw. ROUTEN vorgeschlagenen Methoden.

**Tabelle 5-2.** Systematisierung der Visualisierungsmöglichkeiten für Verlauf, Zustand oder Abschluß eines Evolutionären Algorithmus

	<b>direkte Werte</b>			<b>abgeleitete Werte</b>
	einzelne Indiv.	Unterpopulationen	Population	Population
eine / aktuelle Generation	<ul style="list-style-type: none"> <li>• Variablen des besten Individuums</li> </ul>	<ul style="list-style-type: none"> <li>• Größe bzw. Ordnung der Unterpopulationen</li> </ul>	<ul style="list-style-type: none"> <li>• Variablen der Individuen</li> <li>• ZFW der Individuen</li> </ul>	<ul style="list-style-type: none"> <li>• Distanzverteilung</li> <li>• Distanzkarten</li> <li>• Fitneß-Allel-Tepich</li> </ul>
mehrere Generationen	<ul style="list-style-type: none"> <li>• ZFW des jeweils besten Individuums</li> </ul>	<ul style="list-style-type: none"> <li>• Größe bzw. Ordnung der Unterpopulationen</li> </ul>	<ul style="list-style-type: none"> <li>• Konvergenz: ZFW des besten Individuums</li> </ul>	<ul style="list-style-type: none"> <li>• Konvergenz: mittlerer ZFW, Standardabweichung der ZFW</li> <li>• Statistik: Diversität, Fixierung, Inzucht, Klassen</li> </ul>
mehrere Läufe	<ul style="list-style-type: none"> <li>• bester ZFW je Lauf</li> <li>• bestes Individuum je Lauf</li> </ul>	<ul style="list-style-type: none"> <li>• beste Unterpopulation / Strategie je Lauf</li> </ul>	<ul style="list-style-type: none"> <li>• Anzahl Generationen / Zielfunktionsaufrufe</li> <li>• Anzahl erfolgreicher Läufe</li> </ul>	<ul style="list-style-type: none"> <li>• Abhängigkeit des Erfolgs von veränderten Parametern</li> </ul>

## 5.2 Globale Eigenschaften einer Population

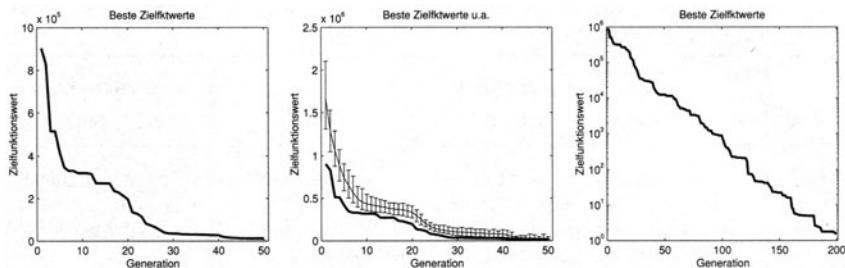
Die in diesem Abschnitt beschriebenen Methoden dienen einer schnellen und übersichtlichen Darstellung von Veränderungen der Population im Verlauf der Generationen, die durch die Wirkungsweise des verwendeten Evolutionären Algorithmus entstehen. Dies entspricht dem Verlauf bzw. Fortschritt einer Population und damit dem globalen Verhalten der Population.

### 5.2.1 Zielfunktionswert des besten Individuums der Population (Konvergenzdiagramm)

Die wohl bekannteste und fast immer genutzte Visualisierung ist die Darstellung der Konvergenz der Population. Dazu wird der Zielfunktionswert des besten Individuums über die vergangenen Generationen dargestellt. Dies gibt einen guten Überblick, wie schnell bessere Lösungen durch den verwendeten Algorithmus gefunden werden und wie groß der Fortschritt zwischen den einzelnen Generationen ist. Diese Darstellung kann mit dem Mittelwert der Zielfunktionswerte aller Individuen der Population sowie deren Standardabweichung ergänzt werden.

Abbildung 5-1 zeigt verschiedene Darstellungen des Verlaufs des besten und des durchschnittlichen Zielfunktionswertes als Liniengrafik. Die Zielfunktions-

werte einer Anzahl zurückliegender Generationen werden über den entsprechenden Generationen aufgetragen.



**Abb. 5-1.** Konvergenzdiagramm; links: Zielfunktionswert des besten Individuums über mehrere Generationen; Mitte: zusätzlich durchschnittlicher Zielfunktionswert aller Individuen und Standardabweichung der Zielfunktionswerte als Fehlergraph; rechts: wie links mit logarithmischer Skalierung

Bei der Anwendung des regionalen Modells können außer dem besten Zielfunktionswert der gesamten Population auch die jeweils besten Zielfunktionswerte der einzelnen Unterpopulationen angezeigt werden. Ein Beispiel dafür ist in Abb. 4-14, S.109 gegeben.

Bei der Darstellung sollten folgende Probleme der Skalierung der Zielfunktionswerte beachtet werden:

- Am Beginn einer Optimierung sind die Zielfunktionswerte durch die allgemein verwendete zufällige Initialisierung der Anfangspopulation oft sehr schlecht. Innerhalb weniger Generationen werden diese deutlich besser und ändern sich im folgenden Verlauf der Optimierung prozentual gesehen nur noch geringfügig. Daraus abgeleitet würde sich eine logarithmische Skalierung empfehlen. Allerdings setzt dies voraus, daß alle Zielfunktionswerte positiv sind (genauer: keine wechselnden Vorzeichen innerhalb der darzustellenden Werte haben). Eine Skalierung könnte diese Einschränkung zwar umgehen, jedoch kämen dann nicht mehr die realen Zielfunktionswerte zur Anzeige. Meist haben die Zielfunktionswerte für den Benutzer aber eine direkte Bedeutung und sollten deshalb nicht mehr als unbedingt notwendig für die Visualisierung verändert werden.
- Bei Verwendung einer linearen Skalierung ist oftmals zum Ende eines Laufs kaum eine Veränderung der Fitneßwerte in der grafischen Darstellung wahrnehmbar, da die Veränderungen der Werte im Vergleich zum Beginn des Laufs sehr klein werden. Eine Lösung ist, daß immer nur eine gewisse Anzahl der letzten Generationen dargestellt wird. Je nach Anzahl der Gesamtgenerationen variiert diese Zahl. Ein guter Richtwert sind 10%-20% der maximalen Anzahl von Generationen.
- Neben dem Zielfunktionswert des besten Individuums kann auch der mittlere Zielfunktionswert der Population als zweite Linie dargestellt werden. Aller-

dings kann es bei großen Sprüngen in den Zielfunktionswerten zwischen den besten und schlechtesten Individuen dazu kommen, daß beide Werte so weit auseinander liegen. Eine sinnvolle Darstellung in einem Diagramm ist dann nicht mehr möglich. Dies tritt z.B. auf, wenn durch verschiedene Straferme oder eine Zielfunktion in Form eines tiefen Tales viele der Individuen sehr schlechte Zielfunktionswerte zugewiesen bekommen. Bei solch unterschiedlich großen Zielfunktionswerten ist die Darstellung des mittleren Zielfunktionswertes nicht sinnvoll.

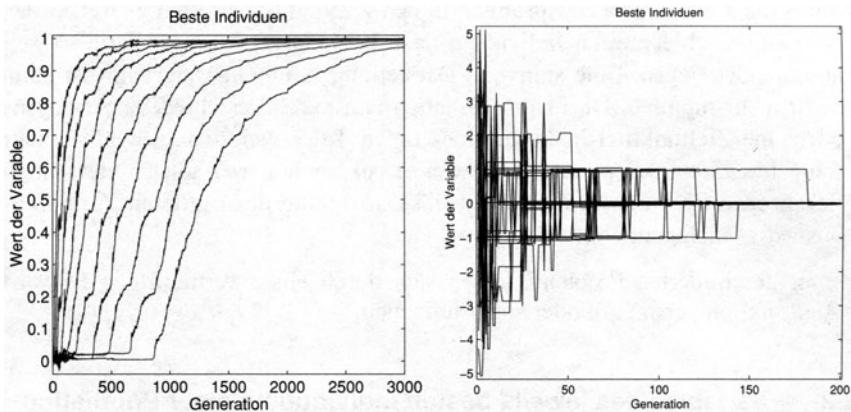
Die hier geschilderten Probleme lassen sich durch einen vernünftigen Entwurf der Zielfunktion verringern oder sogar umgehen.

### 5.2.2 Variablen des jeweils besten Individuums einer Population

Mittels Darstellung der Variablen des jeweils besten Individuums über die bisherigen Generationen wird die Möglichkeit einer tieferen Einsicht in die Vorgänge während der Optimierung gegeben. Bei der oben beschriebenen Konvergenzdarstellung wird nur ein Wert pro Generation dargestellt. Ein Überblick, wie die einzelnen Individuen aussehen, die diesen Zielfunktionswert jeweils produzieren, ist damit nicht gegeben. Mit der Visualisierung aller Variablen des jeweils besten Individuums läßt sich auf einen Blick erkennen, ob im Verlauf der Optimierung große Sprünge in den Variablen auftreten oder nur kleine Änderungen, ob mehrere Individuen mit ähnlichem Zielfunktionswert, aber deutlich unterschiedlichen Variablenwerten auftreten. Außerdem gibt diese Darstellung zusammen mit der Konvergenzdarstellung einen guten Einblick in den Zusammenhang zwischen der Größe der Veränderung der Zielfunktionswerte und der korrespondierenden Veränderung der Variablenwerte. Dies ist besonders bei einer Verlangsamung der Konvergenz wichtig. Durch einen Blick auf die Veränderung der Variablen der besten Individuen kann schnell eingeschätzt werden, ob die Optimierung festgefahren ist oder nicht.

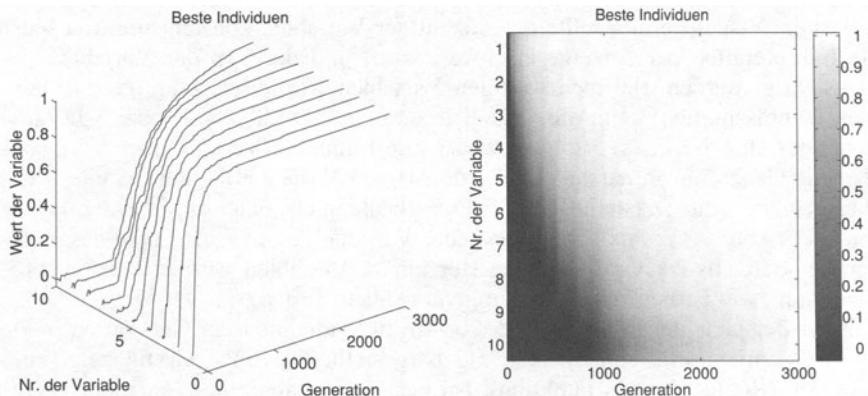
Für die Visualisierung sollte die Anzahl der Variablen konstant sein (da sonst die Interpretation der Anzeige erschwert wird) und der Typ der Variablen berücksichtigt werden. Bei mehrwertigen Variablen (reelle, ganzzahlige oder diskrete Repräsentation) kann die Visualisierung als 2-D (Abb. 5-2) oder 3-D Liniengrafik (Abb. 5-3 links) erfolgen, wobei jede Linie den Verlauf einer Variablen über die Generationen darstellt. Für jede Art von Variablen ist eine zweite Variante geeignet, die Darstellung als 2-D Farbenteppich, auch als *image plot* bezeichnet (Abb. 5-3 rechts). Der Wert jeder Variablen ist hier als Farbe des zugehörigen Bereichs der Grafik kodiert (für binäre Variablen würden hier entsprechend nur zwei Farben zur Darstellung verwendet werden).

Zwei Beispiele der Darstellung des besten Individuums jeder Generation in einer 2-D Liniengrafik sind in Abb. 5-2 dargestellt. Die linke Grafik zeigt eine Funktion (ROSEN BROCK's Funktion), bei der die Variablen untereinander korrelieren. Es gibt nur einen schmalen Weg zum Optimum, Sprünge zwischen den besten Individuen treten nicht auf, die Veränderungen sind kontinuierlich und gleichzeitig bei allen Variablen.



**Abb. 5-2.** Variablen des besten Individuums über mehrere Generationen: 2-D Liniengrafik; links: ROSENROCK's Funktion, rechts: RASTRIGIN's Funktion

Ganz anders ist das Bild des zweiten Beispiels (RASTRIGIN's Funktion), Abb. 5-2 rechts. Es gibt immer wieder starke Sprünge zwischen den Variablen, was darauf hindeutet, daß sich die jeweils besten Individuen von Generation zu Generation in mehreren Variablen unterscheiden. Dies tritt besonders nach einer Migration verstärkt auf (gut zu erkennen nach Generation 60, 80 und 100). Außerdem sind sehr gut die lokalen Minima für Variablenwerte bei Vielfachen von eins zu erkennen. Nur durch einen Sprung in einem Variablenwert um annähernd eins kann ein besserer Zielfunktionswert erreicht werden. Wie schwierig es ist, die letzte Variable auf den richtigen Wert zu bekommen, zeigt sich daran, daß dafür etwa 40 Generationen (Generationen 140 bis 180) benötigt werden.



**Abb. 5-3.** Variablen des besten Individuums über mehrere Generationen; links: 3-D Liniengrafik, rechts: 2-D Farbenteppich

In Abb. 5-3 wird derselbe Lauf wie in Abb. 5-2 links mit zwei weiteren Verfahren dargestellt. Die linke Grafik verwendet eine 3-D Liniengrafik. Diese hat den Vorteil gegenüber der 2-D Liniengrafik, daß zusätzlich die Information entnommen werden kann, welche Variable wann welchen Wert hat. Diese Information läßt sich in einer 2-D Liniengrafik nur durch eine zusätzliche Legende gewinnen, die aber bei einer größeren Anzahl von Variablen nicht anwendbar ist. Wenn dagegen die Zuordnung der Variablenwerte zu den Variablen nicht notwendig ist, ist die 2-D Liniengrafik schneller zu erfassen. Eine weitere Variante der Darstellung der Variablen der besten Individuen zeigt Abb. 5-3 rechts. Die Verwendung eines Farbenteppichs ermöglicht die schnelle Erfassung der Informationen, einschließlich der Zuordnung der Variablenwerte zur Nummer einer Variablen. Je nach Anzahl der verwendeten Farben ist eine exakte Bestimmung des Variablenwertes möglich. Die Anwendung des Farbenteppichs ist aber mehr im Bereich des schnellen Überblicks über die Veränderungen der Variablenwerte von Generation zu Generation und die Unterschiede zwischen den Variablen von Vorteil.

Bei stark unterschiedlichen Wertebereichen der einzelnen Variablen kann eine Normierung der Variablen entsprechend des vorgegebenen Wertebereiches jeder Variablen erfolgen. Dadurch kann der Verlauf von Variablen mit kleinem Wertebereich zusammen mit Variablen mit großem Wertebereich erfolgen. Ein Nachteil ist, daß dadurch die direkte Information zu den Werten der Variablen verloren geht.

### **5.2.3 Zielfunktionswerte (Güte) aller Individuen über mehrere Generationen**

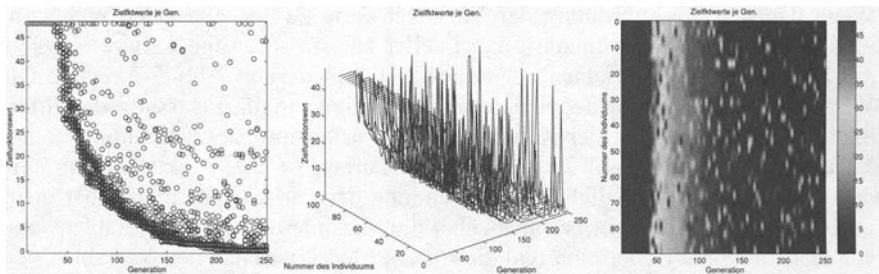
Die Darstellung der Zielfunktionswerte aller Individuen über mehrere Generationen gibt einen direkten Einblick in den Fortschritt der Population. Im Gegensatz zum Zielfunktionswert des besten Individuums bzw. des mittleren Zielfunktionswertes ergibt sich hier ein detaillierteres Bild. Es ist gut zu erkennen, ob es wenige gute Individuen gibt oder ob z.B. viele Individuen in bestimmten Wertebereichen liegen.

Für die grafische Darstellung bieten sich drei Möglichkeiten an:

- 2-D Punktdiagramm, jeder Punkt stellt einen Zielfunktionswert eines Individuums in einer Generation dar, Abb. 5-4 links,
- 3-D Liniendiagramm, wobei die Zielfunktionswerte (z-Achse) der Individuen (y-Achse) über die Generationen (x-Achse) aufgetragen werden, Abb. 5-4 Mitte oder
- 2-D Farbenteppich, in dem die Individuen über den Generationen aufgetragen werden und die jedem Individuum pro Generation zugeordnete Fläche die Farbe erhält, die mit dem entsprechenden Zielfunktionswert auf einer Farbskala korrespondiert, Abb. 5-4 rechts.

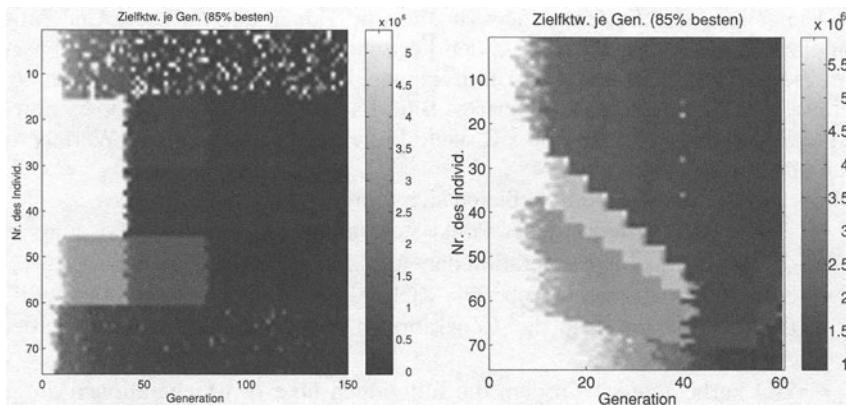
Die Darstellung im Punktdiagramm gibt einen guten Überblick zur Verteilung der Zielfunktionswerte, liefert aber keine Möglichkeit der Zuordnung eines Zielfunktionswertes zu einem Individuum. Im 3-D Diagramm ist eine genauere Zuordnung der Zielfunktionswerte zu den Individuen möglich. Allerdings ist das

3-D Diagramm bei einer großen Datenmenge zu unübersichtlich. Der Farbenteppich ist leichter als das 3-D Diagramm zu erfassen und bietet dieselben Informationen. Veränderungen der Zielfunktionswerte über die Generationen sind einfacher zu erkennen.



**Abb. 5-4.** Zielfunktionswerte aller Individuen einer Population über mehrere Generationen; links: 2-D Punktdiagramm, Mitte: 3-D Liniendiagramm, rechts: 2-D Farbenteppich

Informationen über die Position eines Individuums sind bei der Verwendung eines regionalen oder lokalen Modells von Bedeutung. Durch die räumliche Trennung der Individuen in Unterpopulationen oder Nachbarschaften kommt es zu einer Unterteilung der Population, die auch in der Darstellung der Zielfunktionswerte erkennbar ist.



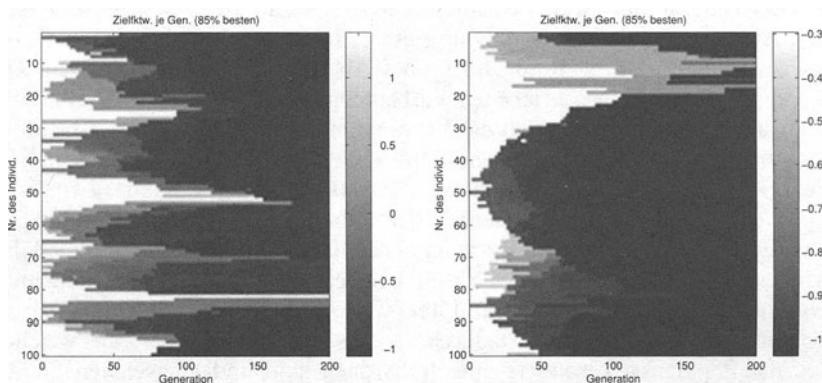
**Abb. 5-5.** Zielfunktionswerte aller Individuen einer Population über mehrere Generationen bei Verwendung des regionalen Modells – 2-D Farbenteppich; links: normales regionales Modell, rechts: Verwendung konkurrierender Unterpopulationen

Für eine eindimensionale Unterteilung der Population (regionales Modell sowie lokales Modell mit eindimensionaler Nachbarschaft) sind in den Abbildun-

gen 5-5 und 5-6 je zwei Beispiele gezeigt. Dabei kommt in allen Beispielen der 2-D Farbenteppich, s. Abb. 5-4 rechts, zur Anwendung. Für höherdimensionale Nachbarschaften des lokalen Modells sind die Methoden aus Abb. 5-4 nicht anwendbar.

Abbildung 5-5 zeigt Ergebnisse bei Anwendung des regionalen Modells. Deutlich sind die Grenzen zwischen den Unterpopulationen zu erkennen. Nach jeder Migration findet eine schnelle Ausbreitung der neuen Information in der Unterpopulation statt, was an den sprunghaften Änderungen der Zielfunktionswerte alle 40 Generationen erkennbar ist. Es findet ein „Heranholen“ der schlechteren Unterpopulationen an den Stand der besten Unterpopulation statt. In Abb. 5-5 rechts konkurrieren die Unterpopulationen miteinander. Sehr gut ist die sich verändernde Größe der Unterpopulationen in jeder vierten Generation (Konkurrenzintervall: 4 Generationen) zu erkennen. In dieser Darstellung ist direkt ablesbar, daß die erste Unterpopulation die größten Schritte bei der Verbesserung der Zielfunktionswerte macht und deshalb als beste Unterpopulation bei jedem Wettbewerb mehr Individuen erhält.

Bei der Anwendung des regionalen Modells findet eine direkte Unterteilung der Population statt, die sich bei der grafischen Darstellung der Zielfunktionswerte gut erkennen lässt. Wird dagegen das lokale Modell verwendet, ist die Unterteilung der Population eine fließende; je weiter Individuen voneinander entfernt sind, um so stärker sind sie voneinander getrennt. Außerdem werden Nachkommen in der Nähe ihrer Eltern in die Population eingefügt. Dadurch kommt es zu einer diffusen Ausbreitung guter Individuen innerhalb der Population.



**Abb. 5-6.** Zielfunktionswerte aller Individuen einer Population über mehrere Generationen in einer eindimensionalen lokalen Nachbarschaft; links: 2-D Farbenteppich mit kleiner Nachbarschaft, rechts: 2-D Farbenteppich mit größerer Nachbarschaft

In Abb. 5-6 sind Ergebnisse von zwei Läufen unter Verwendung des lokalen Modells dargestellt. Beide benutzen, bis auf die Größe der Nachbarschaft, dieselben Parameter. Die linke Grafik verwendet eine sehr kleine Nachbarschaft, in der rechten Grafik kam eine doppelt so große Nachbarschaft zur Anwendung. Es ist

deutlich zu erkennen, daß es bei einer größeren Nachbarschaft, Abb. 5-6 rechts, zu einer schnelleren Verbreitung guter Individuen kommt. Bei einer kleineren Nachbarschaft können, bedingt durch die stärkere Isolation, auch Individuen länger überleben, die nicht so gut sind, Abb. 5-6 links.

Für die Skalierung der Zielfunktionswerte gelten die am Ende von Unterabschn. 5.2.1 angegebenen Aussagen und Probleme. Folgendes Vorgehen kann einige Probleme umgehen: Für die Skalierung der Grafiken wird nur ein Teil der Zielfunktionswerte der Population (z.B. 85% der Population) verwendet, alle schlechteren Zielfunktionswerte werden auf den Maximalwert der verwendeten Zielfunktionswerte gesetzt. Bei den hier verwendeten Darstellungen wurde diese Begrenzung angewandt. In der Titelzeile wird jeweils angegeben, wieviele der Individuen zur Skalierung benutzt wurden.

Eine ähnliche Darstellung aller Gütwerte einer Population wird von ROUTEN und COLLINS in [RC93] vorgeschlagen. Die Autoren benutzen ein modifiziertes HINTON-Diagramm. In diesem Diagramm ist jedem Individuum ein rechteckiger Bereich zugeordnet. Wie groß dieses Rechteck ist, hängt von der Güte des Individuums ab. Schlechte Individuen werden durch kleine Rechtecke repräsentiert, die besten Individuen durch große Rechtecke, die den jedem Individuum zur Verfügung stehenden Platz einnehmen. Durch die gleichzeitige Darstellung mehrerer Generationen kann eine Verschiebung der Fitneß in der Population beobachtet werden. Je besser die Population wird, um so größer sind die bedeckten Bereiche der Individuen. Am Ende des Laufs wird fast das gesamte Diagramm ausgefüllt sein, die Rechtecke berühren sich untereinander. Zusätzlich können die Individuen nach ihrer individuellen Fitneß sortiert werden.

Die von ROUTEN und COLLINS vorgeschlagene Verwendung eines modifizierten Hinton-Diagramms hat den Nachteil, daß die Anzahl der zur Verfügung stehenden Abstufungen für die Darstellung der Fitneß eines Individuums stark begrenzt ist. Dadurch läßt sich nur ein grobes Abbild der Fitneßwerte geben. Die Verwendung des oben beschriebenen Farbenteppichs zeigt sich in der Anwendung als übersichtlicher, da durch die Verwendung von Farben feinere Abstufungen möglich sind. Die vorgeschlagene Sortierung der Population nach der individuellen Fitneß hat nur dann Vorteile, wenn es tatsächlich nur um eine Visualisierung der fortschreitenden Verbesserung der Fitneß in der Population geht. Der oben vorgeschlagene Farbenteppich kann neben der sich verändernden Fitneß der Individuen und der Population gleichzeitig noch ein Bild über die Verteilung und die Veränderung der Verteilung der Fitneßwerte innerhalb der Population bzw. der Unterpopulationen geben. Dadurch ist zusätzlich erkennbar, an welchen Stellen der Population besonders gute Individuen sind und an welchen Stellen bzw. Bereichen durch die Evolutionären Operatoren schlechte Individuen produziert werden (wichtig bzw. interessant bei der Beobachtung von Unterpopulationen mit verschiedenen Strategien, Abschn. 4.5, ab S.96, bzw. miteinander konkurrierenden Unterpopulationen, Abschn. 4.6, ab S.102).

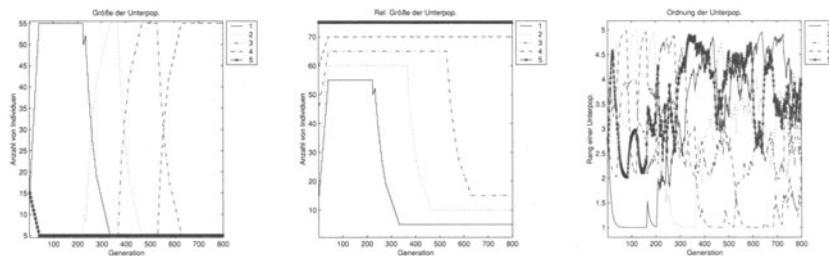
Einen Vorteil hat das modifizierte HINTON-Diagramm, wenn nur 2 Farben (z.B. schwarz und weiß) zur Visualisierung zur Verfügung stehen. Dann bietet die Größe des Rechtecks jedes Individuums mehr Abstufungen, als dies durch 2 Farben möglich wäre.

### 5.2.4 Position und Größe der Unterpopulationen

Die bisher in diesem Abschnitt vorgestellten Verfahren sind beim Einsatz Evolutionärer Algorithmen allgemein anwendbar. Im vorigen Unterabschnitt, bei der Darstellung der Zielfunktionswerte aller Individuen über mehrere Generationen, zeigte sich, daß bei Verwendung erweiterter Populationsmodelle, des regionalen und lokalen Modells, zusätzlich Informationen aus den Darstellungen abgelesen werden können.

Werden neben der Verwendung des regionalen Modells auch verschiedene Strategien je Unterpopulation verwendet, dann ist die Ordnung (Reihenfolge) der Unterpopulationen der Meßwert, an dem der Erfolg der Strategien ablesbar ist. Dies wurde ausführlich in Abschn. 4.5, ab S.96 erläutert. Beispiele für die Visualisierung der Ordnung der Unterpopulationen werden in Abb. 4-11, S.100 gegeben.

Findet zwischen den Unterpopulationen Konkurrenz statt, so ändert sich die Größe der Unterpopulationen. Diese veränderliche Größe der Unterpopulationen läßt sich im Verlauf der Generationen darstellen und zeigt direkt die Ressourcen, die jeder Unterpopulation zur Verfügung stehen und indirekt, wie groß der Erfolg der Unterpopulationen bzw. Strategien ist.



**Abb. 5-7.** Größe der Unterpopulationen, 2-D Liniengrafik; links: direkte Darstellung der Größe jeder Unterpopulation, Mitte: kumulative Größe der Unterpopulationen, rechts: Ordnung der Unterpopulationen

In Abb. 5-7 sind zwei Varianten des Verlaufs der Größe der Unterpopulationen dargestellt. Die erste Variante, Abb. 5-7 links, zeigt die Größe der Unterpopulationen. Bei der zweiten Variante, Abb. 5-7 Mitte, wird die kumulative (aufaddierte) Größe der Unterpopulationen dargestellt. Die zweite Variante hat den Vorteil, daß direkt die relative Verteilung der Ressourcen abgelesen werden kann. Zusätzlich kann aus dieser Grafik auch die Größe einer bestimmten Unterpopulation (als Differenz zwischen zwei Linien) entnommen werden, ohne daß eine Legende vorhanden ist. Dies ist besonders bei einer großen Anzahl von Unterpopulationen von Vorteil. Die rechte Grafik zeigt die Ordnung der Unterpopulationen aus demselben Optimierungslauf.

## 5.3 Lokale Eigenschaften einer Population

Die in diesem Abschnitt beschriebenen Methoden dienen einer schnellen und übersichtlichen Darstellung des Zustandes einer Population. Dazu werden Daten dargestellt, die aus einer Generation stammen. Dies entspricht meist den Eigenschaften der Individuen einer Generation bzw. der aktuellen Population.

### 5.3.1 Variablen aller Individuen einer Generation

Die Variablen aller Individuen einer Population stehen immer zur Verfügung und können einfach und direkt dargestellt werden. Die Darstellung aller Variablen der Individuen einer Population gibt ein momentanes Abbild des Phänotyps der Individuen der Population.

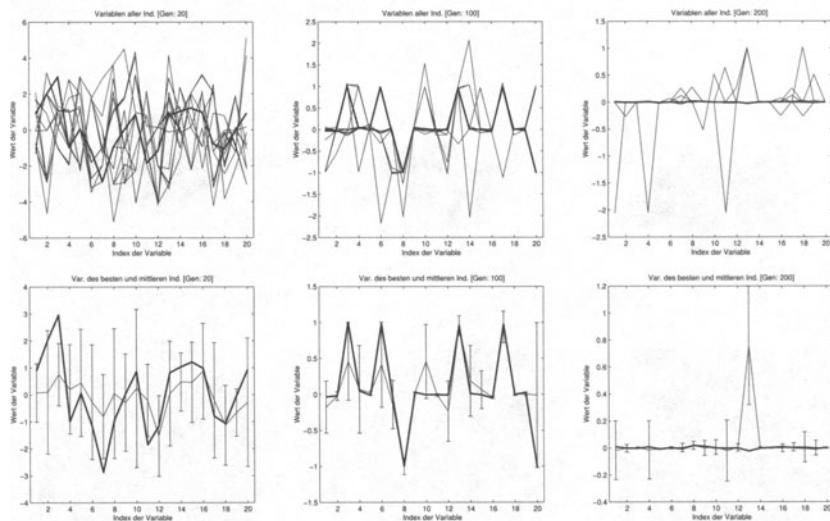
Für die grafische Darstellung bieten sich vier Möglichkeiten an:

- 2-D Liniendiagramm, direkte Darstellung; jedes Individuum wird durch eine Linie repräsentiert, die den Wert der Variablen über der Nummer der Variablen darstellt, s. Abb. 5-8 oben,
- 2-D Liniendiagramm, indirekte Darstellung; Darstellung des besten und durchschnittlichen Individuums zusammen mit der Standardabweichung der Variablen vom Mittelwert der Variablen, s. Abb. 5-8 unten,
- 3-D Liniendiagramm; jedes Individuum wird durch eine Linie repräsentiert, die den Wert der Variablen über der Nummer der Variablen darstellt, s. Abb. 5-9 oben, oder
- 2-D Farbenteppich; die Individuen werden über den Generationen aufgetragen und die jedem Individuum pro Generation zugeordnete Fläche erhält die Farbe, die mit dem entsprechenden Variablenwert auf einer Farbskala korrespondiert, s. Abb. 5-9 unten.

Für eine reelle oder diskrete Repräsentation der Variablen ist die erste Darstellungsmethode gut geeignet und liefert Ergebnisse, die leicht zu interpretieren sind. Die zweite Möglichkeit ergibt besonders bei einer hohen Anzahl von Individuen eine übersichtlichere Darstellung. Die dritte und vierte Variante bieten die Möglichkeit der Zuordnung der Variablenwerte zu den einzelnen Individuen, was insbesondere bei der Anwendung des regionalen und lokalen Modells von Vorteil ist. Außerdem geben sie ein besseres Bild, wieviele Individuen ähnliche Variablenwerte haben.

In Abb. 5-8 sind die ersten beiden Varianten dargestellt. Zur Veranschaulichung der Aussagekraft einer solchen Grafik sind jeweils Grafiken von Beginn, Mitte und Ende eines Laufs dargestellt. Zu Beginn des Laufs (jeweils linke Grafik) ist die Vielfalt in der Population hoch. Die Variablen der Individuen unterscheiden sich stark voneinander, es gibt kaum identische Individuen. Entsprechend hoch ist die Standardabweichung für jede der Variablen (linke untere Grafik). Im weiteren Verlauf (jeweils mittlere Grafik) nimmt die Vielfalt in der Population deutlich ab, viele Individuen sind zu anderen Individuen sehr ähnlich. Die Standardabweichung ist deutlich kleiner als zu Beginn des Laufs. Am Ende der Optimierung (jeweils rechte Grafik) sind fast alle Individuen sehr ähnlich.

Dies zeigt sich auch an der sehr kleinen Standardabweichung für alle Variablen. Eine Ausnahme bildet zu diesem Zeitpunkt noch Variable 13: Erst wenige Individuen haben für diese Variable einen Wert von null, die meisten Individuen arbeiten noch mit einem Wert von eins.

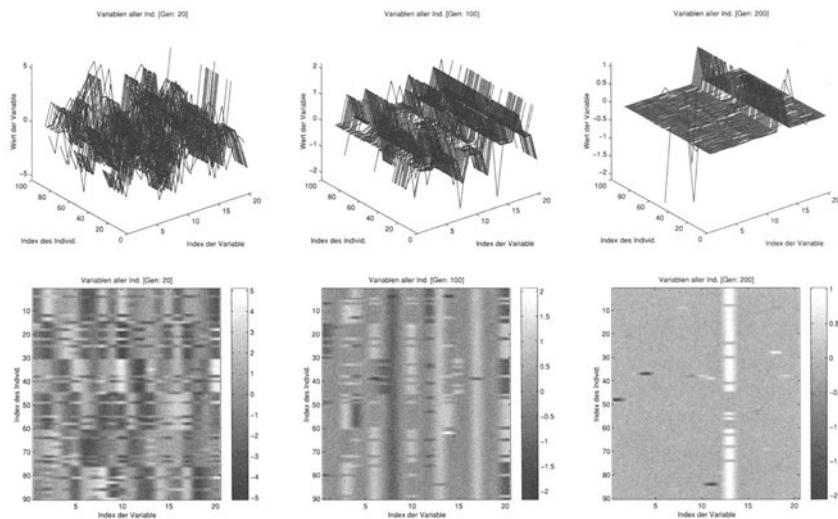


**Abb. 5-8.** Variablen aller Individuen einer Population; oben: 2-D Liniengrafik aller Individuen, unten: 2-D Liniengrafik des besten (dicke Linie) und des durchschnittlichen Individuums sowie der Standardabweichung der Variablen; links: Beginn eines Laufs, Mitte: während eines Laufs, rechts: Ende eines Laufs

Die Verwendung eines 3-D Liniendiagramms ermöglicht eine Zuordnung der einzelnen Variablenwerte zu den Individuen, s. Abb. 5-9 oben. Diese zusätzliche Information erhöht auf der einen Seite die aus der Grafik zu entnehmende Information, gleichzeitig verringert sich dadurch aber die Übersichtlichkeit deutlich. Gerade bei vielen verschiedenen Variablenwerten bzw. unterschiedlichen Individuen ist das 3-D Liniendiagramm nur schwer zu überblicken. Nur bei der Verwendung weniger Individuen oder weniger Variablen pro Individuum kann das 3-D Liniendiagramm seine Vorteile zur Geltung bringen. In Abb. 5-9 oben ist gut zu sehen, wie erst ab der Mitte des Laufs (mittlere Grafik) etwas aus der Darstellung zu entnehmen ist. Am Ende (rechte Grafik), wenn alle Individuen sehr ähnlich sind, lässt sich aus dem 3-D Liniendiagramm gut erkennen, daß einige Individuen schon alle Variablen um 0 haben, während die anderen Individuen an Position 13 noch eine 1 stehen haben.

Den Vorteil des 3-D Linendiagramms, die Zuordnung der Variablenwerte zu den Individuen, kombiniert mit einer guten Übersichtlichkeit, bietet die Darstellung der Variablenwerte in einem 2-D Farbenteppich, s. Abb. 5-9 unten. Auf einen Blick ist zu erkennen, wie die Variablenwerte an den einzelnen Positionen und in der Population verteilt sind, ob bestimmte Variablenpositionen bei allen

Individuen gleich sind, an welchen Positionen große Unterschiede auftreten und welches Individuum an welcher Position welchen Wert hat. In Abb. 5-9 unten, ist gut zu sehen, wie sich vom Beginn des Laufs (linke Grafik), in dem die Variablen noch sehr unterschiedlich sind, im weiteren Verlauf die Unterschiede verringern (mittlere Grafik). Am Ende (rechte Grafik) sind die Individuen in allen Variablen bis auf eine gleich.



**Abb. 5-9.** Variablen aller Individuen einer Population; oben: 3-D Liniendiagramm, unten: 2-D Farbenteppich; links: Beginn eines Laufs, Mitte: während eines Laufs, rechts: Ende eines Laufs (die dargestellten Daten korrespondieren mit Abb. 5-8)

Die Darstellung mit einem 2-D Farbenteppich ist für alle Repräsentationen der Variablen geeignet (reell, ganzzahlig, diskret, binär). Je nach Mannigfaltigkeit der möglichen Werte jeder Variablen müssen bestimmten Wertebereichen gleiche Farben zugeordnet oder die Anzahl der verwendeten Farben beschränkt werden. So würden z.B. bei einer binären Repräsentation nur 2 Farben verwendet werden.

Zusammenfassend können aus diesen Grafiken die folgenden Dinge herausgelesen werden:

- Die Verteilung bzw. Angleichung der Variablenwerte kann eingeschätzt werden. Zu Beginn eines evolutionären Laufs sind die Variablen bei einer zufälligen Initialisierung gleichmäßig in ihrem möglichen Wertebereich verteilt. Durch die Arbeit des Evolutionären Algorithmus schränkt sich die Verteilung immer weiter ein und konzentriert sich auf einen oder mehrere Bereiche, wobei die Variablen der einzelnen Individuen sich immer weiter angleichen. Damit ergibt sich ein visuelles Maß für die Verschiedenartigkeit der Individuen (*diversity*), die am Beginn hoch ist und während eines Laufs kleiner wird.

- Die fortschreitende Beschränkung der Variablen auf wenige bzw. kleine Wertebereiche, die als vielversprechend erkannt wurden, kann abgelesen werden. In diesen Bereichen liegt oftmals das spätere Ergebnis des Laufs.

Wenn sich mehrere Bereiche herausbilden, kann dies ein Indikator dafür sein, daß die Zielfunktion multimodal ist. Oder umgekehrt, bei einer multimodalen Funktion werden sich während des Laufs verschiedene Bereiche (*cluster*) herausbilden. Durch die sich ständig verringernde Verschiedenartigkeit bleibt am Ende meist nur ein Bereich übrig (Ausnahme: Verwendung der mehrkriteriellen (*multiobjective*) Fitneßzuweisung, s. Unterabschn. 3.1.3, ab S.20).

Es erfordert etwas Übung bzw. Vertrautheit mit dem Problem, damit aus dieser Darstellung der Variablen aller Individuen die oben angegebenen Informationen herausgelesen werden können. Weiterhin sind die möglichen Ergebnisse stark von der Repräsentation der Variablen abhängig. Trotzdem ist dieses Diagramm eines der aufschlußreichsten für den aktuellen Zustand der Population und sollte bei keiner Visualisierung fehlen.

Bei der Bearbeitung realer Systeme liegen die Variablen oft in sehr unterschiedlichen Definitionsbereichen. In der Darstellung kann dieses Problem umgangen werden, indem man die Variablen mit diesem Definitionsbereich skaliert und nur der relative Wert der Variablen im Definitionsbereich visualisiert wird.

Die Visualisierung der Variablen aller Individuen wird in [RC93] in zwei Varianten erwähnt. Die erste entspricht dem hier vorgestellten Farbenteppich, die zweite der 2-D Liniengrafik. Für die 2-D Liniengrafik, als ‘Überlagerung der Repräsentation eines jeden Chromosoms’ bezeichnet, wird der Hinweis gegeben, daß damit die Positionen erkannt werden können, an denen die Diversität innerhalb der Population nicht mehr vorhanden ist.

### 5.3.2 Zielfunktionswerte aller Individuen einer Generation

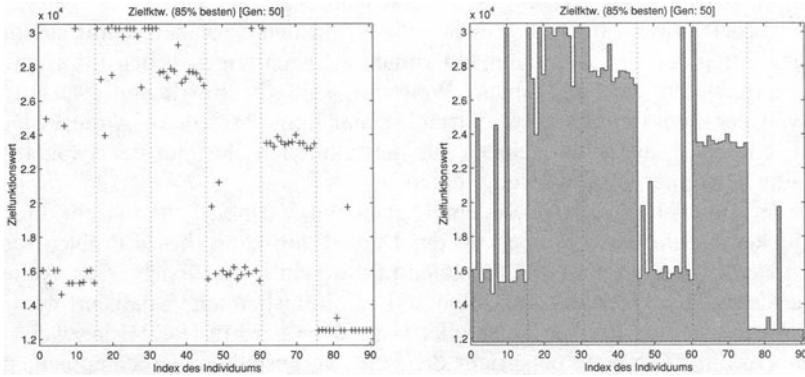
Die Zielfunktionswerte der Individuen einer Population stehen direkt zur Verfügung und bieten sich somit zu einer Visualisierung an. Die Darstellung aller Zielfunktionswerte ergibt ein weiteres detailliertes Abbild des Zustandes der Population.

Für eine Darstellung werden die Zielfunktionswerte über den einzelnen Individuen in einem 2-D Diagramm aufgetragen. Hierfür bieten sich Punkt- oder ausgefüllte Treppendiagramme an. Abbildung 5-10 zeigt die 2 Möglichkeiten für dieselbe Population von Individuen.

In Abb. 5-10 erfolgt die Anordnung der Individuen entsprechend ihrer Position in der Population, wobei nur eine eindimensionale Struktur betrachtet wird. Eine weitere Möglichkeit besteht darin, die Individuen der Population nach ihrer Fitneß zu sortieren und dann die Zielfunktionswerte darzustellen. Diese Darstellung gibt ein direktes Bild der Verteilung der absoluten Werte der Zielfunktionswerte. Auf einen Blick ist zu erkennen, ob viele Individuen gleichwertige Zielfunktionswerte aufweisen, ob nur wenig gute oder nur wenig schlechte Individuen in der Population vertreten sind.

Wesentlich interessanter wird die Darstellung der Zielfunktionswerte aller Individuen einer Population bei der Verwendung von Unterpopulationen. Dabei

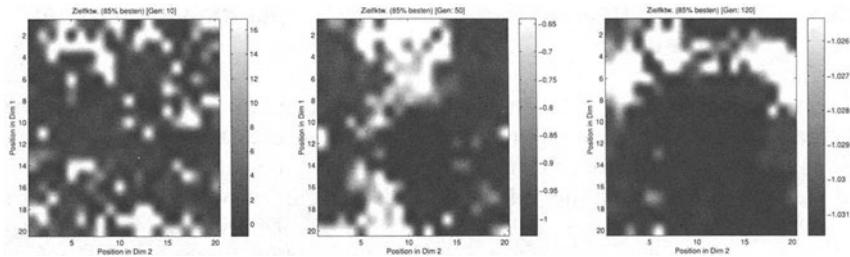
darf die Population nicht entsprechend der Fitneß sortiert sein, sondern muß in der Reihenfolge dargestellt sein, wie die Individuen in den Unterpopulationen angeordnet sind. Außerdem müssen im Diagramm die Grenzen zwischen den Unterpopulationen eingezeichnet werden. In Abb. 5-10 sind die Grenzen zwischen den Unterpopulationen als senkrechte gestrichelte Linien eingezeichnet. Damit kann auf einfache Weise das unterschiedliche Verhalten der Unterpopulationen beobachtet werden.



**Abb. 5-10.** Zielfunktionswerte aller Individuen einer Generation; links: 2-D Punktdiagramm; rechts: ausgefülltes 2-D Treppendiagramm

Noch wichtiger wird diese Darstellung bei der Verwendung unterschiedlicher Strategien für die einzelnen Unterpopulationen. In jeder Generation ist die Unterpopulation zu erkennen, deren Strategie im Moment den besten Fortschritt erbringt. Genauso ist zu erkennen, wenn eine Unterpopulation keinerlei gute Individuen produziert. Werden miteinander konkurrierende Unterpopulationen verwendet, so ist durch die eingezeichneten Grenzen zwischen den Unterpopulationen gleichzeitig die Größe jeder Unterpopulation dem Diagramm zu entnehmen.

Die bisher gemachten Aussagen gelten ganz allgemein für die Darstellung der Zielfunktionswerte. Für den Fall einer ein- oder zweidimensionalen Nachbarschaft unter den Individuen (Verwendung des lokalen Modells, Abschn. 4.3, ab S.77) läßt sich noch mehr Information aus dieser Darstellung gewinnen. Für eine eindimensionale Nachbarschaft können dieselben Diagramme wie in Abb. 5-10 verwendet werden. Für eine zweidimensionale Nachbarschaft bietet sich der 2-D Farbenteppich (Farbe entspricht dem Zielfunktionswert des entsprechenden Individuums), s. Abb. 5-11, und das 3-D Oberflächendiagramm an. Da die Zielfunktionswerte entsprechend der räumlichen Anordnung der Individuen dargestellt sind (x-Achse: Position in Dimension 1, y-Achse: Position in Dimension 2), lassen sich räumliche Bereiche ähnlicher Zielfunktionswerte gut erkennen. Diese Darstellungen geben einen Einblick in die räumliche Verteilung der Zielfunktionswerte.



**Abb. 5-11.** Zielfunktionswerte aller Individuen einer Generation, Anordnung der Individuen in einem zweidimensionalen Gitter – lokales Modell, 2-D Farbenteppich; links: Beginn eines Laufs, Mitte: während eines Laufs, rechts: Ende eines Laufs

In Abb. 5-11 lässt sich gut die Bildung von Bereichen verfolgen. Zu Beginn des Laufs (linke Grafik) sind die Zielfunktionswerte der Individuen sehr unterschiedlich. Im Verlauf der nächsten Generationen werden die Unterschiede zwischen benachbarten Zielfunktionswerten immer kleiner, es kommt zur Ausbildung kleiner Bereiche von ähnlichen Zielfunktionswerten. Gegen Ende des Laufs werden die Bereiche guter (niedriger) Zielfunktionswerte immer größer und verdrängen dadurch die schlechten Zielfunktionswerte. In der rechten Grafik dominiert ein großer Bereich guter Zielfunktionswerte deutlich die Population.

Allerdings lässt sich aus der Verteilung der Zielfunktionswerte nicht eindeutig erkennen, ob verschiedene Bereiche der Population mit gleichen Zielfunktionswerten gleiche oder unterschiedliche Bereiche des Suchraumes durchmustern. An dieser Stelle sei darauf hingewiesen, daß zwei Individuen mit demselben Zielfunktionswert nicht unbedingt auch dieselben Variablenwerte (denselben Phänotyp) besitzen müssen. Die Unterschiede zwischen den Individuen lassen sich aus der Darstellung der Zielfunktionswerte nur bei einigen Problemen ablesen. Besser geeignet zur Erkennung der Unterschiede zwischen den Individuen ist die Darstellung der Variablenwerte selbst, Unterabschn. 5.3.1, bzw. die Darstellung der Distanz zwischen Individuen in Distanzkarten, Unterabschn. 5.3.3.

SCHWEHM beschreibt und verwendet in [Swm96] die Darstellung der Zielfunktionswerte zur Visualisierung der räumlichen Verteilung der Qualität der Individuen und nennt diese Darstellung Qualitätskarten. ROUTEN und COLLINS beschreiben in [RC93] zwei Varianten zur Darstellung der Zielfunktionswerte einer Population einer Generation: 2-D Balkendiagramm (sortiert nach Nummer des Individuums, Fitneß der Individuen oder Distanz zum besten Individuum) und ‘radial fitness plot’<sup>1</sup>.

<sup>1</sup> *radial fitness plot:* Jedes Individuum wird durch ein Kreissegment in einem ‘radial plot’ repräsentiert. Der Abstand jedes Kreissegmentes von der Mitte des Diagramms ist proportional der Güte des Individuums. Wenn die Individuen entsprechend ihrer Güte sortiert sind, ergibt sich zu Beginn eines Laufs eine einfache Schneckenlinie. Zum Ende eines Laufs, wenn alle Individuen eine ähnliche Güte haben, wird die Schneckenlinie zu einem Kreis.

### 5.3.3 Distanzverteilung und Distanzkarten der Individuen einer Generation

Die Individuen einer Population arbeiten während eines Laufs alle im selben Suchraum, jedoch meist in unterschiedlichen Bereichen. Dies wird auch als parallele Durchmusterung des Suchraums bezeichnet. Eine wichtige Eigenschaft ist die Distanz zwischen den Individuen. Die Distanz gibt an, wie weit die Individuen während des Laufs voneinander entfernt sind. Die Distanz ist eine abgeleitete Größe, sie steht nicht auf Grund der Arbeit des Evolutionären Algorithmus zur Verfügung.

Die Nutzung der Distanz zwischen den Individuen kann in zwei Anwendungsbereiche unterteilt werden bzw. kann Aussagen über zwei verschiedene Eigenschaften machen:

- Die Distanz zwischen den Individuen ist ein Ausdruck für die Verschiedenartigkeit der Individuen (*diversity*). Je ähnlicher die Individuen sind, um so geringer ist die Diversität der Population.
- Wenn sich während eines Laufs Nachbarschaften oder Cluster bilden, so kann dies durch eine Visualisierung der Distanz zwischen den Individuen erkannt werden.<sup>2</sup>

Je nachdem, welcher der beschriebenen Anwendungsfälle betrachtet werden soll, kommen zwei verschiedene Darstellungsformen zum Einsatz:

- Distanzverteilung und
- Distanzkarten.

Die Berechnung der Distanz ist abhängig von der Repräsentation der Variablen der Individuen. Für die binäre Repräsentation kann die Hammingdistanz verwendet werden, für die ganzzahlige oder reelle Repräsentation der euklidische Abstand.

#### Distanzverteilung

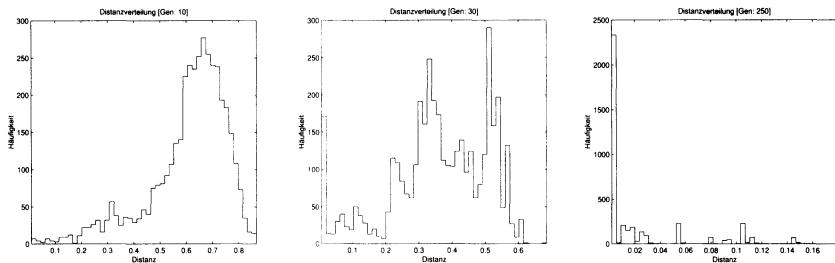
Die Distanzverteilung stellt die Verteilung der Abstände zwischen den Individuen einer Population dar. Zuerst werden die Distanzen zwischen allen Individuen einer Population berechnet.<sup>3</sup> Anschließend wird ermittelt, wie häufig Distanzen in bestimmte Distanzintervalle fallen (Ermittlung der Verteilungsdichte der Distanzen). Diese Verteilungsdichte wird in einem Histogramm dargestellt, s. Abb. 5-12.<sup>4</sup>

---

<sup>2</sup> Dies zeigt sich im besonderen bei einer räumlich beschränkten Selektion (lokales Modell, Abschn. 4.3, S.77 und regionales Modell, Abschn. 4.4, S.87). Bei der Anwendung dieser Verfahren kommt es zur Ausbildung von semiisolierten Bereichen auf Grund einer ‘Isolation durch Distanz’.

<sup>3</sup> Bei großen Populationen kann auch eine repräsentative Auswahl verwendet werden. Dabei kann es zu leichten Verfälschungen in den Ergebnissen kommen, die aber nur selten signifikant sind.

<sup>4</sup> Die Verteilungsdichte kann statt mittels eines Histogramms auch durch eine ‘kernel density estimation’ ermittelt werden [CB96].



**Abb. 5-12.** Distanzverteilung der Individuen einer Generation: 2-D Treppendiagramm; links: Beginn eines Laufs, Mitte: während eines Laufs, rechts: Ende eines Laufs

Das Distanzverteilungsdiagramm gibt ein Bild der Diversität der Population. Zu Beginn eines Laufs sind die Individuen im Suchraum verteilt. Dadurch existieren viele verschiedene Distanzen zwischen den einzelnen Individuen (große Abstände zwischen den Individuen, viele verschiedene Abstände). Je weiter der Lauf fortschreitet, um so stärker nähern sich die Individuen einander an und die Abstände zwischen den Individuen werden immer kleiner. Wenn sich alle Individuen einem Punkt annähern, werden die Abstände fast auf Null absinken. In Abb. 5-12 ist dieser Fall dargestellt. Es ist gut zu erkennen, wie die absoluten Abstände während des Laufs kleiner werden und die Abstände immer weniger verteilt sind.

Außerdem kann aus der Distanzverteilung die Bildung von Clustern erkannt werden. Im Diagramm zeigt sich dies durch die Herausbildung von mehreren sogenannten Bergen. Wenn die Individuen größtenteils zu zwei verschiedenen Clustern gehören, dann sind die Abstände der Individuen in den Clustern untereinander nahe Null, die Abstände zwischen Individuen unterschiedlicher Cluster aber deutlich größer, jedoch in der Gesamtheit auch wieder sehr ähnlich. Im Diagramm zeigt sich dies durch zwei Bereiche, ein Bereich nahe Null, der andere im Bereich des Abstandes der Zentren der beiden Cluster.

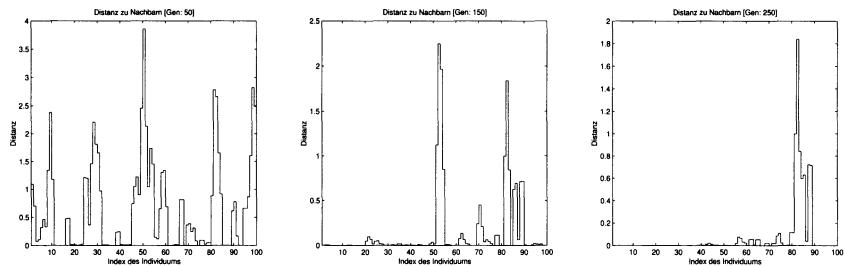
In Abb. 5-12 Mitte ist ein Beispiel mit mehreren Bereichen zu erkennen. Ein großer Teil der Abstände liegt um 0,15. Weiterhin treten viele Abstände mit Werten um Null auf. Eine noch kleinere Anzahl von Abständen hat Werte um 0,05. Am Ende sind die verschiedenen Bereiche verschwunden, alle Individuen liegen sehr dicht beieinander.

Diese Analyse kann auch für mehr als zwei oder drei Cluster durchgeführt werden. Allerdings können sich dann die einzelnen Bereiche so stark überdecken, daß eine Clusterung in dem Sinne nicht mehr erkennbar ist. Hier zeigen sich die Grenzen des Distanzverteilungsdiagramms. Eine Lösung für den Fall des Auftretens mehrerer Cluster bieten die Distanzkarten.

Die Visualisierung der Distanzverteilung wurde von ROUTEN in [Rou94] vorgeschlagen.

## Distanzkarten

Bei den Distanzkarten geht es weniger um die Darstellung der Häufigkeit, mit der Distanzen zwischen allen Individuen auftreten, sondern vor allem um die Darstellung der Distanz zwischen Individuen in ihrer räumlichen Anordnung. Zum Einsatz kommen Distanzkarten überall dort, wo eine räumlich beschränkte Interaktion zwischen den Individuen einer Population stattfindet (lokales Modell, s. Abschn. 4.3, ab S.77).



**Abb. 5-13.** Distanzkarte einer Population in eindimensionaler Nachbarschaft als 2-D Treppendiagramm; links: Beginn eines Laufs, Mitte: während eines Laufs, rechts: Ende eines Laufs

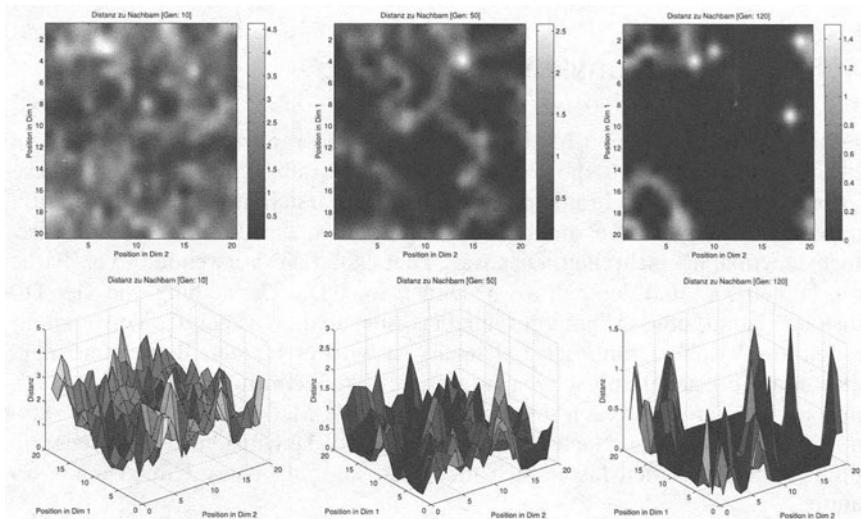
Für die Visualisierung wird die Distanz eines Individuums zu seinen Nachbarn berechnet und anschließend in dieser Nachbarschaft dargestellt. Je nach Struktur der Nachbarschaft kommen unterschiedliche Diagramme zum Einsatz. Für eine eindimensionale (lineare) Nachbarschaft können die Distanzen in einem einfachen 2-D Treppendiagramm dargestellt werden, s. Abb. 5-13.

Bei einer zweidimensionalen Nachbarschaft bietet sich zum einen die Verwendung eines Farbenteppichs an, bei dem das jedem Individuum zugeordnete Rechteck eine Farbe entsprechend der Distanz zu seinen Nachbarn erhält. Eine weitere Möglichkeit ist die Verwendung eines 3-D Oberflächendiagramms, bei dem die Distanz direkt in die Höhe der 3. Koordinate umgesetzt wird. Beide Varianten sind in Abb. 5-14 dargestellt.

Distanzkarten zeigen Eigenschaften von Nachbarschaften innerhalb der Population. Ähnlich wie bei der Distanzverteilung, lässt sich die Diversität der Population erkennen. Zu Beginn ist die Distanz zwischen benachbarten Individuen hoch. Durch die lokalen Nachbarschaften (lokale evolutionäre Operationen) bilden sich recht schnell räumliche Bereiche mit ähnlichen Individuen. Dies wird durch große Bereiche mit geringen Distanzen zwischen den Individuen deutlich. Die Anzahl von Individuen mit großen Distanzen wird gegen Ende des Laufs immer kleiner.

Die Bildung von unterschiedlichen Clustern als zweite zu erkennende Eigenschaft lässt sich aus den Distanzkarten wesentlich leichter ablesen, als dies aus der Distanzverteilung möglich ist. Da die Distanzen entsprechend der räumlichen Verteilung der Individuen dargestellt werden, können Bereiche, in denen die Distanzen zwischen den Individuen sehr klein sind, leicht als Bereiche mit ähnlichen Individuen erkannt werden. An der Grenze von Bereichen gibt es einen

Sprung in der Distanz eines Individuums zu seinen Nachbarn. Die Grenzen von Clustern sind damit an den Individuen mit einer hohen Distanz zu erkennen. Deutlich ist dies in Abb. 5-13 rechts zu erkennen. Bei einer zweidimensionalen Nachbarschaft zeigt sich dies in einer anderen Farbe im Farbenteppich oder durch große Sprünge im 3-D Oberflächendiagramm, s. Abb. 5-14.



**Abb. 5-14.** Distanzkarte einer Population in zweidimensionaler Nachbarschaft; oben: 2-D Farbenteppich, unten: 3-D Oberflächendiagramm; links: Beginn eines Laufs, Mitte: während eines Laufs, rechts: Ende eines Laufs (die dargestellten Daten korrespondieren mit Abb. 5-11)

Wenn zwischen den Individuen eine eindimensionale Nachbarschaft besteht, so kann mit einer Distanzkarte nicht nur die Entwicklung und Veränderung der Distanzen zwischen den benachbarten Individuen in einer Generation dargestellt, sondern über alle Generationen beobachtet werden. Sehr anschaulich dafür ist die Verwendung eines Farbenteppichs. Jede Zeile repräsentiert die Individuen einer Generation, wobei die Farbe jedes Individuums wiederum dessen Distanz zu seinen Nachbarn symbolisiert. Durch die geschlossene farbige Darstellung lassen sich leicht die Bereiche der Population erkennen, in denen sich gleichartige Individuen befinden bzw. es lässt sich erkennen, wie sich die Grenzen zwischen Clustern von ähnlichen Individuen im Verlaufe der Generationen verändern. Allerdings sollte nicht verschwiegen werden, daß für diese Darstellung die Variablen aller Individuen aller Generationen gespeichert werden müssen. Dies führt für viele Probleme realistischer Größe heute zu einem unverhältnismäßig hohen Aufwand bzw. bringt die Programmsysteme an den Rand ihrer Leistungsfähigkeit. Aber in einigen Jahren oder mit speziell auf die Verwendung des lokalen Modells zugeschnittenen Programmen ist die Verwendung dieser Darstellung sehr zu empfehlen.

Distanzkarten zur Visualisierung der Eigenschaften von Nachbarschaften werden u.a. von SCHWEHM in [Swm96] beschrieben und verwendet. SCHWEHM beschränkt sich aber auf die Darstellung der Hammingdistanz zwischen den Individuen. Wie die obigen Beispiele zeigen, können bei einer Verwendung anderer Abstandsmaße (z.B. euklidischer Abstand) Distanzkarten für alle Repräsentationen der Variablen verwendet werden.

## 5.4 Hochdimensionale Visualisierung

Alle bisher angesprochenen Methoden zur Visualisierung beschränken sich auf die Darstellung von Daten, die von ein oder zwei Variablen abhängig sind. Diese Grenze ist durch die Beschränkung der normalen Darstellung auf drei Dimensionen gegeben. Es gibt zwei mögliche Erweiterungen, um diese Grenze der drei Dimensionen zu überschreiten. Dies wäre zum ersten die Verwendung von Farbe als 4. Dimension<sup>5</sup> und der Zeit als 5. Dimension<sup>6</sup>. Die Darstellung von vier Dimensionen ist mit etwas Übung gut nutzbar. Aber spätestens bei der Verwendung der Zeit zur Visualisierung der 5. Dimension wird es für den Betrachter ungewohnt und die Darstellung schwer interpretierbar. Wenn nur vier oder fünf Dimensionen dargestellt werden müssen, sollten beide Methoden in Betracht gezogen werden. Doch ein Nachteil bleibt: bei fünf Dimensionen ist wiederum Schluß. Beide Varianten lassen sich nicht beliebig auf höhere Dimensionen erweitern.

Zur Visualisierung hochdimensionaler Daten muß ein Weg gefunden werden, wie diese Daten auf weniger Dimensionen herunter transformiert oder umgerechnet werden können. Es muß versucht werden, die Datenpunkte aus einer hohen Dimension so in einer niedrigen Dimension abzubilden, daß beide Darstellungen ähnlich zueinander sind. Als Ähnlichkeitsmaß bietet sich die Distanz zwischen den Datenpunkten in der hohen und der niedrigen Dimension an.

### 5.4.1 Verfahren

Die Aufgabe besteht darin, eine Abbildung zu finden, bei der die Abstände zwischen den Datenpunkten im hochdimensionalen Raum mit denen im niedrigdimensionalen Raum korrespondieren. Die Nahordnung der Datenpunkte soll erhalten bleiben. In der Literatur zur Mustererkennung und Klassifikation (u.a. [DH73], S.243-246 und [Rip96], S.305-311) werden solche Methoden als mehrdimensionale Skalierung (*multidimensional scaling*) bezeichnet.

---

<sup>5</sup> In einem 3-D Diagramm, in dem jeder Achse eine der drei Variablen zugeordnet ist, erhält jeder Punkt oder Bereich eine Farbe entsprechend dem Funktionswert an dieser Position. Dies funktioniert bei der Visualisierung einer Punktwolke oder einer Fläche, aber nicht, wenn der gesamte Raum ausgefüllt ist. Dann würden die Punkte und Bereiche sich gegenseitig verdecken.

<sup>6</sup> In Erweiterung zum oben beschriebenen 3-D Diagramm wird eine Abfolge solcher Diagramme erstellt. Die Zeit repräsentiert hierbei die 4. Variable.

Die Distanzen zwischen verschiedenen Datenpunkten können echte Abstände<sup>7</sup> sein oder einen Ersatz für den Abstand darstellen, wenn direkte Werte nicht berechenbar sind. Beide Arten von Abständen werden meist unter dem Begriff Ungleichheit (*dissimilarities*) zusammengefaßt. Diese Ungleichheiten müssen die Dreiecksungleichung<sup>8</sup> nicht zwingend erfüllen. Im folgenden soll nur auf echte Abstände eingegangen werden, für die als Abstandsmaß der euklidische Abstand verwendet wird. In der Literatur (z.B. [Rip96] und [CC94]) finden sich weitere Abstandsmaße, die bei Daten zur Anwendung kommen, bei denen es nur auf die Ordnung der Abstände untereinander ankommt und nicht auf den absoluten Wert des Abstandes.

Die folgenden Erläuterungen der mehrdimensionalen Skalierung orientieren sich an [DH73], ab S.243. Als Bezeichnungen werden verwendet:

$$\begin{array}{ll} x_1, \dots, x_n: & \text{Datenpunkte in } \Re^p \\ \delta_{ij}: & \text{Abstand zwischen } x_i \text{ und } x_j \\ y_1, \dots, y_n: & \text{Bildpunkte in } \Re^q, \quad p \geq q \\ d_{ij}: & \text{Abstand zwischen } y_i \text{ und } y_j \end{array} \quad (5-1)$$

Es müssen die Bildpunkte  $y_1, \dots, y_n$  gefunden werden, für welche die Abstände  $d_{ij}$  so nah wie möglich an den entsprechenden Abständen  $\delta_{ij}$  der originalen Datenpunkte  $x_1, \dots, x_n$  sind. Nur selten ist es möglich, Bildpunkte zu finden, so daß  $d_{ij} = \delta_{ij}$  für alle  $i$  und  $j$  gilt. Deshalb muß ein Kriterium definiert werden, das angibt, wie gut eine Konfiguration von Bildpunkten im Vergleich zu anderen Konfigurationen ist. Die Summe der Fehlerquadrate ist Grundlage der am häufigsten benutzten Kriterien.

In [DH73] werden drei Varianten der Funktionen für die Bewertung des Fehlers bei der mehrdimensionalen Skalierung (Summe der Fehlerquadrate) vorgeschlagen (linke Spalte: Kriterien; rechte Spalte: 1. Ableitung der entsprechenden Funktion):

$$J_{ee} = \frac{1}{\sum \delta_{ij}^2} \sum_{i < j} (\delta_{ij} - d_{ij})^2 \quad \nabla_{y_k} J_{ee} = \frac{2}{\sum \delta_{ij}^2} \sum_{i < j} \left( (\delta_{kj} - \delta_{ij}) \frac{y_k - y_j}{d_{kj}} \right) \quad (5-2)$$

$$J_{ff} = \sum_{i < j} \left( \frac{d_{ij} - \delta_{ij}}{\delta_{ij}} \right)^2 \quad \nabla_{y_k} J_{ff} = 2 \sum_{j \neq k} \left( \frac{d_{kj} - \delta_{kj}}{\delta_{kj}^2} \cdot \frac{y_k - y_j}{d_{kj}} \right) \quad (5-3)$$

$$J_{ef} = \frac{1}{\sum \delta_{ij}^2} \sum_{i < j} \frac{(\delta_{ij} - d_{ij})^2}{\delta_{ij}} \quad \nabla_{y_k} J_{ef} = \frac{2}{\sum \delta_{ij}^2} \sum_{i < j} \left( \frac{d_{kj} - \delta_{kj}}{\delta_{kj}} \cdot \frac{y_k - y_j}{d_{kj}} \right) \quad (5-4)$$

Kriterium  $J_{ee}$  in Gl. 5-2 betont die größeren Fehler, egal ob die Abstände  $\delta_{ij}$  groß oder klein sind. Kriterium  $J_{ff}$  in Gl. 5-3 hebt die größeren relativen Fehler hervor, egal ob der Betrag des Fehlers groß oder klein ist. Kriterium  $J_{ef}$  in Gl. 5-4

<sup>7</sup> Zum Beispiel der euklidische Abstand zwischen den Datenpunkten.

<sup>8</sup> Dreiecksungleichung ('triangle inequality') mit  $d$ : Abstand zwischen zwei Punkten:

$$d_{ik} \leq d_{ij} + d_{jk} .$$

bildet einen vernünftigen Kompromiß zwischen den beiden ersten Kriterien, indem es das größte Produkt aus Fehler und relativem Fehler betont.

Da sich die Ableitungen der Kriterien einfach berechnen lassen, s. rechte Spalte der Gl. 5-2, 5-3 und 5-4, kann eines der vielen bekannten Gradientenverfahren zur Ermittlung einer optimalen Konfiguration von Bildpunkten verwendet werden.

Eine der bekanntesten Methoden der mehrdimensionalen Skalierung ist das Sammon-Mapping [Sam69]. SAMMON verwendete das Kriterium der Gl. 5-4 und ein Verfahren des steilsten Abstiegs (diagonales Newtonverfahren). In [Rip96] wurde berichtet, daß das von SAMMON benutzte Optimierungsverfahren nicht immer konvergiert. Oftmals mußte eine wesentlich kleinere Schrittweite verwendet werden, als von SAMMON angegeben, damit das Verfahren nicht divergiert.

Als Optimierungsverfahren, das den Gradienten verwendet und robust ist, kann das *Rprop*-Verfahren [RB93] eingesetzt werden. *Rprop* verwendet für die Generierung eines neuen Punktes nur die Richtung des Gradienten und nicht den Betrag. Die Schrittweitensteuerung von *Rprop* wird vom Wechsel des Vorzeichens des Gradienten gesteuert. *Rprop* hat sich im Bereich Neuronaler Netze in den letzten Jahren bewährt und wird dort oft eingesetzt. In MATLAB sind außerdem eine Reihe robuster gradientenorientierter Optimierungsverfahren enthalten, mit denen gute Erfahrungen beim Sammon-Mapping gemacht wurden (z.B. *BFGS Quasi-Newton method with a mixed quadratic and cubic line search procedure*).

Die Ausgangskonfiguration von Bildpunkten für die mehrdimensionale Skalierung kann zufällig erstellt werden. Eine deutliche Verbesserung der Suche nach einer optimalen Konfiguration sowie eine Beschleunigung der Optimierung können durch eine sinnvolle Auswahl der Anfangskonfiguration erreicht werden. Die *principal component analysis* (PCA) ist eine dafür geeignete Methode, die durch vorhandene Programmierungswerkzeuge oft bereitgestellt wird.

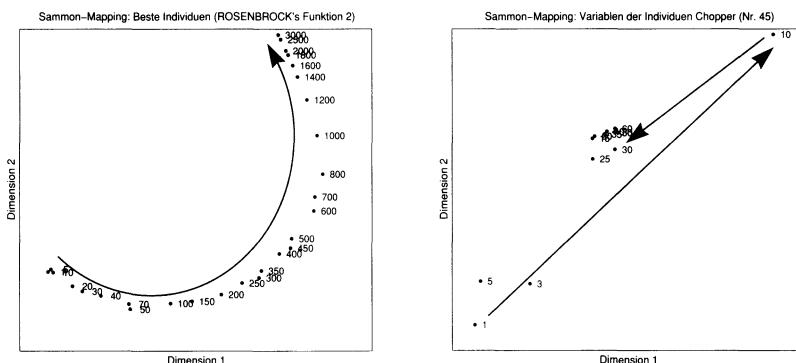
### 5.4.2 Anwendung der mehrdimensionalen Visualisierung

Wofür läßt sich die mehrdimensionale Skalierung im Bereich der Visualisierung Evolutionärer Algorithmen einsetzen? Die Anwendung ist überall dort interessant, wo viele hochdimensionale Datenpunkte in ihrer Beziehung zueinander dargestellt werden sollen. Die folgenden Beispiele geben eine Auswahl:

- Darstellung mehrerer Individuen:
  - Individuen einer, mehrerer oder aller Generationen,
  - beste Individuen mehrerer oder aller Generationen („Pfad des besten Individuums jeder Generation durch den Suchraum“),
- Darstellung mehrkriterieller Zielfunktionswerte:
  - visuelles Ranking der Zielfunktionswerte untereinander („Pfad des besten Individuums jeder Generation durch den Lösungsraum“).
- Vergleich mehrerer Optimierungsläufe:
  - Variablen der besten Individuen („Pfad durch den Suchraum“),
  - Zielfunktionswerte der besten Individuen („Pfad durch den Lösungsraum“).

Mit der Darstellung der Individuen der Population erhält man ein Bild, wie die Individuen im Suchraum verteilt sind bzw. wie der verwendete Algorithmus den Suchraum durchmustert. Damit ist auch zu erkennen, ob die Individuen alle in einem Bereich liegen, oder ob sie weiter voneinander entfernt sind.

Die Darstellung mehrkriterieller Zielfunktionswerte ermöglicht deren optische Bewertung. Bis zu drei Dimensionen geht dies noch direkt mit den Methoden, die von FONSECA in verschiedenen Arbeiten (u.a. [Fon95] und [FF96]) vorgestellt wurden. Bei vier und mehr Kriterien bietet sich die mehrdimensionale Skalierung an.

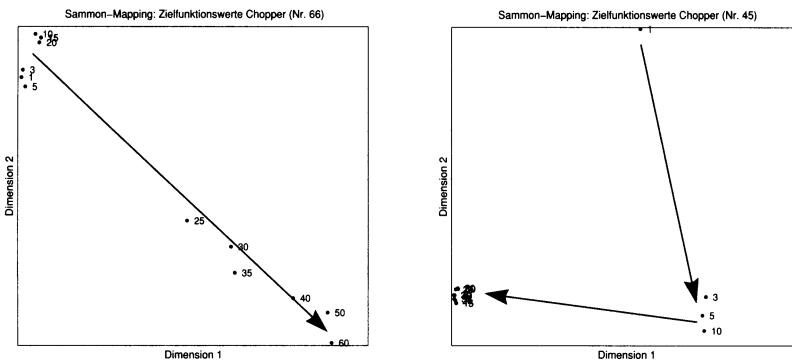


**Abb. 5-15.** Mehrdimensionale Skalierung der besten Individuen eines Laufs; links: ROSENROCK's Funktion 2, rechts: Gleichstrom-Steller Nr. 45

In Abb. 5-15 sind zwei Beispiele der mehrdimensionalen Skalierung für die besten Individuen eines Laufs gegeben. In beiden Beispielen ist jeder der Datenpunkte mit der Nummer der Generation beschriftet, aus der das entsprechende Individuum stammt.

Der Weg des besten Individuums eines Laufs für ROSENROCK's Funktion in zehn Dimensionen (zehn Variablen) ist in Abb. 5-15 links dargestellt. Die zweidimensionale Darstellung ergibt eine stark gekrümmte Linie, ähnlich einer Banane (ROSENROCK's Funktion wird oft als Bananefunktion bezeichnet). Die Darstellung zeigt damit dieselbe Charakteristik für diese Funktion, die schon aus der direkten zweidimensionalen Darstellung bekannt ist, s. Abb. 5-18.

Abbildung 5-15 rechts zeigt den Weg des besten Individuums für eine Optimierung des Gleichstrom-Stellers (s. [Poh98], Abschn. 7.2). Die Individuen haben jeweils neun Variablen. Die zweidimensionale Darstellung unterscheidet sich deutlich von der für ROSENROCK's Funktion. Es ergibt sich keine Linie durch den Suchraum, dem die Individuen während des Laufs folgen. Vielmehr lässt sich erkennen, daß in den ersten Generationen die Individuen in einem Bereich liegen. Das Individuum von Generation 10 ist dagegen weit von den Individuen der vorherigen Generationen entfernt. Die Individuen aller nachfolgenden Generationen finden sich zwischen diesen bisherigen Extremwerten in einem relativ kleinen Gebiet.



**Abb. 5-16.** Mehrdimensionale Skalierung mehrkriterieller Zielfunktionswerte; links: Gleichstrom-Steller Nr. 66, rechts: Gleichstrom-Steller Nr. 45

Abbildung 5-16 zeigt zwei Beispiele der mehrdimensionalen Skalierung für mehrkriterielle Zielfunktionswerte der besten Individuen eines Laufs. Genau wie in Abb. 5-15 ist in beiden Beispielen jeder der Datenpunkte mit der Nummer der Generation beschriftet, aus der das entsprechende Individuum stammt. Jeder Zielfunktionswert besteht aus sechs Kriterien.

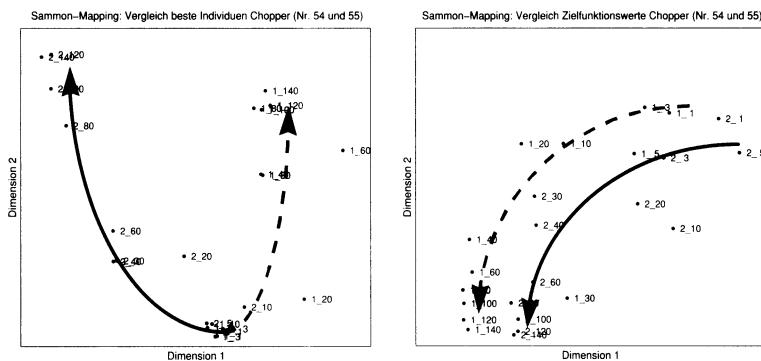
In Abb. 5-16 links ist zu erkennen, wie es nach anfänglichen langsamem Veränderungen zwischen Generation 20 und 25 zu einem Sprung kommt. Danach findet eine kontinuierliche Verbesserung statt, die in etwa einer Linie folgt.

Deutlich anders ist die Darstellung in Abb. 5-16 rechts, die mit Abb. 5-15 rechts korrespondiert. Die Zielfunktionswerte von Generation 1 liegen weit entfernt von denen der Generationen 3 bis 10. Alle folgenden Generationen liegen wiederum weit entfernt von den bisherigen Werten. Ähnlich zu Abb. 5-15 rechts, liegen diese Werte wieder in einem sehr kleinen Bereich. Im Vergleich der mehrdimensionalen Skalierung für Individuen und Zielfunktionswerte für dieses Beispiel zeigen sich einerseits weite Übereinstimmungen, andererseits auch Unterschiede. Ähnlich ist die Clusterung zum Ende des Laufs. Dagegen zeigen sich Unterschiede zu Beginn des Laufs. Die Individuen von Generation 1, 3 und 5 liegen dicht beieinander und 10 weit entfernt, Abb. 5-15 rechts. Bei den Zielfunktionswerten ist der Wert von Generation 1 weit von denen der Generationen 3, 5 und 10 entfernt, Abb. 5-16 rechts. Durch diese Darstellung der mehrdimensionalen Skalierung können damit weitere Informationen zu einem Problem erarbeitet werden, die sonst kaum zugänglich sind.

Die Analyse von Daten durch die mehrdimensionale Skalierung kann auf den Vergleich der Daten von mehreren Läufen ausgedehnt werden. In Abb. 5-17 ist dafür ein Beispiel gegeben. Die Grafiken zeigen gleichzeitig die Individuen bzw. Zielfunktionswerte von zwei Läufen. Jeder Datenpunkt ist mit der Nummer des Laufs (1\_ oder 2\_) und der jeweiligen Generation beschriftet.

Die Individuen beider Läufe liegen für die ersten Generationen im selben kleinen Bereich, s. Abb. 5-17 links. Ab etwa Generation 20 entfernen sich die Individuen vom Startbereich. Dabei folgen die Individuen in den beiden Läufen verschiedenen Wegen. Am Ende der Läufe sind die Individuen weit voneinander

entfernt. Bei den Zielfunktionswerten ist das Bild etwas anders. In den ersten Generationen sind die Datenpunkte in einem weiten Bereich verteilt. Dann bewegen sich die Datenpunkte in denselben Bereich hinein, wobei sie aber unterschiedlichen und voneinander getrennten Linien folgen. Besonders bei den Zielfunktionswerten des ersten Laufs ist dies gut zu erkennen.



**Abb. 5-17.** Vergleich der besten Individuen (links) und der mehrkriteriellen Zielfunktionswerte (rechts) zweier Läufe (Gleichstrom-Steller Nr. 54 und 55) durch mehrdimensionale Skalierung

Durch diese Art des Vergleichs verschiedener Läufe können detaillierte Informationen über den „Weg“ der Individuen durch den Suchraum bzw. die Relation der mehrkriteriellen Zielfunktionswerte zueinander erarbeitet werden. Damit eröffnet die mehrdimensionale Skalierung einen neuen Bereich der Visualisierung von Daten, der mit den bisher vorgestellten Verfahren so nicht möglich war.

### 5.4.3 Analyse der mehrdimensionalen Visualisierung

Erste Arbeiten zur Anwendung des SAMMON-Mapping auf dem Gebiet der Evolutionären Algorithmen werden in [DCW96] und [Col97a] berichtet.

In [DCW96] wird SAMMON-Mapping direkt verwendet. In der Arbeit wird allerdings nur auf die Verwendung von Variablen in einer binären Repräsentation eingegangen. Weiterhin wird in der Arbeit davon ausgegangen, daß es möglich ist, zu Beginn einer Optimierung ein Mapping für alle möglichen Individuen aufzustellen und die entsprechenden Positionen dieser Individuen im niedrigdimensionalen Raum in einer Tabelle abzulegen. Dadurch muß während eines Laufs nur eine Referenz in einer Tabelle gesucht und keine Optimierung der Abstände durchgeführt werden.

Von COLLINS wurde in [Col97a] das ‘*Genotype-Space Mapping*’ präsentiert. Dies ermöglicht die analytische Konstruktion eines Mapping aus einem hochdimensionalen in einen niedrigdimensionalen Raum, ähnlich dem SAMMON-Mapping. An einigen Beispielen für binäre Variablen geringer Länge wurden Vergleiche zum SAMMON-Mapping durchgeführt. Im Ergebnis war das

‘*Genotype-Space Mapping*’ im Fehler zwar nicht ganz so gut wie ein konvertiertes SAMMON-Mapping, dafür lässt sich das ‘*Genotype-Space Mapping*’, zumindest für Individuen mit wenigen Variablen, direkt und schnell berechnen. COLLINS demonstrierte außerdem, wie sich für Variablen einer höheren Basis ein ‘*Genotype-Space Mapping*’ aufstellen lässt.

Für Probleme mit wenigen binären Variablen sind beide Methoden sehr gut geeignet und während eines Laufs sehr schnell. Speziell die Art der Ergebnisdarstellung in [DCW96] und die Anzeige der nächsten Nachbarn sind für die praktische Anwendung gut geeignet. Die Skalierbarkeit beider Methoden ist dagegen schlecht. Schon für Probleme mit 30 und mehr Variablen ist die vorhergehende Berechnung aller Mapping-Positionen und deren Speicherung nicht mehr praktisch anwendbar. Bei einer reellen Repräsentation der Variablen wird es noch schwieriger, da vorher eine Diskretisierung der Domain der Variablen durchgeführt werden muß und dadurch die Dimension des Problems weiter ansteigt.

Für die Darstellung der Positionen von Individuen im Suchraum im Verlauf einer Optimierung stellt sich ein Problem: wird in jeder Generation ein selbständiges SAMMON-Mapping durchgeführt, ist keine Vergleichbarkeit der Abbildungen zwischen den Generationen gegeben. Selbst wenn sich die Position nur eines Individuums ändert, kann die Darstellung des resultierenden SAMMON-Mapping deutlich anders sein.

Für die Lösung bzw. Umgehung dieses Problems sind, je nach beabsichtigter Anwendung, zwei Varianten denkbar. Bei einer direkten Beobachtung der Optimierung wird jeweils das Ergebnis des vorherigen SAMMON-Mapping als Startpunkt verwendet und die Individuen der aktuellen Generation werden als zusätzliche Punkte in die Optimierung einbezogen. Dargestellt werden dann allerdings nur die Individuen der aktuellen Population. Bei einer nachträglichen Betrachtung der Optimierungsergebnisse kann ein SAMMON-Mapping über alle Individuen aller darzustellenden Generationen durchgeführt werden. Bei der Darstellung der Daten einer oder mehrerer Generationen werden dann immer nur die Daten der jeweiligen Generationen gezeigt. Dies kann zwar einen sehr hohen Rechenaufwand für das SAMMON-Mapping zur Folge haben, dürfte aber, besonders bei der Anwendung auf reale Probleme, immer noch deutlich geringer sein, als ein SAMMON-Mapping für alle möglichen Individuen. Es wird wirklich nur für diejenigen Individuen ein Mapping durchgeführt, die auch dargestellt werden sollen.

## 5.5 Eigenschaften der Zielfunktion

Die in den vorangegangenen Abschnitten vorgestellten Verfahren dienten der Darstellung von Daten, die während eines Laufs vorliegen und stehen in direktem Zusammenhang mit der Arbeit des Evolutionären Algorithmus.

In diesem Abschnitt werden Verfahren vorgestellt, die zur direkten Visualisierung von Eigenschaften der Zielfunktion dienen. Damit sind diese Verfahren nicht nur auf die Arbeit mit Evolutionären Algorithmen anwendbar, sondern können auch im Zusammenhang mit anderen Optimierungsverfahren eingesetzt werden. Weiterhin können diese Verfahren zum tiefergehenden Kennenlernen der Zielfunktion verwendet werden.

### 5.5.1 Direkte Darstellung der Zielfunktion

Bei der Arbeit mit einer Zielfunktion steht die Frage, welche Eigenschaften diese Zielfunktion hat, welche Besonderheiten auftreten, wie das Gütegebirge aussieht. Eine der einfachsten Methoden zum Erlangen von zusätzlichem Wissen ist die direkte Darstellung der Zielfunktion.

Für die direkte Darstellung stehen nur drei Dimensionen zur Verfügung, so daß nur Zielfunktionen mit 2 Variablen (Dimensionen) „vollständig“ dargestellt werden können. Da die meisten Probleme hochdimensional sind, ist eine „vollständige“ grafische Darstellung des Gütegebirges nicht möglich. Über die Erstellung von ein- bzw. zweidimensionalen Schnitten durch die Zielfunktion (Variationsdiagramme) können trotzdem Teilauspekte des Aussehens der Zielfunktion veranschaulicht werden. Damit kann z.B. abgeschätzt werden, ob und wie stark die Funktion verrauscht ist, ob Unstetigkeiten enthalten sind bzw. ob lokale Minima zu erwarten sind.

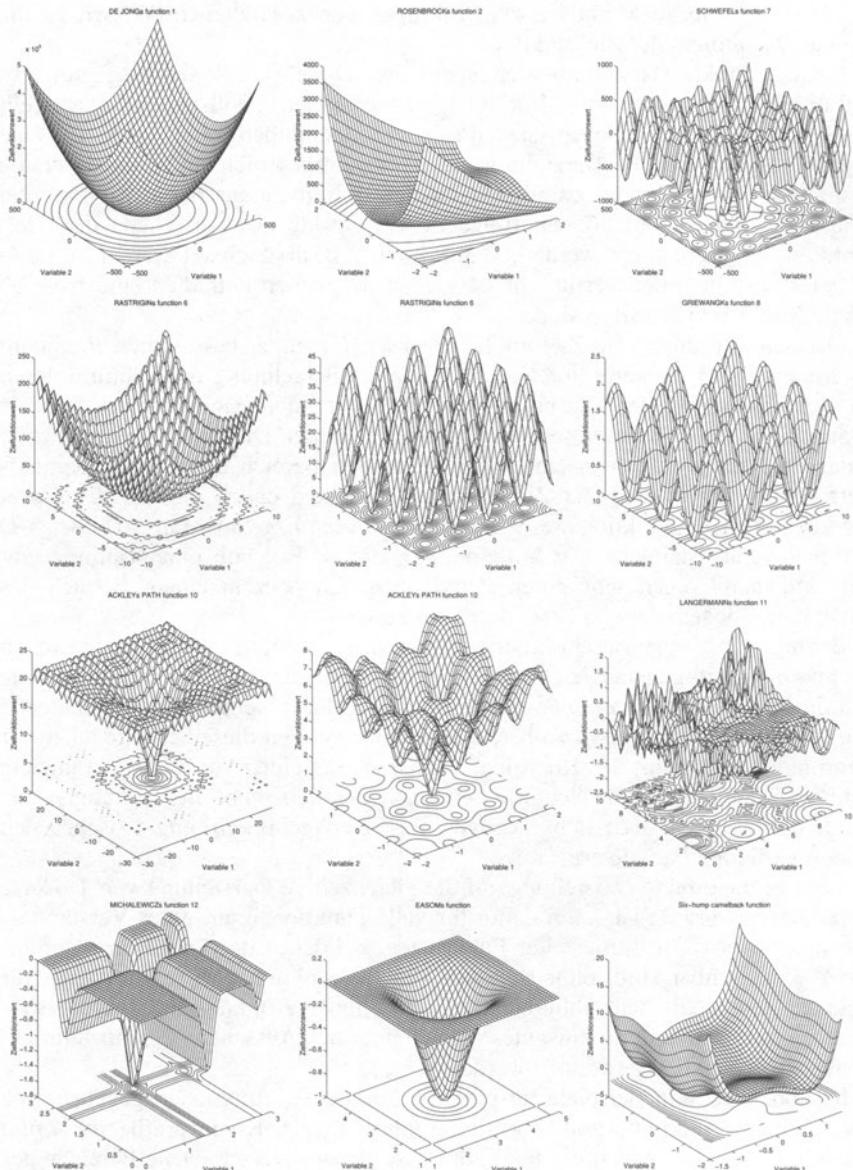
Die Schnitte durch die Zielfunktion werden für einen bestimmten Punkt im Suchraum als Mittelpunkt durchgeführt. Für die Berechnung der Schnitte durch die Zielfunktion werden alle bis auf die zu variierten Parameter des Systems konstant auf den Werten dieses Mittelpunktes gehalten. Die zu variierten bzw. freien Parameter werden in einem festzulegenden Bereich um den Mittelpunktwert verändert (abtasten der Zielfunktion mit einem engen Gitter), die korrespondierenden Zielfunktionswerte berechnet und die Ergebnisse in 2-D oder 3-D Grafiken veranschaulicht. Der Mittelpunkt kann das Ergebnis einer Optimierung sein und damit einen sehr guten Punkt darstellen oder in einem Bereich des Suchraumes liegen, der von besonderem Interesse ist.

Bedingt durch die Beschränkung der Visualisierung auf drei Dimensionen können nur eindimensionale (Variation eines Parameters) und zweidimensionale Schnitte (Variation von zwei Parametern) durchgeführt werden. Eindimensionale Schnitte sind relativ schnell zu berechnen, dafür können diese Schnitte auch nur Informationen entlang der Koordinatenachse dieses einen variierten Parameters liefern. Zweidimensionale Schnitte benötigen deutlich mehr Berechnungen, ermöglichen dagegen zusätzlich Aussagen über die Wechselwirkung zwischen den beiden variierten Parametern.

Obwohl die direkte Darstellung auf die gleichzeitige Darstellung von 1 oder 2 Variablen beschränkt ist, kann damit für viele Funktionen ein gutes Verständnis der Struktur erreicht werden. Für Funktionen, bei denen die Variablen untereinander vertauschbar sind, ohne daß sich die Zielfunktion ändert (dies trifft für viele der gebräuchlichen skalierbaren Testfunktionen zu), kann durch eine zweidimensionale Darstellung ein gutes Verständnis des Aussehens der hochdimensionalen Zielfunktion gewonnen werden.

In Abb. 5-18 sind Beispiele für einige bekannte Testfunktionen gegeben. Die Darstellungen sind Gittergrafiken mit darunterliegender Konturgrafik (*mesh plot* mit *contour plot*). Ein Teil der Funktionen ist für zwei verschiedene Bereiche der Variablen dargestellt, wodurch die unterschiedliche Charakteristik dieser Funktionen in Abhängigkeit des zugrundeliegenden Definitionsbereiches der Variablen deutlich sichtbar wird. Unter Nutzung dieser Darstellungen ist ein gutes Verständnis des Aussehens und der Eigenschaften dieser Funktionen möglich.

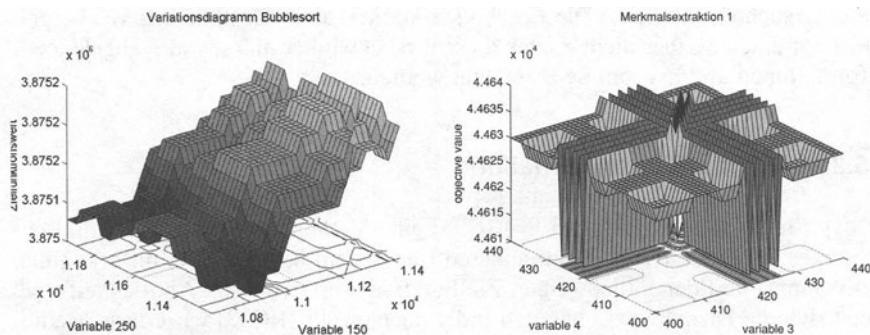
Weitere Beispiele für die direkte Darstellung von Testfunktionen sind in Abschn. 8.2, ab S.194 gegeben.



**Abb. 5-18.** Direkte Darstellung der Zielfunktion für zwei Variablen (ausgewählte Standardtestfunktionen)

Bei realen Problemen sind die Variablen nicht untereinander austauschbar und jede Variable hat ihre eigene Bedeutung. Damit müssen mehrere Variationsplots für die Variablen bzw. Paare von Variablen erstellt werden. Zwei Beispiele sollen im folgenden gezeigt werden. In beiden Beispielen werden Variationsdiagramme für ausgewählte Variablen gezeigt und die abgeleiteten Informationen genannt.

In Abb. 5-19 sind Variationsdiagramme für zwei Softwaretestsysteme (Bestimmung der maximalen Ausführungszeit von Softwaremodulen [PW99], hier Module Bubblesort und Merkmalsextraktion, ganzzahlige Repräsentation der insgesamt 500 bzw. 843 Variablen) gezeigt.



**Abb. 5-19.** Zweidimensionale Schnitte durch das Gütegebirge von Softwaretestsystemen , links: Variation der Variablen 150 und 250 um 0,6% des Definitionsbereiches (System Bubblesort), rechts: Variation der Variablen 3 und 4 um 1% des Definitionsbereiches (System Merkmalsextraktion)

Das linke Diagramm in Abb. 5-19 zeigt eine Variation für die Variablen 150 und 250 einer zufällig generierten Lösung des Systems Bubblesort. Die Darstellung wurde auf 0,6% des Definitionsbereiches der Variablen beschränkt. Dadurch wurde wirklich jeder mögliche Funktionswert in der Grafik berechnet, es fand keine Interpolation zwischen benachbarten Punkten statt. Hier zeigen sich kleine lokale Minima in der Darstellung. Die Zerknüllung des Variationsdiagramms zeigt sehr deutlich, daß dieses System sehr knifflig bzw. unlösbar für viele deterministische Optimierungsverfahren ist. Es dürfen keine lokal orientierten Parameteroptimierungsverfahren verwendet werden, da diese in den lokalen Minima hängenbleiben.

Im rechten Diagramm in Abb. 5-19 ist ein Variationsdiagramm des Systems Merkmalsextraktion zu sehen. Von den insgesamt 843 Variablen wurden hier die Variablen 3 und 4 variiert. In diesem Variationsdiagramm ist so ziemlich alles enthalten, was ein Optimierungsproblem schwierig macht: lokale Minima, Plateaus, starke Sprünge in den Zielfunktionswerten bei kleinen Änderungen der Variablen und Korrelationen zwischen den Variablen. Auf der anderen Seite scheint eine gewisse Struktur erkennbar zu sein, die bei näherer Untersuchung zu einer Einschränkung der Problemgröße führen könnte. Zumindest ist eindeutig, daß diese Optimierungsaufgabe einige Probleme bereiten wird (und bereitet hat).

Eine weitere Anwendung der direkten Darstellung der Zielfunktion wird in Unterabschn. 8.3.2, ab S.207, im Rahmen der Optimierung eines Motormodells präsentiert.

Wie bei den hier gezeigten Beispielen von Testfunktionen und realen Systemen zu sehen ist, können aus der direkten Darstellung der Zielfunktion mittels Variationsdiagrammen Informationen über die Zielfunktion erhalten werden. Es muß allerdings beachtet werden, daß alle aus diesen Diagrammen abgeleiteten Informationen nur für den dargestellten Bereich gelten und auch dort Einschränkungen unterliegen. Auf Grund der oftmals nichtlinearen Zusammenhänge zwischen den Variablen kann es für andere Bereiche zu deutlich anderen Aussagen kommen. Deshalb liefern die abgeleiteten Informationen nur Anhaltspunkte für die untersuchten Bereiche. Die Ergebnisse können auch für andere Bereiche gelten, sie müssen es aber nicht. Unter diesem Blickwinkel müssen alle abgeleiteten Informationen und Ergebnisse betrachtet werden.

### 5.5.2 Fitneß-Distanz-Verteilung

ROUTEN und COLLINS schlagen in [RC93] eine Veranschaulichung der Ähnlichkeit eines Individuums zu einem anderen (meist dem besten bis dahin bekannten Individuum) und der Fitneß (Güte, Zielfunktionswert) vor. Als Ähnlichkeitsmaß bietet sich die Distanz zwischen den Individuen an. In [RC93] wird diese Methode zur Visualisierung des augenblicklichen Zustandes einer Population verwendet.

Wenn statt der aktuellen Population eine größere Anzahl von Individuen verwendet wird, ergibt sich ein Abbild der Ähnlichkeit zwischen allen verwendeten Individuen. Dadurch kann ein abstrahiertes Abbild der Zielfunktion gewonnen werden.

JONES und FORREST stellen in [JF95] eine identische Methode vor, wobei ihr Anliegen vordergründig in eine andere Richtung zielt. Sie wollen eine Maßzahl finden, die einen Ausdruck für die Schwierigkeiten bildet, die eine Zielfunktion einem Genetischen Algorithmus bei der Optimierung bereitet. Frühere Vorschläge für ein solches Maß (*deceptive problems*, u.a. [Gol87]; *rugged fitness landscapes*, u.a. [HG95]; *epistatic interactions*, [Dav91b]) „funktionieren“ zwar am Beispiel einiger Funktionen, aber es lassen sich immer wieder Gegenbeispiele finden. Damit wird deutlich, daß diese Vorschläge manchmal etwas mit der Schwere einer Funktion zu tun haben, aber nicht die entscheidende Antwort liefern können.

In [JF95] wird eine Bewertung des Zusammenhangs zwischen der Fitneß (gemeint ist die Güte bzw. der Zielfunktionswert) einer Lösung (eines Individuums) und seinem Abstand (Distanz) zur besten Lösung (globales Optimum) als Maßstab für die Schwere einer Funktion untersucht. Als Maß wird die Fitneß-Distanz-Korrelation (*fitness distance correlation – FDC*) definiert.

Eine Möglichkeit der Berechnung besteht darin, eine Anzahl von Individuen zu nehmen, von diesen den Zielfunktionswert (hier als Fitneß *Fit* bezeichnet) zu berechnen und zusammen mit der Distanz zum besten Individuum *Dist* daraus nach Gl. 5-5 den FDC-Koeffizienten zu bilden.

$$FDC = \frac{\text{cov}_{FitDist}}{\text{std}_{Fit} \cdot \text{std}_{Dist}}$$

$$\text{cov}_{FitDist} = \frac{1}{n} \sum_{i=1}^n (Fit_i - \text{mean}_{Fit}) \cdot (Dist_i - \text{mean}_{Dist}) \quad (5-5)$$

cov: Kovarianz    std: Standardabweichung    mean: Mittelwert

Die Berechnung der Distanz ist abhängig von der Repräsentation des Problems. Für eine binäre Repräsentation kann die Hammingdistanz verwendet werden, für ganzzahlige und reelle Repräsentationen der euklidische Abstand als leicht zugängliches Maß. Für andere Repräsentationen ist die Definition des Abstandes etwas komplizierter bzw. problemspezifisch.

Für die Stellen, an denen  $FDC$  als einzelnes Maß zu einfach ist, um den Zusammenhang zwischen Fitneß und Distanz vollständig wiederzugeben, kann die visuelle Bewertung der Fitneß-Distanz-Verteilung in einem 2-D Punktdiagramm (*scatter plot*) vorgenommen werden.

Es bleibt die Frage, ob die Anwendung der Fitneß-Distanz-Verteilung hilfreiche Aussagen liefert. Bei den Untersuchungen in [JF95] wird mit einfachen Testfunktionen niedriger Dimension sowie einer binären Repräsentation gearbeitet. Für diese einfachen Beispiele kann die Fitneß-Distanz-Korrelation bzw. die Darstellung der Fitneß-Distanz-Verteilung einen Hinweis auf die Schwere der Funktion liefern. Die Anwendung auf andere Funktionen scheint vielversprechend.

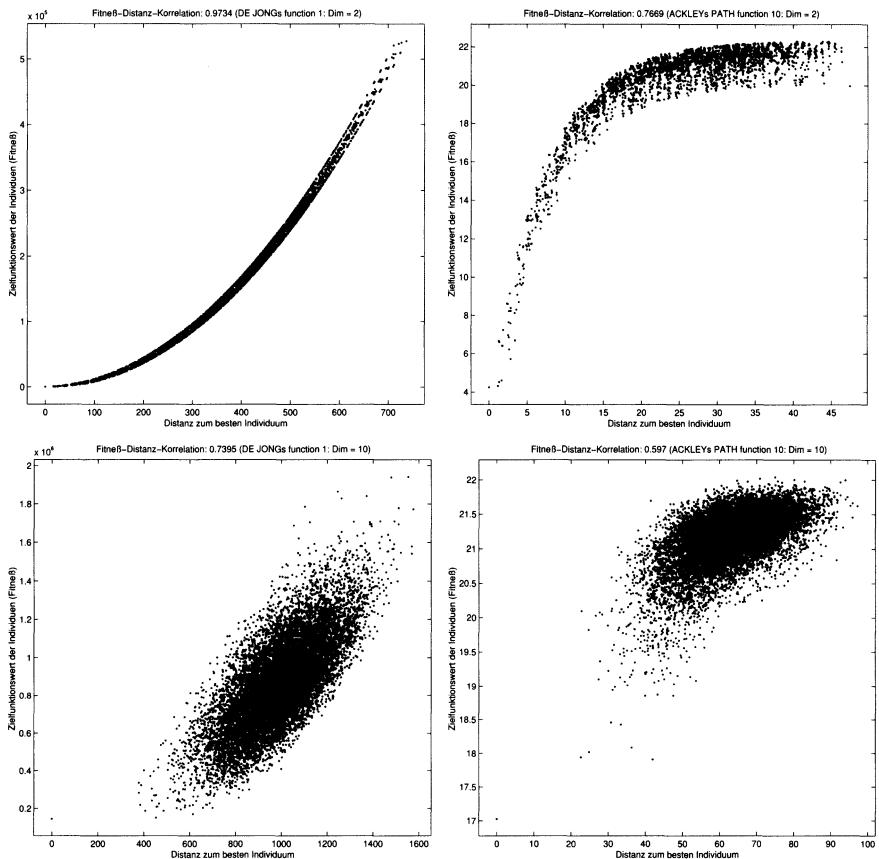
Im Rahmen von [Poh98], Unterabschn. 5.4.2, wird die Fitneß-Distanz-Verteilung auf weitere Standardtestfunktionen höherer Dimension und einer reellen Repräsentation angewendet. In diesen Untersuchungen zeigt sich, daß die Fitneß-Distanz-Verteilung bei niedrigdimensionalen Funktionen ( $n < 8$ ) Hinweise zur Schwere der Funktion geben kann. Für alle praktisch relevanten Beispiele mit höheren Dimensionen sind die Ergebnisse aber nicht zu interpretieren bzw. sehr stark verschwommen.

Zur Verdeutlichung dieser Erfahrungen sollen an dieser Stelle Beispiele für zwei bekannte Testfunktionen gezeigt werden. In Abb. 5-20 sind die Fitneß-Distanz-Verteilung und die Fitneß-Distanz-Korrelation zweier bekannter Testfunktionen dargestellt (DE JONG's Funktion 1 und ACKLEY's path). Die oberen Diagramme verwenden die Funktion in 2 Dimensionen, die unteren in 10 Dimensionen.

Während die oberen Diagramme gut interpretierbare Abbildungen liefern, sieht dies bei den unteren deutlich anders aus. Der Zusammenhang zwischen Zielfunktionswert und der Distanz zum besten Individuum ist immer weniger erkennbar. Bei der Verwendung noch höherer Dimensionen ist nur noch ein Punkthaufen zu sehen. Außerdem wird der Aussagewert der Maßzahl Fitneß-Distanz-Korrelation  $FDC$  (in der Titelzeile der Diagramme angegeben) mit zunehmender Dimension immer geringer.

Zusammenfassend muß festgestellt werden, daß die Fitneß-Distanz-Verteilung und -Korrelation nicht die Hoffnungen erfüllen konnte, die an ihren Vorschlag geknüpft waren. Für niedrigdimensionale bzw. einfache Funktionen können zwar zusätzliche Hinweise zu Eigenschaften der Funktion abgeleitet werden. Bei höherdimensionalen Funktionen ist dies aber nicht mehr möglich. Die zunehmende

Anzahl der Dimensionen verwischt die Zusammenhänge immer mehr, so daß aus den Diagrammen keine zusätzlichen Informationen entnommen werden können.



**Abb. 5-20.** Fitneß-Distanz-Verteilung in einem 2-D Punktdiagramm; oben: Darstellung der 2-dimensionalen Funktion, unten: wie oben als 10-dimensionale Funktion

### 5.5.3 Visualisierung problemspezifischer Daten und Ergebnisse

In allen bisherigen Verfahren zur Visualisierung wurde davon ausgegangen, daß die Variablen eines Individuums die kleinste bzw. genaueste Einheit der zur Verfügung stehenden Daten ist. Aus der Sicht des Evolutionären Algorithmus ist dies auch richtig. Der Evolutionäre Algorithmus arbeitet mit diesen Variablen, mehr „sieht“ der Algorithmus vom zu lösenden Problem nicht.

Für den Anwender stellen die Variablen, die vom Evolutionären Algorithmus bearbeitet werden, nur bei den einfachsten Problemen eine vollständige Beschreibung des zu bearbeitenden Systems dar. In fast allen praktischen Aufgaben sind

die Variablen eine Abstraktion bzw. repräsentieren nur einen kleinen Teil des Problems. Hier einige Beispiele:

- die Variablen stellen die Parameter für eine Simulation dar,
- die Variablen stellen die Steuerung eines zu simulierenden Systems dar,
- die Variablen bilden die Grundlage eines Touren- oder Produktionsplanes.

Für den Anwender ist es für eine genaue Bewertung der Güte einer Lösung bzw. den Vergleich ähnlicher Lösungen unerlässlich, daß nicht nur die Variablen verglichen werden, sondern auch der Verlauf der Simulation mit einer weitergehenden Darstellung des Verlaufs der Zustands- und Ausgangsgrößen der Simulation.

Für diese Darstellungen können keine allgemeinen Hinweise gegeben werden. Sie fallen für jedes zu lösende Problem unterschiedlich aus. Da sie aber direkt bei der Lösung des Problems helfen, sollte auf eine umfassende und verständliche Darstellung aller zur Verfügung stehenden Informationen Wert gelegt werden.

In der Anwendung dieser problemspezifischen Darstellungen lassen sich die folgenden Bereiche benennen:

- ausführliche Darstellung aller Zustands- und Ausgangsgrößen sowie weiterer Maßzahlen,
- Vergleich verschiedener Lösungen, d.h. Vergleich verschiedener Sätze von Variablen,
- „Überprüfung“ der Zielfunktion: Werden die verschiedenen Größen und Verläufe wirklich so bewertet, wie es der Anwender erwartet?

Der zweite und dritte Punkt stehen in direktem Zusammenhang. Oftmals kommt der Hinweis, daß eine Lösung mit einem schlechteren Zielfunktionswert doch eigentlich besser sei. An der Stelle muß dann die Zielfunktion so verändert werden, daß die bessere Lösung auch den besseren Zielfunktionswert hat. Dieser Prozeß kann mehrere Wiederholungen erfordern, bis die Zielfunktion eine Lösung so bewertet, wie es der Anwender erwartet. Ohne eine ausführliche und übersichtliche Visualisierung ist dies nicht zu erreichen.

Bei der Vorstellung von Anwendungen Evolutionärer Algorithmen auf praktische Probleme in Kap. 8, ab S.185, wird ausgiebig Gebrauch von der problemspezifischen Visualisierung gemacht. In diesen und allen weiteren bisher bearbeiteten Anwendungen wurden mit Hilfe der problemspezifischen Visualisierung Mängel der Zielfunktion bzw. der darunterliegenden Simulation aufgedeckt. Die Änderungen konnten nachfolgend komfortabel überprüft werden.

Es sei an dieser Stelle betont, daß die problemspezifische Visualisierung für die Bearbeitung und Lösung realer Probleme extrem wichtig ist. Meistens ist nur anhand dieser ausführlichen Darstellungen der problemspezifischen Daten eine genaue Bewertung sowie der Vergleich von mehreren Lösungen möglich. Deshalb sollte die Erstellung einer problemspezifischen Visualisierung mit einer der ersten Punkte bei der Bearbeitung eines neuen Systems sein – noch vor der Durchführung der Optimierung. Wie wichtig diese problemspezifischen Darstellungen sind, zeigt sich bei allen realen Anwendungen, die in Kap. 8 vorgestellt werden. Ohne diese Darstellungen wäre eine Auswertung und Dokumentation der Ergebnisse nicht möglich gewesen. Die vielen zusätzlichen Einblicke in das Verhalten des Systems, die während der Vorbereitung und Durchführung der Optimierungen ermittelt wurden, können hier gar nicht wiedergegeben werden.

## 5.6 Protokollierung von Daten und Ergebnissen

Die Protokollierung von Daten und Ergebnissen kann in drei Bereiche unterteilt werden:

- Ausgabe von Daten auf den Bildschirm während eines Laufs,
- Ausgabe von Daten in eine Textdatei zur späteren Auswertung,
- Ausgabe von Daten in eine binäre Datei zur späteren Auswertung.

Diese drei Bereiche unterscheiden sich signifikant im Umfang der Daten, die pro Schritt ausgegeben oder abgespeichert werden. Die Ausgabe auf den Bildschirm ist durch die Größe dieses Mediums stark beschränkt. Hier sollten nur die wichtigsten Daten, die etwas über den Fortschritt der Berechnungen aussagen, ausgegeben werden. Die Daten, die in einer vom Anwender direkt lesbaren Textdatei gespeichert werden, können deutlich umfangreicher sein. Genau betrachtet, ist die Ausgabe in eine Textdatei eine Erweiterung der Ausgabe auf den Bildschirm. Bei der Ausgabe von Daten in eine maschinenlesbare binäre Datei hängt der Umfang der ausgegebenen Daten nur vom Speicherplatz ab. Diese Variante dient dazu, all die Daten abzuspeichern, die für eine spätere detaillierte Auswertung eines oder mehrerer Läufe notwendig sind (Visualisierung). Je nach Aufgabenstellung kann der Datenumfang dem von der Ausgabe in die Textdatei entsprechen, wird aber meist um Größenordnungen umfangreicher sein.

Zur Ausgabe auf den Bildschirm haben sich die folgenden Daten bewährt:

- zu Beginn des Laufs:
  - alle Parameter des Evolutionären Algorithmus,
  - problemspezifische Parameter.
- für jede Generation (tabellarisch):
  - Nummer der Generation,
  - Anzahl der Zielfunktionsaufrufe,
  - Zielfunktionswert des besten Individuums,
  - Reihenfolge (und Größe) der Unterpopulationen (nur bei unterschiedlichen Strategien bzw. Konkurrenz),
  - Abstand zu einem der definierten Abbruchkriterien (prozentual)
  - Rechenzeit (letzte Generation und Gesamtrechenzeit).
- am Ende des Laufs:
  - Zielfunktionswert des besten Individuums und in welcher Generation es gefunden wurde,
  - Variablen des besten Individuums,
  - Grund des Abbruchs der Berechnungen mit Angabe der Generationen und Rechenzeit.

Die Ausgabe in eine Textdatei ist fast identisch zur Ausgabe auf den Bildschirm. Hier werden jedoch zusätzlich je Generation die Variablen des besten Individuums mit ausgegeben. Wenn weitere Maßzahlen für den Zustand der Population berechnet werden, so sollten diese hier mit ausgegeben werden (Diversität, Fixierung, Inzucht, u.a.).

Die Ausgabe auf den Bildschirm und die Speicherung haben eine ähnliche Aufgabe. Mit diesen Angaben ist einmal eine gute Kontrolle der fortschreitenden

Berechnungen möglich und durch die Speicherung in eine Textdatei eine problemlose spätere Auswertung (selbst, wenn die Optimierung zwischendurch abbricht). Durch die Speicherung der wichtigsten Angaben erfolgt gleichzeitig eine Dokumentation der durchgeföhrten Berechnungen.

Die Speicherung von Daten im binären Format dient der späteren detaillierten Auswertung. Dies betrifft insbesondere die grafische Auswertung. Dazu müssen alle Daten gespeichert werden, die für die Visualisierung benötigt werden.

Zur Speicherung in einer binären Datei sind die folgenden Daten notwendig:

- Daten für jede (oder alle  $x$ ) Generation:
  - Variablen aller Individuen einer Generation,
  - Zielfunktionswerte aller Individuen einer Generation.
- Daten für einen Lauf:
  - Variablen des besten Individuums je Generation,
  - Zielfunktionswerte des besten Individuums je Generation,
  - Reihenfolge und Größe der Unterpopulationen,
  - Index mit den Generationen, zu denen Daten gespeichert wurden.

Durch die Speicherung von Daten im binären Format sind die spätere komfortable Auswertung und die spezielle Visualisierung des Verlaufs sowie der Ergebnisse eines Laufs möglich. Gleichzeitig werden dadurch die Möglichkeiten der genauen Dokumentation des Verlaufs und der Ergebnisse verbessert.

Die hier vorgestellten drei Stufen der Protokollierung von Daten und Ergebnissen erlauben eine einfache und vollständige Aufzeichnung aller relevanten Daten eines Laufs. Damit können der Fortschritt eines Laufs genau beobachtet und eine ausführliche Auswertung vorgenommen werden. Je nach den Anforderungen und der Tiefe, mit der ein Problem behandelt wird, können alle Methoden miteinander kombiniert werden.

## 5.7 Zusammenfassung und Ausblick

Die Visualisierung von Daten und Ergebnissen stellt eines der leistungsfähigsten Werkzeuge zur Analyse großer Datenmengen dar. Bei der Verwendung Evolutionärer Algorithmen werden große Mengen an Daten produziert, die für eine effektive und verständliche Auswertung der Bearbeitung und übersichtlichen Darstellung bedürfen. In diesem Kapitel wurden die verschiedensten Verfahren und Techniken vorgestellt, die eine übersichtliche Darstellung für die unterschiedlichsten Daten vornehmen.

Die erläuterten und in ihrer Anwendung demonstrierten Verfahren decken die Bereiche ab, die bei der praktischen Anwendung von Evolutionären Algorithmen und für deren Analyse benötigt werden. Am häufigsten kommen die Verfahren zum Einsatz, die den Verlauf der Anwendung eines Evolutionären Algorithmus darstellen:

- bester Zielfunktionswert je Generation (Konvergenz),
- Variablen des besten Individuums je Generation,
- alle Zielfunktionswerte über alle Generationen.

Einen genaueren Einblick in den Zustand einer Population gewähren die Verfahren zur Darstellung der Daten einer Generation:

- Variablen aller Individuen,
- Zielfunktionswerte aller Individuen,
- Distanzverteilung.

Bei der Verwendung eines der Populationsmodelle ergeben sich mit den vorgestellten Verfahren vertiefte Einsichten oder es können beim Einsatz spezieller Visualisierungsverfahren neue Zusammenhänge erkannt werden:

- Position und Größe der Unterpopulationen beim regionalen Modell,
- Zielfunktionswerte aller Individuen,
- Distanzkarten beim lokalen Modell.

Die hochdimensionale Visualisierung hilft bei der Bewertung der Zusammenhänge zwischen Punkten, die im hochdimensionalen Raum verteilt liegen. Durch die Transformation in einen niedrigdimensionalen Bereich können die Daten dargestellt werden. Dies kann bei der Auswertung von Daten helfen, die auf anderem Wege nicht oder nur sehr schwer zugänglich wären bzw. eröffnet eine neue Sicht auf diese hochdimensionalen Daten. An Beispielen für die Darstellung von Individuen im Suchraum, des „Pfads der besten Individuen durch den Suchraum“ und die visuelle Bewertung mehrkriterieller Zielfunktionswerte wurde dies gezeigt. Die Anwendung der hochdimensionalen Visualisierung auf dem Gebiet der Evolutionären Algorithmen ist noch neu. Durch weitere Arbeiten auf diesem Gebiet dürften in nächster Zeit weitere Methoden und Verfahren aufbereitet werden, die über die hier gezeigten ersten Anwendungen und Ergebnisse hinausgehen.

In eine etwas andere Richtung gehen die Verfahren zur Darstellung von Eigenschaften der Zielfunktion. Die direkte Darstellung der Zielfunktion ist nur bei niedrigdimensionalen Funktionen bzw. bei Funktionen mit vertauschbaren Variablen vollständig anwendbar. Bei vielen praktischen Problemen kann ein Blick auf die direkte Darstellung trotzdem einige neue Informationen bringen, die anders nur schwer oder nicht zugänglich wären. Die Leistungsfähigkeit dieses einfachen Verfahrens wird bei der praktischen Anwendung oft übersehen.

Eine wichtige Unterstützung bei der Bearbeitung komplexer Probleme bildet die Visualisierung problemspezifischer Daten und Ergebnisse. Damit können die oftmals sehr abstrakten Variablen auf eine Art und Weise umgesetzt werden, daß der Anwender leicht zu interpretierende Ergebnisse zu sehen bekommt. Diese Ergebnisgrafiken sind ein wichtiges Hilfsmittel bei der Beurteilung des Verlaufs bzw. der Ergebnisse eines Laufs. Oftmals geben sie wichtige Rückschlüsse, ob die Zielfunktion wirklich die Dinge beinhaltet, die optimiert werden sollen bzw. welche Bereiche des Problems in der Zielfunktion noch nicht ausreichend berücksichtigt sind. Die problemspezifische Visualisierung sollte bei allen komplexen Problemen verwendet werden.

Alle in diesem Kapitel vorgestellten Visualisierungsmethoden entstanden bei der Bearbeitung und Untersuchung praktischer Probleme und haben sich im Einsatz bewährt. Die Verfahren ermöglichen ein genaues Verfolgen eines Laufs sowie die gleichzeitige oder nachfolgende Analyse wichtiger Größen. Ohne diese Darstellungen ist es fast unmöglich, sich durch die anfallenden Daten zu kämpfen und eine Analyse vornehmen zu können.

Je nach Schwere und Größe sowie dem schon vorhandenen Verständnis des zu bearbeitenden Problems können Verfahren von den hier vorgestellten für die Anwendung ausgewählt werden. Es lassen sich mehrere Detailstufen zusammenstellen, von einer einfachen Darstellung der Konvergenz am Ende des Laufs bis zur Verwendung aller vorgestellten Verfahren in jeder Generation.

Mit der weiteren Entwicklung der grafischen Darstellungsmöglichkeiten können die hier vorgestellten Methoden in wenigen Jahren deutlich erweitert werden. Es zeichnen sich schon heute stark verbesserte Möglichkeiten der dreidimensionalen Darstellung ab, die ein schnelles Betrachten von allen Seiten und ein Durchlaufen der Darstellung erlauben. Die Verfahren zur Animation von Grafiken erweitern sich sehr schnell und mit der nötigen Rechenleistung lässt sich damit bald die nächste Dimension nutzen.

## 6 Genetic and Evolutionary Algorithm Toolbox for Matlab

Dieser Abschnitt beschreibt die GENETIC AND EVOLUTIONARY ALGORITHM TOOLBOX FOR USE WITH MATLAB [GEATbx]<sup>1</sup>. Sie stellt eine Umgebung zum Arbeiten mit Evolutionären Algorithmen und eine Vielzahl von Funktionen und Routinen zur Implementierung spezieller Evolutionärer Algorithmen unter MATLAB [MW94] zur Verfügung.

In der GEATbx sind alle Algorithmen implementiert, die in den Kapiteln 3, 4 und 5 beschrieben wurden. Die in Kap. 8 gezeigten Anwendungen wurden mit der GEATbx bearbeitet und gelöst.

Die GEATbx wird seit 1995 kontinuierlich entwickelt. Im Verlauf der Jahre entstanden mehrere Versionen. Für den Einstieg in die Arbeit mit Evolutionären Algorithmen ist Version 1.9x am besten geeignet. Durch die Beschränkung auf grundlegende Verfahren und Operatoren wird die Einarbeitung erleichtert. Bereits mit dieser Version können viele Probleme bearbeitet werden.

In der Version 2.x der GEATbx sind alle in diesem Buch vorgestellten Verfahren und Operatoren implementiert. Außerdem enthält diese Version Verbesserungen und Erweiterungen, die sich aus der Anwendung der Toolbox auf große und komplexe Praxisprobleme ergaben (z.B. Visualisierung und Protokollierung). Version 2.x stellt zudem weitgehende Möglichkeiten der Beeinflussung von Operatoren und Optionen bereit. Bei der Lösung komplexer Praxisprobleme ermöglichen diese Eingriffsmöglichkeiten dem Anwender, den Optimierungsalgorithmus genau auf seine Erfordernisse abzustimmen.

Beide Versionen der GEATbx laufen unter MATLAB 5.x und werden bei Notwendigkeit auf neue MATLAB-Versionen angepasst. Version 1.9x läuft auch unter Matlab 4.2.

In diesem Kapitel wird ein Überblick zu Struktur und Aufbau der GEATbx gegeben. Dabei wird sich auf die Grundprinzipien beschränkt, die in der Entwicklung der Toolbox umgesetzt wurden. In den nachfolgenden Abschnitten werden die implementierten Operatoren und Verfahren genannt. Daher gelten die hier gemachten Aussagen im Prinzip für alle Versionen der GEATbx.

Für eine direkte Einführung in die Arbeit mit der GEATbx sei auf das Tutorial sowie die anderen Teile der Dokumentation der GEATbx verwiesen. Die Doku-

---

<sup>1</sup> Der in der Bezeichnung der Toolbox enthaltene Begriff *Evolutionary Algorithm* schließt den Begriff *Genetic Algorithm* mit ein. Da aber, gerade im englischen Sprachraum, der Begriff *Genetic Algorithm* weiterhin der gebräuchlichere ist, werden in der Bezeichnung der Toolbox beide Begriffe verwendet.

mentation enthält eine genaue Auflistung, welche Funktionen in der jeweiligen Version enthalten sind.

Für die GEATbx wird eine spezielle Webseite – <http://www.geatbx.com/> – unterhalten. Auf dieser steht die Dokumentation der verschiedenen Versionen der GEATbx zur Verfügung. Außerdem sind dort weitere Informationen zu Evolutionären Algorithmen zu finden.

Diesem Buch liegt eine CD bei, auf der die vollständige Version 1.95 der GEATbx enthalten ist. Dies beinhaltet den Quelltext (*m-files*) der Toolbox sowie die Dokumentation. Mit dem Kauf des Buches hat der Leser gleichzeitig eine Einzelnutzer-Lizenz der GEATbx erworben. Diese Lizenz berechtigt auch zu vergünstigten Updates auf neue Versionen der GEATbx. Angaben zu weiteren Lizenzbedingungen (*multi user license, site license*), Updatebedingungen sowie Adressen zum Bestellen der Toolbox sind auf der Webseite der GEATbx zu finden (<http://www.geatbx.com/contact.html>) bzw. können per Email erfragt werden: support@geatbx.com. Außerdem wird eine Mailingliste unterhalten, um die Anwender der GEATbx über neue Funktionen und Versionen zu informieren. Senden Sie eine Email an support@geatbx.com mit Ihrer Emailadresse.

## 6.1 Aufbau und Struktur der GEATbx

Die Toolbox ist in mehreren Schichten aufgebaut. Abbildung 6-1 gibt einen Überblick über das der Toolbox zugrunde liegende Schichtenmodell. Im Mittelpunkt steht eine Zentralfunktion. Auf diese Funktion wird von außen über die Anwenderschnittstelle (Scriptfunktionen) zugegriffen. Eine Zwischenschicht enthält vordefinierte Algorithmen (Toolboxfunktionen). Die Zentralfunktion ruft alle benötigten evolutionären Operatoren auf. Dieser Aufruf der evolutionären Operatoren erfolgt über zwei Schichten. Außerdem übernimmt die Zentralfunktion den größten Teil der Verwaltungsaufgaben und führt verschiedene Auswertungen durch.

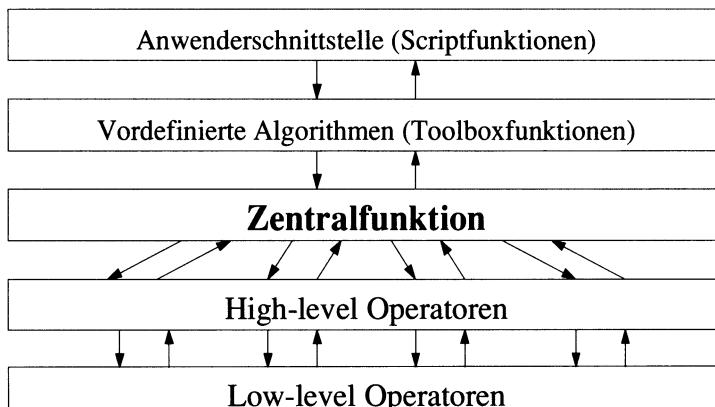


Abb. 6-1. Schichtenmodell der GEATbx

Dieses abstrahierte Schichtenmodell soll im folgenden weiter unterstellt werden. Abbildung 6-2 zeigt die Struktur der GEATbx unter Verwendung der Namen der entsprechenden Matlab-Funktionen detaillierter. Auch aus dieser Abbildung ist das Schichtenmodell gut zu erkennen. Mittelpunkt ist die Zentralfunktion **geemain**. Der Aufruf von **geemain** erfolgt über die Parametrisierungsfunktionen, die zweischichtig ausgeführt sind (**scr\*** und **tbx\***). Die Zentralfunktion selbst ruft die evolutionären Operatoren auf, auch dieser Aufruf ist zweischichtig strukturiert.

Von den Parametrisierungsschichten beinhaltet die untere Ebene die sogenannten Toolboxfunktionen **tbx\***, in denen jeweils die Parameter für einen Evolutionären Algorithmus definiert sind. Jede der **tbx**-Funktionen definiert einen anderen Algorithmus. Die oberste Schicht ist für den Anwender bestimmt und enthält die sogenannten Scriptfunktionen **scr\***. In diesen können die speziellen Parameter für das gerade zu lösende Problem definiert werden. Dabei müssen nur die Parameter angegeben werden, die anders als in der verwendeten **tbx**-Funktion bzw. anders als die Standardparameter der Toolbox sind.

### Aufrufbaum der Genetic and Evolutionary Algorithm Toolbox

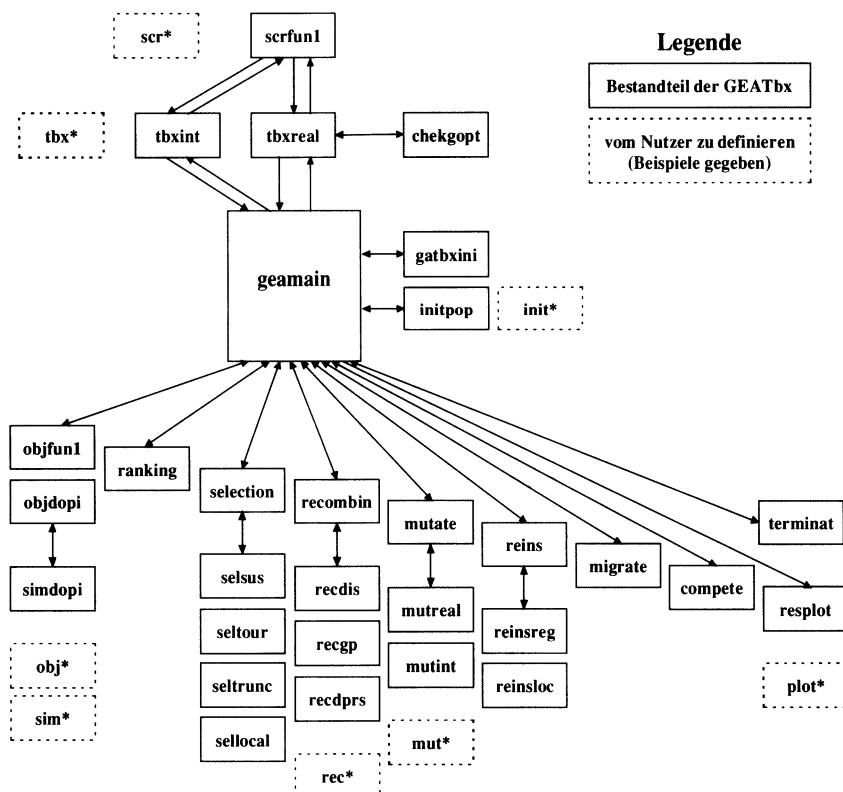


Abb. 6-2. Aufrufbaum der GEATbx

Die Zentralfunktion ***geamain*** bietet eine einheitliche Umgebung für die verschiedenen, mit der Toolbox zu verwendenden Evolutionären Algorithmen. Dies beinhaltet neben dem Aufruf der evolutionären Operatoren die gesamte Parameterverwaltung, die Verbindung der verschiedenen Algorithmen und die Auswertung der Ergebnisse.

Der Aufruf der evolutionären Operatoren erfolgt in zwei Stufen. Von der Zentralfunktion wird die oberste Schicht aufgerufen, die allgemeine Dinge, wie z.B. die Funktionen zur Selektion (***selection***), Rekombination (***recombin***) und Mutation (***mutate***) enthält. Diese Funktionen rufen dann selbst die unterste Schicht mit den speziellen evolutionären Operatoren auf.

Für die Behandlung von Populationen sind die in Kap. 4, ab S.73, erläuterten Verfahren implementiert (regionales Modell: in allen high-level Operatoren und ***migrate***; lokales Modell: ***selllocal*** und ***reinsloc***, Konkurrenz: ***compete***).

Die von Seiten des Nutzers wichtigste Funktion ist die Zielfunktion (*objective function* ***obj\****). Diese Funktion muß für jedes Problem speziell definiert werden. Mit der Toolbox werden eine große Anzahl von Zielfunktionen als Beispiele mitgeliefert, die einen guten Überblick über die Verschiedenartigkeit dieser Funktionen geben. Weitere Beispiele sind in Kap. 8 beschrieben.

Die gesamten Visualisierungsmöglichkeiten aus Kap. 5, ab S.115, sind in einer Funktion (***resplot***) implementiert, die nach Abschluß einer Generation von der Zentralfunktion aufgerufen werden kann.

Innerhalb der gesamten GEATbx wurde eine einheitliche Benennung der Funktionen angewendet. Alle Funktionen bzw. Operatoren einer Gruppe beginnen mit demselben Präfix. In Tab. 6-1 sind diese Namenskonventionen aufgeführt. Dies erleichtert den schnellen Überblick und die Einordnung der einzelnen Funktionen

Neben den bisher erwähnten Funktionen werden noch eine Reihe von Hilfsfunktionen benötigt, die an den verschiedensten Stellen zum Einsatz kommen.

**Tabelle 6-1.** Namenskonventionen für die Benennung von Funktionen in der GEATbx

Präfix	Funktions- oder Operatorgruppe
<b><i>scr</i></b>	Scriptfunktionen
<b><i>tbx</i></b>	Toolboxfunktionen
<b><i>sel</i></b>	Selektionsoperatoren
<b><i>rec</i></b>	Rekombinationsoperatoren (auch <i>crossover</i> )
<b><i>mut</i></b>	Mutationsoperatoren
<b><i>reins</i></b>	Wiedereinfügefunktionen
<b><i>obj</i></b>	Zielfunktionen ( <i>objective functions</i> )
<b><i>sim</i></b>	Simulationsroutinen, von den Zielfunktionen aufgerufen
<b><i>init</i></b>	spezielle Initialisierungsfunktionen
<b><i>plot</i></b>	spezielle Anzeigefunktionen ( <i>plot functions</i> )

Der modulare Aufbau der Toolbox und die Verwendung sinnvoller Standardparameter in Verbindung mit fertig definierten Algorithmen erlaubt ein Arbeiten

mit der Toolbox, das sich je nach Interesse des Anwenders zwischen Verwendung der vorgegebenen Evolutionären Algorithmen und Parameter bis hin zu individueller Veränderung jedes einzelnen Parameters von außen bewegt.

In den folgenden Unterabschnitten werden die einzelnen Funktionsgruppen kurz erläutert und jeweils die Verbindung zu den in den vorigen Abschnitten erläuterten Algorithmen aufgestellt. Die Erläuterung erfolgt entsprechend der Reihenfolge, in der die Funktionen bei der Berechnung aufgerufen werden.

## 6.2 Anwenderschnittstelle – Scriptfunktionen

Den Beginn der Toolbox und damit die Schnittstelle zum Anwender stellen die Scriptfunktionen dar. Neben der Definition problemspezifischer Parameter werden von diesen Funktionen Toolboxfunktionen mit vordefinierten Evolutionären Algorithmen aufgerufen. Die vorgefertigten Scriptfunktionen stellen jeweils ein Beispiel einer bestimmten Optimierung oder Funktionalität dar.

**Tabelle 6-2.** Beispiele für Scriptfunktionen

Script	Beschreibung
<i>scrfun1</i>	Beispielscript für eine Optimierung der Zielfunktion <i>objfun1</i> – DE JONG's Funktion 1
<i>scrbin1</i> / <i>scrint2</i>	Beispielscript für die Verwendung einer Zielfunktion mit binären bzw. ganzzahligen (integer) Variablen
<i>scrdopi</i>	Beispielscript für die Optimierung eines dynamischen Systems – Optimierung der Steuerung des Doppelintegrators

Dabei müssen nur die Parameter angegeben werden, die von den Standardparametern der Toolbox und den Parametern, die in aufgerufenen Toolboxfunktionen definiert sind, abweichen. Im Extremfall kann dies bedeuten, daß nur der Aufruf der gewählten Toolboxfunktion *tbx\** mit dem entsprechenden Namen der Zielfunktion enthalten ist. Auf der anderen Seite können in einer Scriptfunktion alle Parameter der Toolbox geändert und neu gesetzt werden.

## 6.3 Vordefinierte Algorithmen – Toolboxfunktionen

Die Toolboxfunktionen definieren jeweils einen spezifischen Evolutionären Algorithmus. Dazu werden in der Funktion die dafür notwendigen Parameter gesetzt. Die Toolboxfunktionen erlauben die komplette Definition bzw. Zusammenstellung der unterschiedlichsten Evolutionären Algorithmen. Für die verschiedensten Einsatzfälle können damit fertige Algorithmen bereitgestellt werden. Dies erleichtert die Anwendung der GEATbx erheblich.

**Tabelle 6-3.** Beispiele für vordefinierte Evolutionäre Algorithmen (Toolboxfunktionen)

Funktion	Beschreibung
<i>tbxreal</i>	parametrisiert einen <i>Evolutionären Algorithmus für reelle Variablen</i> (lineare Ranking-Selektion, diskrete Rekombination, reelle Mutation mit verschiedenen Strategien, mehrere Unterpopulationen – regionales Modell)
<i>tboxes1</i>	parametrisiert die Verwendung einer Evolutionsstrategie (reelle Variablen, Truncation-Selektion, keine Rekombination, eine der Evolutionsstrategie-Mutationen, 1 Gesamtpopulation)
<i>tbxtsp</i>	Evolutionärer Algorithmus für die Optimierung von TSP Problemen (Permutation von Variablen, <i>edge recombination, reverse/swap mutation</i> )
<i>tboxbin</i>	parametrisiert einen <i>Evolutionären Algorithmus für binäre Variablen</i> (lineare Ranking-Selektion, <i>uniform crossover</i> Rekombination, binäre Mutation, mehrere Unterpopulationen – regionales Modell)

## 6.4 Zentralfunktion

Von den Script- bzw. Toolboxfunktionen wird jeweils die Zentralfunktion der Toolbox aufgerufen:

- **geemain**.

In der Zentralfunktion findet die gesamte Verwaltungsarbeit statt. Neben der Initialisierung der internen Variablen und Strukturen erfolgt die Initialisierung der Population. Danach beginnt die Ausführung des Evolutionären Algorithmus mit der einhergehenden Anzeige und Sicherung von Daten und Ergebnissen.

Von der Zentralfunktion werden für die Ausführung des Evolutionären Algorithmus nur high-level Funktionen aufgerufen. Dies ermöglicht eine einfache Unterstützung der verschiedenen Konzepte zur Behandlung von Populationen, speziell des regionalen Modells. Der Ablauf des Evolutionären Algorithmus lässt sich durch die folgenden Schritte und die korrespondierenden Funktionen untergliedern:

- Fitneßzuweisung (**ranking**)
- Selektion (**selection**)
- Rekombination (**recombin**)
- Mutation (**mutate**)
- Berechnung der Zielfunktion (**obj\***)
- Wiedereinfügen (**reins**)
- regionales Modell – Migration (**migrate**)
- Konkurrenz (**compete**)
- Visualisierung (**resplot**)
- Terminierung (**terminat**)

Innerhalb der high-level Funktionen werden die parametrisierten evolutionären Operatoren (low-level Funktionen) aufgerufen.

### 6.4.1 Initialisierung

Eine Initialisierung findet in der Zentralfunktion an zwei Stellen statt. Zuerst werden die übergebenen Parameter für die einzelnen Operatoren und die Arbeitsweise des Evolutionären Algorithmus den internen Strukturen zugewiesen. Danach erfolgt die Initialisierung der Individuen der ersten Generation. Dies kann zufällig erfolgen oder durch eine erweiterte Initialisierung.

**Tabelle 6-4.** Initialisierungsfunktionen für Individuen

Funktion	Beschreibung
<i>initrp/ip/bp</i>	initialisiert Individuen mit reellen/ganzzahligen/binären Variablen
<i>initpp</i>	initialisiert Individuen mit einer Permutation ganzer Zahlen
<i>initpop</i>	initialisiert eine Population von Individuen durch Verwendung und Abwandlung übergebener Individuen (s. Abschn. 3.6)
<i>initdopi</i>	benutzerspezifische Funktion zur Initialisierung von Individuen für die Zielfunktion <i>objdopi</i> – Optimierung der Steuerung des Doppelintegrators
<i>initgrhs</i>	benutzerspezifische Funktion zur Initialisierung einer Population von Individuen für die Zielfunktion <i>objgrhs</i> – Steuerung des Gewächshausklimas

### 6.4.2 Fitneßzuweisung und Selektion

Für die Fitneßzuweisung wird nur eine Funktion benötigt:

- **ranking** (lineares und nichtlineares Ranking).

Diese verwendet in Abhängigkeit des parametrisierten Selektionsdruckes lineares oder nichtlineares Ranking. Eine low-level Funktion wird nicht benötigt.

Zusätzlich wird beim Vorliegen mehrerer Zielfunktionswerte pro Individuum ein mehrkriterielles Ranking durchgeführt. Dabei kommt Pareto-Ranking in Verbindung mit *goals* und *fitness sharing* zur Anwendung, s. Unterabschn. 3.1.3.

**Tabelle 6-5.** Selektionsoperatoren

Funktion	Beschreibung
<i>selsus</i>	<i>stochastic universal sampling</i>
<i>selrws</i>	Rouletteselektion
<i>seltrunc</i>	Truncation-Selktion
<i>seltour</i>	Turnierselektion
<i>sellocal</i>	lokale Selektion (lokales Modell)

Eine einheitliche Schnittstelle für alle Selektionsoperatoren wird durch eine high-level Funktion bereitgestellt:

- ***selection*** (high-level Selektionsfunktion).

In Abhängigkeit des parametrisierten Selektionsalgorithmus ruft diese Funktion für jede Unterpopulation den jeweiligen Selektionsoperator auf, der dann die direkte Auswahl der Individuen durchführt.

#### 6.4.3 Rekombination

Für alle Rekombinationsverfahren steht mit einer high-level Funktion eine einheitliche Schnittstelle zur Verfügung:

- ***recombin*** (high-level Rekombinationsfunktion).

Je nach parametrisiertem Rekombinationsalgorithmus wird für jede Unterpopulation der entsprechende Rekombinationsoperator aufgerufen. Hierbei muß darauf geachtet werden, daß die Variablen der Individuen durch den jeweiligen Operator sinnvoll rekombinierbar sind. Soweit möglich werden entsprechende Kompatibilitätstests durchgeführt. (Durch die Verwendung fertiger Toolboxfunktionen kann dieses Problem leicht umgangen werden.)

**Tabelle 6-6.** Rekombinationsoperatoren

---

Funktion	Beschreibung
<b><i>recdis</i></b>	diskrete Rekombination
<b><i>recint</i></b>	intermediäre Rekombination
<b><i>reclin</i></b>	Linien-Rekombination
<b><i>reclinex</i></b>	erweiterte Linien-Rekombination
<b><i>recsp/recdp/recsh</i></b>	<i>single point / double point / shuffle crossover</i>
<b><i>recsprs/dprs/shrs</i></b>	<i>single point / double point / shuffle crossover mit reduced surrogate</i>
<b><i>recgpx</i></b>	<i>position crossover (order crossover)</i>
<b><i>recede</i></b>	<i>edge recombination</i>

---

Alle Rekombinationsoperatoren für binäre Variablen sind intern auf eine Funktion, ***recmp***, zurückgeführt, die alle Rekombinationsalgorithmen für binäre Variablen implementiert und jeweils nur mit anderen Parametern aufgerufen wird.

Für ganzzahlige Variablen wird im Moment nur die diskrete Rekombination, ***recdis***, bereitgestellt. Mit entsprechenden Anpassungen können natürlich die anderen Operatoren für reelle Variablen auch bei ganzzahligen Variablen eingesetzt werden.

#### 6.4.4 Mutation

Genau wie die Rekombination wird die Mutation über eine high-level Funktion aufgerufen, die eine einheitliche Schnittstelle für die verschiedenen Mutationsooperatoren bietet:

- ***mutate*** (high-level Mutationsfunktion).

Trotz einer recht unterschiedlichen Arbeitsweise, verschiedener Parameter und anderer Datenstruktur wird durch die Integration innerhalb der GEATbx eine einheitliche Handhabung aller Mutationsoperatoren ermöglicht. Jede Unterpopulation wird mit dem jeweils festgelegten Mutationsoperator bearbeitet (regionales Modell).

**Tabelle 6-7.** Mutationsoperatoren

Funktion	Beschreibung
<b><i>mutreal</i></b>	Mutation reeller Variablen
<b><i>mutes1</i></b>	Mutation einer Evolutionsstrategie <i>derandomized ES-algorithm with individual step sizes</i> [OGH93]
<b><i>mutes2</i></b>	Mutation einer Evolutionsstrategie <i>derandomized mutative step-size control using accumulated information</i> [OGH93]
<b><i>mutint</i></b>	Mutation ganzzahliger ( <i>integer</i> ) Variablen
<b><i>mutbin</i></b>	Mutation binärer Variablen
<b><i>mutinvert</i></b>	Mutation durch Inversion von Variablen
<b><i>mutswap</i></b>	Mutation durch Austausch bzw. Verschieben von Variablen

#### 6.4.5 Wiedereinfügen

Zum Einfügen der Nachkommen in die Population wird eine high-level Funktion bereitgestellt:

- ***reins*** (high-level Wiedereinfügefunktion).

Entsprechend dem verwendeten Populationsmodell wird von dieser Funktion eine low-level Funktion zum Einfügen der Nachkommen aufgerufen:

- ***reinsreg*** (regionales Wiedereinfügen der Individuen)
- ***reinsloc*** (lokales Wiedereinfügen von Individuen in die Unterpopulation)

Die Wahl der Wiedereinfügefunktion hängt nur von der Art des Selektionsoperators (und damit des Populationsmodells) ab. Bei der Verwendung der lokalen Selektion (lokales Populationsmodell) muß auch das lokale Wiedereinfügen verwendet werden. Diese Entscheidung wird in der high-level Wiedereinfügefunktion automatisch vorgenommen.

### 6.4.6 Migration und Konkurrenz zwischen Unterpopulationen

Das regionale Modell wird durch die Zentralfunktion sowie alle high-level Operatoren unterstützt. Für den Austausch von Individuen zwischen den Unterpopulationen (Migration) wird eine Funktion bereitgestellt, die alle Fähigkeiten enthält, die in Abschn. 4.4 erläutert wurden:

- ***migrate*** (Migration von Individuen zwischen den Unterpopulationen).

Die Funktionsweise der Migration kann mit wenigen Parametern komplett von außen gesteuert werden.

Für die Durchführung der Konkurrenz zwischen den Unterpopulationen wird eine weitere Funktion bereitgestellt, welche die gesamte Funktionalität aus Abschnitt 4.6 beinhaltet:

- ***compete*** (Konkurrenz zwischen den Unterpopulationen – *competition*).

Die Anwendung dieser Funktion setzt die Nutzung des regionalen Modells voraus. Die Funktionsweise der Konkurrenz zwischen den Unterpopulationen kann mit wenigen Parametern komplett von außen gesteuert werden.

Die Verwendung verschiedener Strategien für die einzelnen Unterpopulationen ist direkter Bestandteil der Parametrisierung der GEATbx. Eine Definition verschiedener Operatoren oder Parameter für die einzelnen Unterpopulationen führt zum Einsatz verschiedener Strategien je Unterpopulation.

### 6.4.7 Terminierung

Für den Abbruch der Berechnungen werden von der Toolbox verschiedene Möglichkeiten (s. Abschn. 3.7) unterstützt. Diese Abbruchkriterien werden alle innerhalb einer Funktion getestet, die von der Zentralfunktion zum Ende einer Generation aufgerufen wird:

- ***terminat*** (meldet, ob eines der eingestellten Abbruchkriterien erreicht ist).

Mit dieser Funktion kann getestet werden, ob:

1. eine maximale Anzahl an Generationen vergangen ist,
2. eine maximale Zeit für die Berechnungen vergangen ist,
3. ein bestimmter Zielfunktionswert erreicht wurde,
4. seit einer Anzahl von Generationen kein Fortschritt mehr erzielt wurde.

Alle diese Kriterien können einzeln oder zusammen angewendet werden. Sobald eines der Kriterien erfüllt ist, werden die Berechnungen beendet. Ein Hinweis über den Grund des Endes der Berechnungen wird in der Auswertung angegeben.

### 6.4.8 Visualisierung

Die GEATbx unterstützt alle Visualisierungsmöglichkeiten, die in Kap. 5, ab S.115 beschrieben wurden. Alle Grafikfunktionen sind in einer Funktion implementiert:

- ***resplot*** (grafische Auswertung der Zwischenergebnisse).

Die Anzeige problemspezifischer Ergebnisse (Zustandsgrafiken) erfolgt mit speziellen Funktionen, die für jede Zielfunktion einzeln zu definieren sind, z.B.:

- ***plotdopi*** (Zustandsgrafiken des jeweils besten Individuums von ***objdopi***),
- ***plotauto*** (Zustandsgrafiken der Fahrzeuglenkung, ***objauto***),
- ***plotgrhs*** (Zustandsgrafiken der Gewächshausklimatisierung, ***objgrhs***).

### 6.4.9 Protokollierung

Die Protokollierung von Zwischen- und Endergebnissen ist direkter Bestandteil der Zentralfunktion.

Es besteht die Möglichkeit zur Anzeige der Ergebnisse auf dem Bildschirm. Dadurch ist der Nutzer über den Fortgang und Stand der Berechnungen auf dem Laufenden. Im Vergleich zur Visualisierung werden hierbei nur wenige wichtige Daten angezeigt (Generation, Anzahl der Zielfunktionsaufrufe, Zielfunktionswert des besten Individuums, bei Konkurrenz bzw. unterschiedlichen Strategien auch Größe und Reihenfolge der Unterpopulationen, Erfüllung der eingestellten Abbruchkriterien).

Dieselben Angaben, die zur Anzeige auf den Bildschirm kommen, können in einer Textdatei gesichert werden. Dabei wird zusätzlich das beste Individuum (Variablen des Individuums) mit abgespeichert.

Zu Beginn der Berechnungen erfolgt jeweils eine vollständige Anzeige bzw. Speicherung aller Parameter des aktuellen Laufs. Am Ende wird eine Auswertung mit Angabe des besten Individuums, seines Zielfunktionswertes und in welcher Generation es gefunden wurde, gegeben. Außerdem wird die aufgewendete Rechenzeit angezeigt.

Mit diesen Angaben ist einmal eine gute Kontrolle der fortschreitenden Berechnungen möglich und zum anderen durch die Speicherung in eine Textdatei die problemlose spätere Auswertung. Durch die Speicherung der wichtigsten Angaben erfolgt gleichzeitig eine Dokumentation der durchgeföhrten Berechnungen.

Zusätzlich können alle Daten, die für die Visualisierung benötigt werden, in einer binären Datei (mat-Datei, spezifisches Format von Matlab) gespeichert werden. Durch einen Parameter wird angegeben, wie oft diese Daten gespeichert werden. Eine spezielle Funktion dient dem späteren Auslesen und Aufbereiten dieser Daten sowie dem Aufruf der Visualisierungsfunktion:

- ***reslook*** (Auslesen und Aufbereiten der gespeicherten Daten sowie Aufruf von ***resplot***).

Damit ist die Möglichkeit gegeben, längere Berechnungen ohne Visualisierung durchzuführen und sich hinterher schnell und einfach die Ergebnisse anzuschauen. Gleichzeitig ist dadurch die Möglichkeiten der genauen Dokumentation des Verlaufs und der Ergebnisse gegeben.

## 6.5 Zielfunktionen und Beispiele

In den Zielfunktionen wird jeweils der Zielfunktionswert eines Individuums berechnet. Dies stellt die Schnittstelle zwischen dem Evolutionären Algorithmus und dem zu lösenden Problem dar. Für jedes zu lösende Problem muß eine eigene Zielfunktion geschrieben werden.

Für die ersten Schritte mit der Toolbox und als Grundlage für eigene Zielfunktionen sind in der Toolbox eine große Anzahl von Beispiefunktionen enthalten.

Die Beispiefunktionen lassen sich in verschiedene Gruppen unterteilen, für die im folgenden jeweils einige der in der Toolbox enthaltenen Beispiefunktionen angegeben werden.

Standardtestfunktionen (n-dimensional) mit reellen Variablen:

- ***objfun1*** (DE JONG's Funktion 1 [DeJ75]),
- ***objfun2*** (ROSENBROCK's Funktion, auch Bananenfunktion),
- ***objfun1a*, *objfun1b*** (Abwandlungen von *objfun1* [Sch81]),
- ***objfun6* – *objfun12*** (weitere reelle Testfunktionen, u.a. RASTRIGIN, SCHWEFEL [Sch81], GRIEWANGK, ACKLEY' Path [Ack87], LANGERMANN [Lan95], MICHALEWICZ [Mic92]).

Testfunktionen für die Verwendung binärer bzw. ganzzahliger Variablen:

- ***objone1*** (ONEMAX Funktion),
- ***objint1*** (INTMAX Funktion).

Standardtestfunktionen zur Optimierung mit zwei reellen Variablen:

- ***objbran*** (BRANIN's rcos Funktion [Bra72]),
- ***objeaso*** (EASOM's Funktion [Eas90]),
- ***objgold*** (GOLDSTEIN-PRICE Funktion [GP71]),
- ***objsixh*** (*six hump camelback* Funktion [DS78]).

Mehrkriterielle Testfunktionen:

- ***mopfonseca1*** (zweidimensionale Funktion mit 2 Zielfunktionswerten),
- ***mopfonseca2*** (n-dimensionale Funktion mit 2 Zielfunktionswerten),
- weitere ***mop\**** Funktionen (s. auch [Vel99] und [ZDT99]).

Funktionen zur Optimierung dynamischer Systeme:

- ***objdopi*** (Steuerung eines Doppelintegrators),
- ***objlinq*** (linear quadratisches Problem [Mic92]),
- ***objauto*** (Regelung einer Fahrzeuglenkung, Abschn. 8.4),
- ***objgrhs*** (Steuerung des Gewächshausklimas, Abschn. 8.5).

An die Zielfunktionen können zusätzliche Parameter übergeben werden. Dies wird unter anderem bei der Optimierung der Steuerung eines Doppelintegrators (***scrdopi***, ***objdopi***, ***initdopi***, ***plotdopi***) genutzt. Dadurch wird die vollständige Steuerung der Arbeitsweise der Zielfunktion von außen möglich.

Eine ausführliche Beschreibung aller in der Toolbox enthaltenen Zielfunktionen ist in [GEATbx] enthalten. Einige der Standardtestfunktionen werden in Abschn. 8.2, ab S.194, zur Veranschaulichung der Optimierung einer mathematischen Funktion eingesetzt.

## 6.6 Einbeziehung problemspezifischen Wissens

Die GEATbx bietet mehrere Schnittstellen für die einfache Einbeziehung problemspezifischen Wissens:

- spezielle Initialisierung der Anfangsgeneration, s. Abschn. 3.6,
- Anzeige erweiterter Ergebnisgrafiken für Individuen (Zustandsgrafiken, problemspezifische Visualisierung),
- Verwendung spezieller Selektions-, Rekombinations- und Mutationsoperatoren.

Für die spezielle Initialisierung (*init\**) und die Ausgabe von Zustandsgrafiken (*plot\**) sind eine Reihe von Beispelfunktionen in der Toolbox enthalten. Die Namen der jeweils zu benutzenden Funktionen werden in der Scriptfunktion definiert, die Funktionen selbst von der Zentralfunktion an den entsprechenden Stellen aufgerufen. Dabei orientiert sich der Aufruf dieser Funktionen an dem der Zielfunktion. Im besonderen heißt dies, daß alle zusätzlichen Parameter, die an die Zielfunktion übergeben werden, auch an die speziellen Initialisierungs- und Ausgabefunktionen übergeben werden. Dies sorgt für eine einfache Anwendung sowie eine hohe Flexibilität.

Ein Beispiel der Einbeziehung problemspezifischen Wissens ist mit den Funktionen zur Optimierung der Steuerung eines Doppelintegrators (*initdopi*, *plotdopi*) oder der Gewächshaussteuerung (*initgrhs*, *plotgrhs*) gegeben. Außerdem machen die in Kap. 8 vorgestellten Anwendungen starken Gebrauch problemspezifischer Initialisierungs- und Visualisierungsfunktionen. Die zur Darstellung und zum Vergleich der Ergebnisse verwendeten Grafiken sind direkte Abbilder dieser problemspezifischen Visualisierungsfunktionen zur Darstellung wichtiger Größen der jeweiligen Systeme.

Die Einbeziehung spezieller evolutionärer Operatoren entspricht der Definition eines neuen Operators (wobei ein existierender als Grundlage verwendet werden kann). Der Aufruf dieser Operatoren erfolgt dann anstelle eines der in der Toolbox enthaltenen Operatoren.

## 6.7 Dokumentation

Für die Dokumentation der GEATbx mußte eine Variante gefunden werden, die dieselbe leichte und problemlose Portierbarkeit ermöglicht, wie dies für den Quelltext der GEATbx der Fall ist. Deshalb wurde die gesamte Dokumentation in HTML geschrieben. Für alle Rechnerplattformen, auf denen Matlab verfügbar ist, gibt es auch einen HTML-Browser zum Ansehen der Dokumentation.

Die Dokumentation der GEATbx ist in drei Teile gegliedert. Im ersten Teil, dem Tutorial, wird eine Einführung in die Arbeit und die Anwendung der Toolbox gegeben. Eine ausführliche Erläuterung der implementierten Algorithmen sowie einiger theoretischer Erkenntnisse bilden den zweiten Teil. Im dritten Teil werden alle Bestandteile der Toolbox ausführlich dokumentiert. Dies beinhaltet die Dokumentation des Quelltextes aller Funktionen und Routinen, die Bedeutung und möglichen Belegungen der Parameter sowie eine ausführliche Erläute-

rung aller Beispielfunktionen. Alle Teile der Dokumentation der GEATbx sind durch Querverweise miteinander verbunden, wodurch eine einfache Navigation zwischen den Teilen und zu verwandten Themen gewährleistet ist.

Die Dokumentation des Quelltextes der Toolbox wird durch ein speziell entwickeltes Perlscript generiert. Dieses Skript erstellt alle notwendigen Verzeichnisse, extrahiert Informationen, faßt Angaben zusammen und führt eine Aufsplittung des Quelltextes jeder Funktion in die einzelnen Bereiche (Funktionskopf, Hilfertext, Quelltext) durch. Außerdem werden alle Abhängigkeiten zwischen den Funktionen erkannt und für jede Funktion die aufrufenden und aufgerufenen Funktionen zusammengestellt. Das Perlscript arbeitet über einen Verzeichnisbaum. Damit kann eine Vorstrukturierung der Funktionen in Gruppen vorgenommen werden, die sich dann in der Dokumentation widerspiegeln.

Durch die automatisierte Erstellung der Dokumentation des Quelltextes ist gewährleistet, daß die Dokumentation nach Tastendruck auf dem neuesten Stand ist und selbst letzte Änderungen sofort berücksichtigt werden. Weiterhin wird durch die Hypertextverbindungen zwischen den Funktionen ein schnelles Kennenlernen der GEATbx auf der Ebene des Quelltextes ermöglicht. Schließlich wird die Aufgabe zur Kommentierung der Implementierung von Algorithmen direkt in den Quelltext verlagert und muß dadurch nur an einer Stelle vorgenommen werden.

## **6.8 Zusammenfassung und Ausblick**

Mit der GEATbx steht ein Werkzeug zur komfortablen Arbeit mit Evolutionären Algorithmen zur Verfügung. Eine große Anzahl von vordefinierten Algorithmen ist bereits in der Toolbox enthalten. Durch die Wahl bewährter Standardparameter ist der Einstieg in die Arbeit mit Evolutionären Algorithmen und deren Anwendung einfach. Der erfahrene Anwender kann das Verhalten jedes Operators und jeder Funktion von außen kontrollieren und verändern.

Durch die konsequent plattformunabhängige Programmierung der GEATbx, die Verfügbarkeit von MATLAB auf einer großen Anzahl von Rechnerplattformen und die im HTML-Format vorliegende Dokumentation ist ein Werkzeug entstanden, das auf nahezu allen Rechnerplattformen direkt einsetzbar ist. Die hohe Anzahl von weiteren Toolboxen, die in MATLAB zur Verfügung stehen, erleichtern einerseits die Anwendung der GEATbx, andererseits können andere Problembe-reiche von der durch die GEATbx zur Verfügung gestellten Funktionalität profitieren.

Sobald neue theoretische Erkenntnisse bekannt werden, welche die Arbeit und Funktionalität der GEATbx betreffen, werden diese in die GEATbx einbezogen. Informationen darüber werden auf der Webseite der GEATbx bekanntgegeben (<http://www.geatbx.com/>). Spezielle Anfragen zur Toolbox können per Email gestellt werden: support@geatbx.com.

## 7 Kombination von Operatoren zu Evolutionären Algorithmen

In den bisherigen Kapiteln wurden einzelne Verfahren und Operatoren beschrieben. In diesem Kapitel wird erläutert, wie aus diesen Operatoren vollständige Optimierungsalgorithmen zusammengestellt werden können. Jeder dieser Optimierungsalgorithmen stellt einen Evolutionären Algorithmus dar<sup>1</sup>.

Da es eine große Menge unterschiedlicher Problemklassen gibt und für jede Problemklasse ein anderes Verfahren bzw. eine andere Variante am besten geeignet ist, sollen hier einige häufig gültige Empfehlungen gegeben werden. Diese Empfehlungen beziehen sich auf die Erfahrungen des Autors aus der Optimierung verschiedener Probleme sowie Ergebnissen anderer Autoren aus der Literatur.

Durch dieses Vorgehen erhält der Nutzer Einblicke in die kombiniert spezifischen Eigenschaften der Operatoren und die Abhängigkeiten zwischen den Operatoren. Der Anwender kann besser verstehen, wie diese Eigenschaften und Abhängigkeiten in ihrem Zusammenwirken einen Optimierungsalgorithmus mit bestimmten Eigenschaften definieren. Ausgehend davon ist es leichter, die vorgeschlagenen Algorithmen an neue Problembereiche anzupassen. Außerdem kann besser eingeschätzt werden, warum ein Algorithmus bei einem bestimmten Problem Schwierigkeiten hat oder versagt.

Neben der Wahl eines Optimierungsalgorithmus (Kombination von Operatoren) lässt sich das Verhalten der Optimierungsalgorithmen durch eine Anzahl von Optionen (auch als Parameter bezeichnet) von außen steuern. Da sich viele Optionen eines Optimierungsalgorithmus nicht genau festlegen lassen, werden hier meist Bereiche für deren Werte bzw. eine Abhängigkeit der Werte von anderen Größen angegeben. Dies ermöglicht eine leichte Anpassung auf unterschiedliche Problemgrößen und lässt besser erkennen, welche wechselseitigen Abhängigkeiten beachtet werden müssen.

Für einige Verfahren und Operatoren kann man die Optionen relativ unabhängig vom zu lösenden Problem angeben. Diese Verfahren und die zugehörigen Optionen werden in Abschn. 7.1 erläutert und beschrieben.

Für die Anwendung auf spezielle Problemklassen werden in den Abschn. 7.2-7.5 spezifische Operatoren und deren Optionen zusammengestellt.

---

<sup>1</sup> Damit wird klar, daß es nicht „den Evolutionären Algorithmus“ gibt. Vielmehr definiert sich jeder Evolutionäre Algorithmus über die verwendeten Operatoren und deren Optionen. Ohne diese (vollständigen) Angaben lässt sich nicht nachvollziehen, wie ein Problem gelöst wurde.

Die speziellen Problemklassen lassen sich in die folgenden Gruppen mit ihren entsprechenden Varianten einteilen:

- Parameteroptimierung (z.B. Parameteridentifikation) mit weiterer Unterscheidung entsprechend der Repräsentation der Variablen:
  - global orientierte Parameteroptimierung reeller bzw. ganzzahliger Variablen, Abschn. 7.2,
  - lokal orientierte Parameteroptimierung reeller Variablen, Abschn. 7.3,
  - Parameteroptimierung binärer Variablen, Abschn. 7.4,
- kombinatorische Optimierung, Abschn. 7.5, z.B.:
  - *travelling sales person TSP* (Handlungsreisender), *production scheduling* (Produktionsreihenfolge, z.B. *job shop scheduling JSS*).

Bei der Beschreibung der Optimierungsalgorithmen für die einzelnen Problemklassen werden nur die speziellen Operatoren mit ihren Optionen erläutert. Ansonsten kommen die in Abschn. 7.1 beschriebenen Verfahren mit ihren Optionen zur Anwendung.

Die in diesem Kapitel vorgeschlagenen Werte für die einzelnen Parameter/Optionen können natürlich verändert werden. Allerdings sollte der Anwender dann wissen, wie diese Änderungen sich in etwa auswirken und ob sie Änderungen an anderen Parametern bedingen. Dies gelingt meist erst nach einer etwas längeren Einarbeitung sowie einer Auseinandersetzung mit der Arbeitsweise der einzelnen Operatoren und der Funktion und den kombinierten Auswirkungen der Parameter.

Für eine ausführlichere Erläuterung der Bedeutung der einzelnen Operatoren und ihrer Parameter sei auf Kap. 3 und 4 verwiesen sowie die Erläuterungen zu den Algorithmen in [GEATbx].

## 7.1 Allgemein einstellbare Verfahren und Operatoren

In diesem Abschnitt werden Verfahren und Operatoren beschrieben, für welche die Optionen relativ unabhängig vom zu lösenden Problem eingestellt werden können. Mit diesen Einstellungen ergibt sich eine robuste Arbeitsweise für viele zu lösende Probleme.

### 7.1.1 Fitneßzuweisung und Selektion

Wie in Unterabschn. 3.1.2 erläutert, sollte für die Fitneßzuweisung immer ein reihenfolgebasiertes Verfahren (*ranking*) eingesetzt werden. Für die Selektion können dann alle in Abschn. 3.2 vorgestellten Verfahren eingesetzt werden. Entscheidend ist weniger das verwendete Selektionsverfahren, sondern die Größe der entsprechenden Option – des Selektionsdruckes. Ein Vergleich der Verfahren untereinander und eine Behandlung ihrer Unterschiede wurde in Unterabschn. 3.2.6 vorgenommen.

Gute Erfahrungen wurden mit Werten des Selektionsdruckes im Bereich von 1,5 bis 2 gemacht. Nur bei sehr großen Populationen (über 500 Individuen) sollte ein höherer Selektionsdruck zum Einsatz kommen.

Einen größeren Einfluß auf den Verlauf der Optimierung hat die Option *generation gap* (Generationslücke). Damit wird angegeben, wieviele Nachkommen im Vergleich zur Anzahl der Individuen in der Population produziert werden. Durch ein *generation gap* von kleiner als 1 wird festgelegt, daß weniger Nachkommen produziert werden, als Individuen in der Population enthalten sind. Dadurch überleben einige der Eltern in die folgende Generation, in diesem Falle die besten bisherigen Individuen. Dies entspricht einer *elitest* Strategie. Bei einem *generation gap* von genau 1 werden genau so viele Nachkommen wie Eltern produziert und die Nachkommen ersetzen alle Eltern.<sup>2</sup> Durch ein *generation gap* etwas kleiner als 1 (0,8–0,99) wird sichergestellt, daß einmal gefundene sehr gute Lösungen erst durch noch bessere Lösungen ersetzt werden.

### 7.1.2 Anwendung verschiedener Strategien und Konkurrenz zwischen Unterpopulationen

Wenn im folgenden mehrere Verfahren bzw. Varianten von Operatoren für eine Problemklasse beschrieben werden, dann können diese zusammen in einem Optimierungslauf eingesetzt werden. Dies geschieht durch die *Anwendung verschiedener Strategien*, wie sie in Abschn. 4.5 beschrieben wird. Diese Anwendung verschiedener Strategien erlaubt den gleichzeitigen Einsatz unterschiedlicher Strategien. Dabei kann untersucht werden, welche dieser Strategien für das zu lösende Problem erfolgreich sind. In nachfolgenden Optimierungen reicht es dann aus, sich auf den Einsatz dieser erfolgreichen Strategien zu beschränken.

Ein Beispiel dafür ist die gleichzeitige Anwendung mehrerer Mutationsoperatoren bzw. diese mit unterschiedlichen Parametern arbeiten zu lassen. Dadurch werden gleichzeitig verschiedene Suchstrategien angewendet. Dies führt sehr oft zu besseren Ergebnissen. Es können z.B. drei Unterpopulationen verwendet werden, deren Parameter sich nur im Mutationsbereich (groß, mittel, klein) unterscheiden (Mutationsbereich – s. Unterabschn. 3.4.1). Damit wird gleichzeitig eine grobe, eine mittlere und eine feine Suche durchgeführt. Die grobe Suche ist meist am Anfang erfolgreich und die feine Suche meist am Ende eines Laufs.

Durch den zusätzlichen Einsatz von *Konkurrenz zwischen den Unterpopulationen*, Abschn. 4.6, wird gleichzeitig eine effiziente Verteilung der Rechenressourcen zwischen den Strategien zugunsten erfolgreicher Strategien erreicht. Dies macht sich besonders dann bemerkbar, wenn verschiedene Strategien zu unterschiedlichen Zeiten eines Optimierungslaufes erfolgreich sind. Die zum jeweiligen Zeitpunkt erfolgreichsten Strategien erhalten mehr Ressourcen zugewiesen und können damit verstärkt nach besseren Lösungen suchen. Sobald eine andere Strategie und damit Unterpopulation erfolgreicher wird, erhält diese mehr

---

<sup>2</sup> Durch den Parameter Wiedereinfügerate kann angegeben werden, daß nicht alle produzierten Nachkommen in die Population eingefügt werden. Dies wird aber nur in speziellen Fällen angewendet, s. Abschn. 3.5 für weitere Erläuterungen.

Ressourcen (in diesem Falle mehr Individuen) zugewiesen und wird verstkt in der Suche bercksichtigt.

Diese beiden Konzepte, *Anwendung verschiedener Strategien* und die Erweiterung zur *Konkurrenz zwischen den Strategien*, knnen universell eingesetzt werden. Sie bewren sich vor allem bei der Suche nach geeigneten Verfahren und Operatoren fr die Lsung neuer und in ihrem Verhalten noch nicht so gut bekannter Systeme. Durch deren Einsatz werden vor allem in der Anfangsphase der Arbeit an einem System schneller bessere Ergebnisse erreicht, als wenn nacheinander verschiedene Strategien ausprobiert werden. Außerdem kann es durchaus vorkommen, daß eine Strategie allein gar nicht zum Erfolg fhrt, die miteinander laufenden Strategien aber in relativ kurzer Zeit erfolgreich sind.

### 7.1.3 Regionales Populationsmodell (Migration zwischen Unterpopulationen)

Beide Methoden (*Anwendung verschiedener Strategien* und die Erweiterung zur *Konkurrenz zwischen den Strategien*) basieren auf einer Unterteilung der Population in Unterpopulationen, dem *regionalen Populationsmodell* (s. Abschn. 4.4). Die Entwicklung findet fr eine gewisse Isolationszeit getrennt in den Unterpopulationen statt. Von Zeit zu Zeit wird ein Austausch von (guten) Individuen zwischen den Unterpopulationen vorgenommen. Dadurch wird erreicht, daß die in einer Unterpopulation gefundenen Informationen auch in die anderen Unterpopulationen gelangen.

Bei einer Anwendung des regionalen Modells wird meist mit einer Migrationszeit von 20 (bis zu 40) Generationen gearbeitet, abhig davon, wie lange ein Lauf 脚berhaupt bentigt, um zu Ergebnissen zu gelangen. Wenn mit 脚ber 1000 Generationen gearbeitet wird, dann kann die Migrationszeit auf alle 50 Generationen heraufgesetzt werden, bei kurzen Lufen 脚ber 50 Generationen ist eine Migrationszeit von 5-10 Generationen besser. Die Migrationsrate (Anteil der auszutauschenden Population) wird meist auf 10% gesetzt, die Migrationsstruktur auf unbeschrkt (Austausch zwischen allen Unterpopulationen). Detaillierte Hinweise zu den Parametern werden in Abschn. 4.4 gegeben.

### 7.1.4 Zusammenfassung allgemein einstellbare Verfahren

Mit den Erluterungen in diesem Abschnitt konnte gezeigt werden, daß sich viele Optionen eines Evolutionn Algorithmus auf sehr vernnftige und robuste Werte einstellen lassen, ohne daß sie fr jedes Problem neu angepat werden mssen.

An dieser Stelle soll eine kurze Zusammenfassung der bisher erlerten Verfahren und einer robusten Parametrisierung gegeben werden:

- **Fitnezuweisung und Selektion:**

Fitnezuweisung durch lineares Ranking mit Selektionsdruck von 1,5-2, *generation gap* von 0,95, *stochastic universal sampling* als Selektionsverfahren,

- **Migration:** (bei mehreren Unterpopulationen, regionales Modell)  
Migrationszeit von 20 Generationen, Migrationsrate von 10%, Migrationsstruktur: vollständiges Netz,
- **Anwendung verschiedener Strategien:**  
Definition unterschiedlicher Verfahren bzw. Optionen für die einzelnen Unterpopulationen (z.B. unterschiedliche Mutationsbereiche),
- **Konkurrenz zwischen Unterpopulationen:**  
Konkurrenzintervall von 4-10 Generationen, Konkurrenzrate von 10%, Unterpopulationsminimum bei 10%-20% der anfänglichen Größe der Unterpopulation.

Damit sind die allgemein anwendbaren Erläuterungen und Hinweise gegeben. Die Angaben in den folgenden Abschnitten beziehen sich auf spezielle Problemklassen.

## 7.2 Global orientierte Parameteroptimierung

Ein großer Teil der Anwendungsprobleme in der ingenieurtechnischen Arbeit umfaßt Parameteroptimierungsprobleme mit reellen und ganzzahligen Variablen, bei denen meist keine (starke) Korrelation zwischen den Variablen vorliegt (siehe Glossar: korrelierte Variablen). In diesem Abschnitt werden Evolutionäre Algorithmen zusammengestellt, die speziell für die global orientierte Optimierung dieser Probleme geeignet sind. Unter global orientierter Suche ist an dieser Stelle gemeint, daß keine Annahmen über die Art der Zielfunktion vorgenommen werden und hauptsächlich entlang der Koordinatenachsen gesucht wird. Die Variablen werden jeweils für sich verändert.

### **Rekombination**

Es können fast alle Rekombinationsoperatoren verwendet werden (auch die für binäre Variablen). Am günstigsten sind die Diskrete Rekombination (*redis*) und (seltener) die Linien-Rekombination (*reclin*). Für ganzzahlige Variable können die Linien- und die Raum-Rekombination nur eingeschränkt benutzt werden<sup>3</sup>. Die Rekombinationsrate sollte in der Regel auf 1 gesetzt sein.

---

<sup>3</sup> Die Linien- und Raum-Rekombination führen zu einer Veränderung des Variablenwertes (und nicht nur zu einen Austausch von Variablenwerten, wie dies bei der Diskreten Rekombination geschieht). Wenn diese Operatoren auf ganzzahlige Variablen angewendet werden, sind die Variablen nach der Rekombination nicht mehr ganzzahlig. Deshalb muß für ganzzahlige Variablen eine angepaßte Variante dieser Rekombinationsoperatoren verwendet werden, die eine entsprechende Überprüfung der Variablenwerte durchführt und gegebenenfalls ein "Runden" der Variablen vornimmt.

### **Mutation**

Die zur Verfügung stehenden Mutationsooperatoren für reelle und ganzzahlige Variablen (***mutreal*** und ***mutint***), Unterabschn. 3.4.1, können durch entsprechende Parametrisierung ein sehr unterschiedliches Verhalten zeigen. Deshalb gibt es nur einen Operator je Variablentyp, wobei beide (fast) identisch arbeiten. Eine Anpassung an den Einsatzfall (grobe Mutation bis sehr feine Mutation) erfolgt durch entsprechende Optionen. Insbesondere kann eine stufenlose Einstellung zwischen einer grob und einer fein ausgerichteten Mutation (und damit auch der entsprechenden Suchstrategie) vorgenommen werden.

Für eine grobe Mutation wird ein Mutationsbereich von 0,2-0,05 verwendet, für eine feine Suche Werte bis zu  $10^{-8}$  hinab und kleiner. Der Mutationsbereich gibt den Wert des größten Mutationsschrittes im Verhältnis zum Definitionsbereich des jeweiligen Parameters an.

Mit einer Mutationspräzision von 16 (Bereich von 8-20) wird zusätzlich eine vernünftige Verteilung der Mutationsschritte erreicht (meist kleine Mutationsschritte, nur ab und zu große Schritte).

Eine mögliche Parametrisierung der Mutation für reelle Variablen unter Verwendung von 4 verschiedenen Strategien könnte Mutationsbereiche von [0,1, 0,03, 0,001, 0,0003] enthalten, alle mit einer Mutationspräzision von 10. Damit wird ein großes Abrastern des Suchraumes mit einer feinen Suche verbunden. Dieses Vorgehen hat sich bei vielen praktischen Anwendungen bewährt.

Die Mutationsrate sollte auf (1/Anzahl der Variablen) gesetzt sein und nur in Ausnahmefällen bzw. bei einem Selektionsdruck größer als 2 vergrößert werden.

Die Parametrisierung des Mutationsooperators hat in den meisten Fällen den größten Einfluß auf die Leistungsfähigkeit und damit den Verlauf der Optimierung. Deshalb sollte die Suche nach einem leistungsfähigen Algorithmus zur Lösung des aktuellen Problems auf diesen Operator konzentriert werden. Dabei kann durch die Verwendung verschiedener Mutationsbereiche in den einzelnen Unterpopulationen (damit Anwendung verschiedener Strategien) schnell eine Anpassung an das Problem vorgenommen werden.

Implementierung in GEATbx: ***tbxreal*** bzw. ***tbxit***.

## **7.3 Lokal orientierte Parameteroptimierung**

In diesem Abschnitt werden Evolutionäre Algorithmen zusammengestellt, die speziell für die lokal orientierte Optimierung reeller Variablen geeignet sind. Dies umfaßt im allgemeinen Anwendungsprobleme, bei denen eine starke Korrelation zwischen den (reellen) Variablen vorliegt. Deshalb müssen die Variablen gleichzeitig (koordiniert) verändert werden, die Suche muß in beliebigen Richtungen im Suchraum möglich sein.

Um dies zu erreichen, ist ein „Erlernen“ erfolgversprechender Suchrichtungen notwendig. Für solche Probleme stehen spezielle Mutationsooperatoren zur Verfügung, die aus dem Bereich der Evolutionsstrategien stammen, Unterabschn. 3.4.2. Diese versuchen, die Richtung einer Verbesserung zu lernen und ermöglichen dadurch ein zielgerichtetes Suchen.

Allerdings ist die Anwendung dieser Operatoren auf glatte Funktionen und niedrigdimensionale Probleme beschränkt (je größer ein Problem ist, um so länger dauert das Erlernen einer Richtung). Wenn eine Zielfunktion verrauscht ist, bleiben diese Evolutionären Algorithmen mit hoher Wahrscheinlichkeit im nächsten kleinen Minimum stecken. Genauso haben diese Algorithmen so gut wie keine Chance, aus einem lokalen Minimum wieder herauszukommen. Damit sind sie für multi-modale Probleme ungeeignet. Für weitergehende Hinweise zur Anwendung dieser Strategien sei auf [Ost97] und [Han98] verwiesen.

### ***Rekombination***

Bei diesen Verfahren findet (meistens) keine Rekombination statt.

### ***Mutation***

Es stehen spezielle Mutationsoperatoren zur Verfügung, die aus dem Bereich der Evolutionsstrategien stammen, Unterabschn. 3.4.2. Bei diesen Operatoren wird nur die Anfangsschrittweite vorgegeben, danach führen diese Funktionen eine selbständige Adaption der Schrittweiten bzw. einer Suchrichtung durch.

Der Mutationsbereich ist nur eine Vorgabe für den Bereich der Anfangsschrittweite. Gute Erfahrungen wurden mit Werten von  $10^{-3}$  bis  $10^{-7}$  (bezogen auf den Definitionsbereich der einzelnen Variablen, wie dies in der GEATbx implementiert ist) gesammelt. Die Festlegung einer vernünftigen Anfangsschrittweite ist problemspezifisch und nicht so einfach.

Auch bei diesen Operatoren ist es möglich, mit verschiedenen Strategien zu arbeiten. Diese würden sich im Mutationsbereich unterscheiden (z.B. groß, mittel, klein). Jede der Strategien (Unterpopulationen) startet dadurch mit einem anderen Bereich der Anfangsschrittweite. Sobald allerdings die Adaption der Schrittweiten stattfand, haben die unterschiedlichen Mutationsbereiche keinen Einfluß mehr. Dies dient demnach nur der Ermittlung vernünftiger bzw. erfolgreicher Anfangsschrittweiten.

### ***Fitneßzuweisung und Selektion***

Es ist zu beachten, daß diese Verfahren zur lokal orientierten Optimierung nur mit wenigen Individuen arbeiten und daß viele Nachkommen erzeugt werden, von denen nur die Besten in die Population eingefügt werden.

Um dies zu erreichen, stehen zwei Möglichkeiten zur Auswahl. Einerseits kann die Population wenige Individuen enthalten, die jeweils viele Nachkommen produzieren. Nur die besten dieser Nachkommen ersetzen die Eltern und bilden die neue Population. Dabei wären die folgenden Parameter für Populationsgröße und Selektion zu verwenden:

- Populationsgröße: 1-5 Individuen,
- *generation gap*: 3-10 (Anzahl der Nachkommen pro Elter),
- Selektionsdruck: 1 (kein Selektionsdruck).

Anderseits wird mit einer größeren Population gearbeitet, bei der nur die allerbesten Individuen Nachkommen produzieren. Hier ersetzen (fast) alle Nachkommen die Eltern und bilden die neue Population:

- Populationsgröße: 5-20 Individuen,
- *generation gap*: 1 (so viele Nachkommen wie Eltern produzieren),
- Selektion: Abschneideselektion (*truncation selection*) mit Selektionsdruck von 3-10 (nur die allerbesten Individuen werden ausgewählt und produzieren jeweils mehrere Nachkommen).

Die erste Variante orientiert sich an der ursprünglichen „Definition“ dieser Verfahren und hat ihre volle Berechtigung. Bei der zweiten Variante gibt es allerdings einige kleine Vorteile, die sich insbesondere bei der praktischen Anwendung gezeigt haben. Zum ersten können alle produzierten Individuen (Nachkommen) in die (spätere) Auswertung einbezogen werden, da sie zumindest für eine Generation in die Population aufgenommen wurden (bei der ersten Variante wird nur ein kleiner Teil der produzierten Nachkommen überhaupt in die Population aufgenommen). Weiterhin kann bei der zweiten Variante sehr einfach eine *elitest* Selektion definiert werden, bei der das bisher beste Individuum immer überlebt. Dazu muß einfach ein *generation gap* von etwas kleiner als 1 (z.B. 0,99) definiert werden.

Implementierung in GEATbx: ***tboxes1*** und ***tboxes2***.

## 7.4 Parameteroptimierung binärer Variablen

Die Parameteroptimierung binärer Variablen findet vor allem bei den klassischen Genetischen Algorithmen Anwendung. Bei diesen werden die reellen oder ganzzahligen Variablen durch Diskretisierung in eine größere Anzahl binärer Variablen umgewandelt. Der Genetische Algorithmus arbeitet dann auf diesen binären Variablen. Vor der Berechnung der Zielfunktion muß wieder eine Umwandlung der binären Variablen in das ursprüngliche Format stattfinden<sup>4</sup>.

Da diese Genetischen Algorithmen immer noch im Einsatz sind, stehen eine Anzahl von entsprechenden Operatoren zur Verfügung. Außerdem gibt es einige reale Probleme, die direkt binäre Variablen verwenden.

### ***Rekombination***

Es können alle Rekombinationsoperatoren für binäre Variablen verwendet werden (*recsprs*, *recdp*, *recshrs*, usw., aber auch *recdis*). Günstig sind die Diskrete Rekombination (*recdis*) und die Operatoren mit *reduced surrogate* (\**rs*). Die Rekombinationsrate wird in der Literatur meist auf 0,7 gesetzt.

---

<sup>4</sup> Diese Umwandlung bzw. Dekodierung der Variablen aus der binären Repräsentation wird in der GEATbx automatisch vorgenommen.

### **Mutation**

Für die Mutation binärer Variablen steht ein Operator (***mutbin***) zur Verfügung. Für diesen Operator kann nur die Mutationsrate eingestellt werden, andere Einflußmöglichkeiten kann es für binäre Variablen nicht geben. Die Mutationsrate wird meist auf (1/Anzahl der Variablen) gesetzt und nur in Ausnahmefällen vergrößert.

Implementierung in GEATbx: ***tboxbin***.

## **7.5 Kombinatorische Optimierung**

Die bisher zusammengestellten Evolutionären Algorithmen sind alle für die Parameteroptimierung gedacht. Daneben existiert aber eine weitere große Problemklasse, die der kombinatorischen Optimierung. Bekannte Problemklassen innerhalb der kombinatorischen Optimierung sind *travelling sales person TSP* (Handlungsreisender), *quadratic assignment problem QAP* und *production scheduling* (Produktionsreihenfolge, z.B. *job shop scheduling JSS*).

Einige Rekombinations- und Mutationsoperatoren für diese Problemklasse wurden in Kap. 3 vorgestellt. Genau wie in den anderen Abschnitten dieses Kapitels sollen hier Evolutionäre Algorithmen für ausgewählte Problemklassen zusammengestellt werden.

An dieser Stelle müssen jedoch zwei Einschränkungen gemacht werden. In der kombinatorischen Optimierung gibt es extrem viele verschiedene Anwendungsbereiche, die alle ein etwas anderes Vorgehen verlangen. Und gerade bei der Lösung sehr großer Probleme müssen spezielle Verfahren und Operatoren zur Anwendung gelangen, um mit vertretbarem Aufwand zu einem Ergebnis zu kommen. Dies kann im Rahmen dieses Abschnitts nicht umfassend behandelt werden.

Es sollen einige Hinweise gegeben werden, die sich in der bisherigen Arbeit bewährt haben. Diese zeigen, wie sich die „neue“ Anwendung kombinatorische Optimierung in die bisher verwendete Struktur Evolutionärer Algorithmen einordnet und entsprechend mit den spezifischen Operatoren ausgefüllt wird. Die Aussagen zur Verwendung und Parametrisierung der allgemein einstellbaren Verfahren und Operatoren aus Abschn. 7.1 lassen sich hier direkt anwenden.

### **Rekombination**

Im Rahmen der bisherigen Arbeit wurde mit drei Rekombinationsoperatoren für kombinatorische Probleme gearbeitet: *maximal preservative crossover MPX (recmpx)*, *generalized position recombination GPX (recgpx)* und *edge recombination EX (recedge)*. Diese sind speziell für die Bearbeitung von Reihenfolgeproblemen entworfen. Alle Operatoren sorgen dafür, daß die durch die Rekombination erzeugten Individuen gültig sind.

Der grundlegende Unterschied zwischen den Operatoren liegt in der Information, die sie bei der Rekombination bewahren. Bei der kombinatorischen Opti-

mierung sind drei Eigenschaften wichtig: die Nachbarschaftsbeziehung zwischen den Variablen (relative Anordnung), die absolute Anordnung der Variablen bzw. die absolute Position der Variablen. Beim TSP ist die relative Anordnung entscheidend, die Information über die absolute Position dagegen nicht. Beim QAP ist die absolute Position der Variablen wichtig und nicht die Nachbarschaft zu anderen Variablen. Beim JSS wiederum ist die absolute Anordnung wichtig. Das zu lösende Problem muß dahingehend untersucht werden, welche Informationen bewahrt werden sollen bzw. zu welcher der oben angeführten Problemklassen es gehört.

*GPX* bewahrt besonders die absolute Anordnung der Variablen, zeigt aber zusätzlich ein gutes Verhalten beim Bewahren der relativen Anordnung. *EX* ist sehr gut geeignet zur Bewahrung der relativen Anordnung der Variablen, aber schlecht bei der Bewahrung der absoluten Anordnung. *MPX* ist am besten zur Bewahrung der relativen Anordnung der Variablen geeignet, wenn mit lokal optimierten Individuen gearbeitet wird.

Welcher dieser Operatoren (oder weiterer Rekombinationsoperatoren für kombinatorische Probleme) am besten für die Lösung eines Problems geeignet ist, kann außerdem durch die Anwendung verschiedener Strategien recht einfach ermittelt werden. Die Unterpopulationen verwenden jeweils einen der Rekombinationsoperatoren. Die beste Unterpopulation gibt dann Hinweise darauf, welcher der verwendeten Operatoren für weitere Optimierungsläufe eingesetzt werden sollte.

Die Rekombinationsrate wird meist auf Werte von 1 oder etwas kleiner gesetzt.

### **Mutation**

Zur Mutation stehen mehrere Operatoren zur Verfügung (*swap* / *insert* / *reverse* / *scramble mutation*). Diese Mutationsoperatoren können durch die Vorgabe von Mutationsbereich und Mutationspräzision auf verschiedenen großen Mutationsschritte eingestellt werden. Dies führt zu einer Funktionalität im Sinne von großen und kleinen Änderungen. Durch den Mutationsbereich wird der maximale Abstand vorgegeben, wie weit die zu verändernden Variablen innerhalb des Individuums voneinander entfernt sind. Die Mutationspräzision sorgt für eine Verteilung der Mutationsschritte in diesem Bereich. Damit kann eine Verteilung zwischen großen und kleinen Mutationsschritten vorgenommen werden.

Über die Mutationsrate wird gesteuert, wieviele Veränderungen von Variablen pro Individuum durchgeführt werden. Mit einer Mutationsrate von (1/Anzahl der Variablen), die z.B. einem Austausch von zwei Variablen entspricht, sollte in den meisten Fällen ein gutes Verhalten erreicht werden.

Implementierung in GEATbx: *tbxtsp*.

## 7.6 Parameteroptimierung von Variablen verschiedener Repräsentation

Zum Abschluß dieses Kapitels soll noch auf ein Problem eingegangen werden, daß in der Praxis öfters auftritt und die Abschnitte zur Parameteroptimierung in diesem Kapitel verbindet. Bisher wurde davon ausgegangen, daß alle Variablen eines zu lösenden Problems in der gleichen Repräsentation vorliegen. Dies ist aber nicht immer so. Bei der Lösung einiger Praxisprobleme müssen gleichzeitig Variablen unterschiedlichen Typs behandelt werden. Dies bedeutet, daß einige der Variablen eines Individuums reell, andere ganzzahlig bzw. binär sind.

Wie schon öfters angesprochen, sind für jeden Datentyp andere Operatoren am besten geeignet. Außerdem müssen für jeden Datentyp gewisse Eigenheiten beachtet werden. Es muß ein Weg gefunden werden, wie trotz der unterschiedlichen Anforderungen alle Variablen gleichzeitig optimiert werden können.

Für die Bearbeitung von Problemen mit Variablen mehrerer Repräsentationen gibt es zwei Vorgehensweisen:

- Einmal können auf die Variablen eines Typs die jeweils entsprechenden Operatoren angewandt werden (wie dies in den vorhergehenden Abschnitten erläutert wurde). Wenn reelle und binäre Variablen vorliegen, werden innerhalb des Evolutionären Algorithmus zwei Rekombinations- und zwei Mutationsoperatoren mit den jeweiligen Variablen der Individuen aufgerufen. Dies führt neben einem erhöhten Aufwand innerhalb des Evolutionären Algorithmus vor allem zu einem deutlich erhöhten Aufwand in der Parametrisierung des Evolutionären Algorithmus, worunter vor allem die Übersichtlichkeit leidet. Im Endeffekt ist dies eine Frage der (aufwendigen) Implementierung und wird hier nicht weiter betrachtet.
- In der zweiten Variante werden die unterschiedlichen Variablentypen auf einen Typ zurückgeführt und mit diesem arbeitet der Evolutionäre Algorithmus. Dies scheint auf den ersten Blick der beschwerlichere Weg zu sein, hat sich in der praktischen Anwendung aber als deutlich besser zu handhaben und einfach in der Umsetzung erwiesen.

Wenn alle Variablentypen auf eine Repräsentation zurückgeführt werden, müssen einmal die verschiedenen Kombinationen von vorliegenden Typen betrachtet werden und in welche Repräsentation umgewandelt wird. Prinzipiell ist die Umwandlung aller Variablen in jede Repräsentation möglich. Einige der Vor- und Nachteile sowie die zu beachtenden Dinge werden im folgenden erläutert.

### Ganzzahlige und binäre Variablen

Die Kombination von ganzzahligen und binären Variablen soll hier als erstes betrachtet werden, da sie sehr einfach und ohne Einschränkungen möglich ist. Eine ganzzahlige Variable in den Grenzen  $[0, 1]$  entspricht einer binären Variable. Die Evolutionären Operatoren für ganzzahlige Variablen arbeiten mit diesen binären Variablen gut. Die Einstellungen für ganzzahlige Variable werden verwendet, Abschn. 7.2.

### **Verwendung der ganzzahligen Repräsentation**

Wenn reelle und ganzzahlige/binäre Variablen vorliegen, kann die ganzzahlige Repräsentation verwendet werden. Dabei gibt es keine Einschränkung für binäre und integer Variablen. Diese können in ihrer ursprünglichen Repräsentation belassen werden.

Für die reellen Variablen muß dagegen eine Umwandlung bzw. Anpassung vorgenommen werden. Wenn der Definitionsbereich der reellen Variablen sehr groß ist und die Beschränkung auf ganzzahlige Werte keine Probleme bereitet, ist dies der Weg der Wahl. Wenn allerdings eine höhere Genauigkeit der Werte der reellen Variablen als ganzzahlige Werte benötigt wird, dann muß eine Diskretisierung des reellen Definitionsbereiches vorgenommen werden. Diese diskreten Werte können dann auf ganzzahlige Werte abgebildet werden, die in der Optimierung verwendet werden.

Diese Umwandlung bzw. Anpassung kann durch eine relativ einfache Umrechnung der entsprechenden Variablen in der Zielfunktion des Problems erreicht werden. Dabei muß zuerst festgelegt werden, mit welcher Genauigkeit jede der reellen Variablen zu optimieren ist. Diese Genauigkeit kann durch einen Skalierungsfaktor für jede Variable festgelegt werden. Ein Skalierungsfaktor von 10 bedeutet, daß diese Variable mit einer minimalen Auflösung von 0,1 ( $1/_{10}$ ) behandelt wird. Entsprechend bedeutet ein Skalierungsfaktor von 200 eine minimale Auflösung von 0,005 ( $1/_{200}$ ).

Anschließend wird auf dieser Grundlage eine Vergrößerung des Definitionsbereiches der Variablen (für die Optimierung) um diesen Genauigkeitsfaktor oder Skalierungsfaktor vorgenommen. Mit diesem Definitionsbereich und ganzzahligen Variablen arbeitet der Evolutionäre Algorithmus und die vorgegebenen Operatoren. Vor jeder Anwendung der Zielfunktion werden die ganzzahligen Variablen mit dem festgelegten Skalierungsfaktor umgerechnet und liegen dann als reelle Variablen in der gewünschten Auflösung vor.

Das beschriebene Vorgehen soll an zwei Beispielen verdeutlicht werden. Im Ersten wird eine reelle Variable verwendet, die einen Definitionsbereich von  $[-1, +1]$  hat. Für diese Variable ist eine Genauigkeit von 0,01 gewünscht. Dafür werden die ursprünglichen Grenzen des Definitionsbereiches dieser Variable mit dem Skalierungsfaktor  $1/_{0,01}=100$  multipliziert. In der Optimierung wird diese Variable als ganzzahlig in den Grenzen  $[-100, +100]$  behandelt. Vor der Anwendung der Zielfunktion wird diese Variable durch den Skalierungsfaktor 100 dividiert. Die zu verwendende reelle Variable liegt mit der gewünschten Genauigkeit wieder vor.

Im zweiten Beispiel liegen 3 Variablen vor, die erste und dritte als reelle, die zweite als ganzzahlige Variable. Die erste Variable soll eine minimale Auflösung von 0,2 haben, die dritte von 0,005. Die Skalierungsfaktoren für alle 3 Variablen müssen damit zu  $[5, 1, 200]$  festgelegt werden. Durch Multiplikation mit diesen Skalierungsfaktoren wird der Definitionsbereich für diese Variablen vergrößert (bzw. beibehalten für die ganzzahlige Variable). Vor der Auswertung der Zielfunktion erfolgt mittels Division durch die Skalierungsfaktoren eine Umwandlung der ganzzahligen Variablenwerte in die reellen Variablenwerte gewünschter Auflösung im problemspezifischen Definitionsbereich.

Dieses Vorgehen ist in der Anwendung sehr flexibel. Auf der einen Seite kann genau die erwünschte Auflösung der Variablen beeinflußt werden. Auf der anderen Seite stehen weiterhin alle Möglichkeiten des Einsatzes der leistungsfähigen evolutionären Operatoren für ganzzahlige Variablen zur Verfügung.

Als Nachteil ist nur zu nennen, daß für die vormals reellen Variablen der direkte Zusammenhang zwischen den Werten der Variablen in der Optimierung und dem problemspezifischen Wert verloren geht. Die Variablenwerte aus der Optimierung müssen erst mit dem Skalierungsfaktor umgerechnet werden.

Dieser Nachteil ist aber zu verschmerzen und wiegt nicht die gute Handhabbarkeit dieser Methode auf. Bei der Lösung von Systemen mit Variablen unterschiedlicher Repräsentation hat sich die Verwendung ganzzahliger Variablen für die Optimierung bewährt und wird hier für die Anwendung empfohlen.

Für die Zusammenstellung von Operatoren und deren Parametrisierung zu Evolutionären Algorithmen gelten die Hinweise, die für die Parameteroptimierung der entsprechenden Repräsentation in den vorherigen Abschnitten gegeben wurden, speziell Abschn. 7.2. Deshalb wird an dieser Stelle nicht weiter darauf eingegangen.

### **Verwendung der binären Repräsentation**

Neben einer Umwandlung reeller Variablen in die ganzzahlige Repräsentation kann auch die binäre Repräsentation als kleinster gemeinsamer Nenner verwendet werden. Dazu werden alle ganzzahligen und reellen Variablen in die binäre Repräsentation umgewandelt und mit dieser arbeitet der Evolutionäre Algorithmus. Dieses Vorgehen entspricht den klassischen Genetischen Algorithmen.

Auch bei dieser Umwandlung muß für jede Variable eine gewünschte Auflösung vorgegeben werden. Bei ganzzahligen Variablen ergibt sich diese von selbst; eine ganzzahlige Variable in den Grenzen [10, 500] kann durch  $\log_2(491) < 9$  binäre Variablen kodiert werden. Bei reellen Variablen gelten die oben gemachten Angaben. Es muß allerdings beachtet werden, daß mit der Basis 2 gearbeitet wird. Für eine reelle Variable, die mit 10 binären Variablen dekodiert wird, bedeutet dies eine Auflösung von etwa 0,001 ( $1/2^{10}$ ) bezogen auf den Definitionsbereich (bei linearer Skalierung).

Wie in Abschn. 7.4 angedeutet, arbeiten heute noch viele Anwender mit dieser Methode. Sie hat historische Bedeutung und Berechtigung. In vielen Anwendungen zeigte sich aber, daß die Verwendung der für ganzzahlige und reelle Variablen vorliegenden evolutionären Operatoren leistungsfähigere Evolutionäre Algorithmen ergeben. Trotzdem wird die Dekodierung von binären Variablen in ganzzahlige (**bin2int**) und reelle Variablen (**bin2real**) in der GEATbx durch die genannten Funktionen unterstützt.

### **Verwendung der reellen Repräsentation**

Prinzipiell ist es auch möglich, mit einer reellen Repräsentation zu arbeiten und eine Beschränkung der Variablenwerte in der Zielfunktion durchzuführen (runden der Variablen auf den nächsten ganzzahligen Wert).

Dies kann allerdings dazu führen, daß eine Mutation der ursprünglich binären oder ganzzahligen Variablen (durch das anschließende Runden) keine Auswirkung hat. Besonders extrem kann dies bei binären Variablen sein, da erst eine Veränderung um durchschnittlich ein Viertel des Definitionsbereiches zu einer Veränderung des binären Variablenwertes führt. Bei ganzzahligen Variablen ist dieses Problem um so geringer, je größer der Definitionsbereich der entsprechenden Variable ist.

Die reelle Repräsentation sollte deshalb nur in Ausnahmefällen bzw. bei der Kombination von ganzzahligen Variablen mit großem Wertebereich (Differenz zwischen unterer und oberer Grenze ist größer als 1000) und reellen Variablen verwendet werden.

## **7.7 Zusammenfassung**

In diesem Kapitel wurden Hinweise für die Zusammenstellung und Parametrisierung leistungsfähiger Evolutionärer Algorithmen gegeben. Dazu wurde eine Unterteilung in häufig auftretende Problemklassen vorgenommen (Parameteroptimierung von reellen, ganzzahligen und binären Variablen sowie Reihenfolgeoptimierung). Den Anfang der Erläuterungen bildeten Hinweise, die für die meisten Problemklassen gleichartig angewendet werden können (Fitneßzuweisung, Selektion, Anwendung des regionalen Populationsmodells mit verschiedenen Strategien, Konkurrenz). Danach wurden die für jede Problemklasse spezifischen Operatoren mit robusten Parametern zusammengestellt.

Für die Lösung von Problemen mit Variablen unterschiedlicher Repräsentation bestehen mehrere Möglichkeiten. Je nach den vorliegenden Variabtentypen wurde auf die Vor- und Nachteile der Umwandlung und die Verwendung der einzelnen Repräsentationen eingegangen. In der Anwendung hat sich die Umwandlung reeller Variablen in die ganzzahlige Repräsentation und deren Verwendung für die Optimierung bewährt und wird daher empfohlen.

Mit den in diesem Kapitel gegebenen Evolutionären Algorithmen sind die Grundlagen für deren Anwendung auf Systeme verschiedener Problemklassen gelegt. Der Anwender kann nach der Analyse seines zu lösenden Systems den entsprechenden Evolutionären Algorithmus auswählen. Eine Anpassung ist nur auf Grund von problemspezifischen Anforderungen notwendig. Bei den in Kap. 8 vorgestellten Anwendungen Evolutionärer Algorithmen auf Praxisprobleme wird deshalb nur auf die wirklich problemspezifischen Einstellungen des verwendeten Evolutionären Algorithmus eingegangen. Die für die jeweilige Problemklasse sich ergebenden Einstellungen wurden bereits hier erläutert.

## 8 Anwendung Evolutionärer Algorithmen auf Praxisprobleme

In den folgenden Abschnitten werden ausgewählte Anwendungen vorgestellt, die einen Einblick in die Spannweite des Einsatzes Evolutionärer Algorithmen zur Lösung von Praxisproblemen geben. Alle Probleme wurden in den vergangenen Jahren mit Hilfe Evolutionärer Algorithmen aus der GEATbx, Kap. 6, bearbeitet und gelöst. Schwerpunkt der Darstellungen ist die Veranschaulichung des zu lösenden Problems, seine Zuordnung in eine der Problemklassen aus Kap. 7 und der beschrittene Lösungsweg bzw. der am Ende verwendete Lösungsansatz. Bei allen größeren Problemen wird Gebrauch von Methoden der problemspezifischen Visualisierung gemacht.

Am Beginn dieses Kapitels wird in Abschn. 8.1 das Vorgehen zur Lösung eines neuen Problems beschrieben. Damit wir der Versuch unternommen, eine schrittweise Anleitung zur Bearbeitung eines neuen Optimierungsproblems zu geben. Dieses Vorgehen erwuchs aus den Erfahrungen des Autors und half bereits anderen Ingenieuren, systematisch zu Ergebnissen bei der Lösung von Optimierungsproblemen zu gelangen. Hauptaugenmerk wird bei diesem schrittweisen Vorgehen auf die Einordnung des Problems, die Aufstellung der Zielfunktion und die Voruntersuchungen zum Systemverhalten bis hin zur Durchführung und Auswertung von Optimierungen gelegt. Die Vorgehensbeschreibung sollte dem nicht so erfahrenen Anwender die ersten Schritte zur Optimierung seiner Probleme deutlich erleichtern.

Der erste Anwendungskomplex sind mehrdimensionale Funktionen, Abschn. 8.2. Diese stellen relativ einfache Probleme dar und werden oft als Testfunktionen verwendet. Sie dienen hier als Einleitung in die Lösung von Problemen. An diesen Funktionen werden einige grundlegende Schritte gezeigt.

Eine in der Praxis oft vorkommende Aufgabe ist die Identifikation der Parameter eines Modells. Als Grundlage liegen für ein reales System Meßdaten der Ein- und Ausgangssignale vor. Außerdem besteht eine ungefähre Vorstellung der Struktur des Systems. Daraus kann ein Strukturmodell des Systems aufgestellt werden. Die Ermittlung der Werte der freien Parameter des Modells ist eine schwierige und langwierige Aufgabe, die mit der richtigen Zielfunktion gut von Evolutionären Algorithmen gelöst werden kann. Mit der Optimierung der Parameter eines Modells der Verbrennung für direkteinspritzende Dieselmotoren wird in Abschn. 8.3 ein Beispiel vorgestellt.

Eng verwandt mit der Parameteridentifikation eines Modells ist die Einstellung der Parameter eines Reglers. Es treten allerdings deutliche Unterschiede in der Formulierung der Zielfunktion auf. Außerdem sind einige regelungstechni-

sche Besonderheiten zu beachten (z.B. Stabilität). In Abschn. 8.4 wird die Reglerdimensionierung für die Querlenkung eines Straßenfahrzeugs vorgestellt.

Zum Abschluß der Beispiele wird in Abschn. 8.5 eine große und komplexe Anwendung vorgestellt, die Optimierung der Steuerung von Klimagrößen in einem Gewächshaus. Damit soll gezeigt werden, daß heute Aufgaben mit Evolutionären Algorithmen gelöst werden können, die bisher als zu groß und komplex eingestuft wurden. Eine umfassende Aufbereitung des Problems ist allerdings notwendig, ein gewisser Aufwand muß in die Voruntersuchung des Systems und die Aufbereitung des zur Verfügung stehenden Wissens gesteckt werden. Einige oft anzuwendende Methoden werden hier erläutert und können auf andere Probleme übertragen werden.

Vergleiche verschiedener Evolutionärer Algorithmen werden bei den hier vorgestellten Anwendungen nur dann gezeigt, wenn dies relevant ist und zusätzliche Einblicke in das zu lösende Problem verschafft. Die Art des zu verwendenden Evolutionären Algorithmus ergibt sich sehr oft aus der Definition des Problems (Kap. 7) bzw. aus den Erkenntnissen, die während der Voruntersuchungen ermittelt wurden (Abschn. 8.1). Die Einbeziehung problemspezifischen Wissens hat in den meisten Fällen einen größeren Einfluß auf den Verlauf der Optimierung, als es durch eine Veränderung der verwendeten Optimierungsverfahren möglich war. Oftmals wird erst dadurch die Lösung der Optimierungsaufgabe in einem vernünftigen Zeitrahmen möglich.

## **8.1 Vorgehen bei der Lösung von Optimierungsaufgaben**

Ein neues Problem muß gelöst werden – wie sollte dies als Ingenieur angegangen werden? Welche Fragestellungen sollten zuerst untersucht, welche Untersuchungen angestellt werden? Dieser Abschnitt unternimmt den Versuch, eine Anleitung zur schrittweisen Bearbeitung eines neuen Optimierungsproblems zu geben. Dabei werden die einzelnen Punkte exemplarisch behandelt. So oft wie möglich wird auf entsprechende Bereiche in den folgenden Abschnitten verwiesen, in denen die allgemeinen Anmerkungen an Beispielen konkretisiert werden.

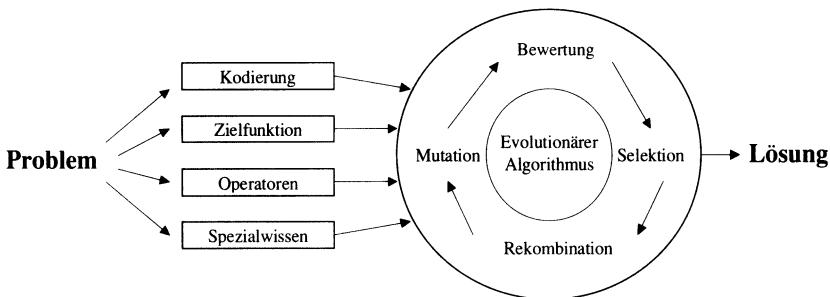
Das in diesem Kapitel beschriebene Vorgehen erwuchs aus den Erfahrungen des Autors bei der Lösung einer größeren Anzahl von realen Optimierungsaufgaben unter Verwendung der GEATbx (Kap. 6, ab S.157 und [GEATbx]). Das hier vorgestellte Vorgehen hat sich auch bei anderen Ingenieuren bewährt und verhalf diesen, systematisch zu Ergebnissen bei der Lösung ihrer komplexen Probleme zu gelangen.

Die Hauptpunkte des hier beschriebenen Vorgehens sind:

- Einordnung des Problems und Aufstellung der Zielfunktion,
- Voruntersuchungen zum Systemverhalten,
- Festlegung des zu verwendenden Optimierungsverfahrens (dieser Punkt wird intensiv in Kap. 7, ab S.171 behandelt),
- Durchführung und Auswertung von Optimierungen.

Diese Punkte sind in Abb. 8-1 in einem Ablaufschema dargestellt. Wenn ein neues Problem gestellt wird, muß zuerst eine Analyse des Problems vorgenommen

werden, aus der sich die Kodierung der Variablen, deren Anzahl usw. ergibt. Dann erfolgt die Aufstellung der Zielfunktion. Oftmals muß dazu weiteres Wissen über das Systemverhalten ermittelt werden. Erst danach beginnt die Auswahl des zu verwendenden Evolutionären Algorithmus mit seinen Operatoren. Damit sind die Voraussetzung zur Durchführung der Optimierungen und Ermittlung der Lösung des Problems gegeben.



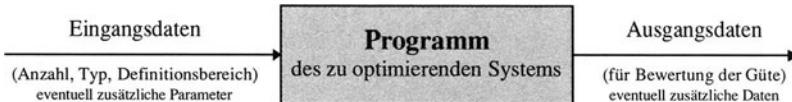
**Abb. 8-1.** Vorgehensweise zur Lösung von Problemen unter Verwendung Evolutionärer Algorithmen

Einige der hier aufgeführten Schritte sind auch bei der Verwendung anderer Optimierungswerkzeuge bzw. -verfahren einsetzbar. Zusätzlich wird auf einige Varianten zur Untersuchung des Systemverhaltens eingegangen, die der Erarbeitung zusätzlichen Wissens über das zu lösende Problem dienen und auch ohne Optimierung anwendbar sind. Die entsprechenden Werkzeuge sind zumeist in der GEATbx enthalten.

### 8.1.1 Einordnung des Problems und Festlegung der Zielfunktion

Zu Beginn der Arbeit an neuen Anwendungen muß eine Einordnung als Optimierungsproblem stattfinden. Dabei muß überlegt werden, wie das zu untersuchende System definiert ist und welche Schnittstellen nach außen vorhanden sind oder eingebaut werden müssen. Dies beinhaltet die Festlegung der Eingangsdaten (mit ihrem jeweiligen Typ und Wertebereich), die Definition von zusätzlichen Parametern (Parameter, die für die Steuerung des Verhaltens des Systems notwendig sind, aber nicht in die Optimierung mit eingebunden werden sollen) sowie die erforderlichen Ausgangsdaten, die für eine Bewertung der Güte des Systems verwendet werden.

Das System muß am Ende als ein Programm vorliegen, daß vollständig durch die übergebenen Eingangsdaten (und eventuell zusätzliche Parameter) gesteuert wird und als Ergebnis der Ausführung die Ausgangsdaten zurück liefert. Dieses Programm entspricht der Zielfunktion. In Abbildung 8-2 ist diese Struktur grafisch dargestellt.



**Abb. 8-2.** Darstellung der Struktur des zu optimierenden Systems als Zielfunktion

Die Zielfunktion findet nicht nur innerhalb der Optimierung Verwendung. Sie kann generell bei der Simulation des Programms/Systems mit Eingangsdaten eingesetzt werden, z.B.:

- interaktiver Test mit systematisch ermittelten Eingangsdaten (Aufruf der Zielfunktion mit einem Datensatz, Rückgabe des Gütwertes),
- automatische Simulation mehrerer Datensätze (Verwendung bei Variationsberechnungen – direkte Darstellung der Zielfunktion, s. Unterabschn. 5.5.1, ab S.145, sowie den folgenden Unterabschn. 8.1.2),
- Einsatz bei der problemspezifischen Visualisierung (Simulation der Zielfunktion und Auswertung eventueller zusätzlicher Ausgabeparameter, die zusätzliche Informationen über das System liefern, z.B. Verlauf von Zuständen bei einer Simulation dynamischer Systeme).

Auf die umfassende und universal einsetzbare Definition und programmtechnische Umsetzung der Zielfunktion zur Abdeckung all dieser Einsatzbereiche sollte besonderer Wert gelegt werden. Die Verwendung ein und desselben Programms für alle Simulationen des zu bearbeitenden Problems (Optimierung, problemspezifische Visualisierung, Einzelsimulation) hat sich in vielen Anwendungen bewährt und erleichtert die Pflege und Verwaltung der entsprechenden Programme.

### 8.1.2 Untersuchungen des Systemverhaltens

Zu Beginn der Arbeit mit einem neuen Problem steht die Frage, welche Eigenschaften das zu untersuchende System bzw. dessen Zielfunktion hat, wie das Gütegebirge aussieht und ob spezielle Besonderheiten auftreten. Diese Fragestellungen treten oft auch nach ersten durchgeföhrten Optimierungen auf, bei denen nicht die erhofften Ergebnisse erhalten wurden.

Die Ergebnisse der im folgenden vorgeschlagenen Untersuchungen können neben neuen Informationen über das Verhalten des Systems wichtige Hinweise auf die Art der Zielfunktion bzw. die für eine erfolgreiche Optimierung anzuwendenden Optimierungsverfahren geben.

In der Regel stellt das zu untersuchende System für die Optimierung ein hochdimensionales Problem dar. In Abhängigkeit der übergebenen Parameter wird ein Zielfunktionswert als Ergebnis der Anwendung der Zielfunktion zurückgegeben.

Die allgemein anwendbaren Untersuchungen des Systemverhaltens lassen sich in die folgenden Bereiche unterteilen:

1. ein- und zweidimensionale Schnitte durch die Zielfunktion, Abschn. 5.5,
2. mehrdimensionale Visualisierung der Zielfunktion, Abschn. 5.4,
3. Überlegungen zur Beschränkung der Systemgröße.

### ***Ein- und zweidimensionale Schnitte (Variationsdiagramme)***

Die zu untersuchenden Systeme sind meist hochdimensional (mehr als fünf Dimensionen bzw. fünf Parameter). Dadurch ist eine direkte grafische Darstellung des Gütegebirges nicht möglich. Um trotzdem Teileaspekte des Aussehens der Zielfunktion zu veranschaulichen, können ein- bzw. zweidimensionale Schnitte durch die Zielfunktion angefertigt werden, s. Abschn. 5.5, ab S.144. Die Ergebnisse dieser Schnitte sind direkt mit Standarddiagrammen zu visualisieren.

Die Schnitte durch die Zielfunktion werden durch einen bestimmten Punkt im Suchraum (hier als Mittelpunkt bezeichnet) durchgeführt. Für die Berechnung der Schnitte durch diesen Mittelpunkt werden alle nicht zu variierenden Parameter des Systems konstant auf den Werten des Mittelpunktes gehalten. Die zu variierenden oder freien Parameter werden in einem festzulegenden Bereich um den Mittelpunktswert verändert, die korrespondierenden Zielfunktionswerte berechnet und die Ergebnisse in 2-D oder 3-D Grafiken veranschaulicht. Der Mittelpunkt kann das Ergebnis einer Optimierung sein und damit einen sehr guten Punkt darstellen oder in einem Bereich des Suchraumes liegen, der von besonderem Interesse ist.

Bedingt durch die Beschränkung der Visualisierung auf drei Dimensionen können nur eindimensionale (Variation eines Parameters) und zweidimensionale Schnitte (Variation von zwei Parametern) durchgeführt werden. Eindimensionale Schnitte sind relativ schnell zu berechnen, dafür können diese Schnitte auch nur Informationen entlang der Koordinatenachse dieses einen variierten Parameters liefern. Zweidimensionale Schnitte benötigen deutlich mehr Berechnungen, ermöglichen dagegen zusätzlich Aussagen über die Wechselwirkung zwischen den beiden variierten Parametern.

Die folgenden Informationen können aus den Schnitten durch die Zielfunktion abgeleitet werden:

- grober Eindruck von der Zielfunktion,
- ist die Zielfunktion glatt oder rauh,
- sind mehrere Minima in der Zielfunktion enthalten (Existenz lokaler Minima),
- bestehen Korrelationen zwischen einzelnen Variablen,
- welche Variablen sind sehr empfindlich, welche Variablen haben kaum Einfluß auf die Zielfunktion,
- Eingrenzen der interessierenden Bereiche der Variablen (wichtig für neue Probleme, bei denen die Grenzen der Variablen wenig oder nicht bekannt sind).

Es muß allerdings beachtet werden, daß alle aus diesen Schnitten abgeleiteten Informationen nur für den dargestellten Bereich gelten und auch dort Einschränkungen unterliegen. Auf Grund der oftmals nichtlinearen Zusammenhänge zwischen den Parametern kann es für andere Bereiche zu deutlich anderen Aussagen kommen. Deshalb liefern die abgeleiteten Informationen nur Anhaltspunkte für die untersuchten Bereiche. Die Ergebnisse können auch für andere Bereiche gelten, sie müssen es aber nicht. Unter diesem Blickwinkel müssen die Informationen und Ergebnisse betrachtet werden.

Beispiele für ein- und zweidimensionale Schnitte werden in Unterabschn. 5.5.1, ab S.145 gezeigt. Insbesondere sei auf Abb. 5-19 eben dort verwiesen. Bei der Parameteridentifikation eines Dieselmotormodells werden Variationsdiagramme zur Untersuchung des Systemverhaltens eingesetzt, s. Unterabschn. 8.3.2, ab S.207.

### **Mehrdimensionale Visualisierung**

Mit herkömmlichen Visualisierungsmethoden ist es möglich, die Abhängigkeit von bis zu zwei Parametern direkt darzustellen, da diese Methoden auf 3 Dimensionen beschränkt sind. Für eine Erweiterung auf höhere Dimensionen müssen weitere Zwischenschritte realisiert werden.

Dazu werden Verfahren benötigt, welche die Konfiguration von hochdimensionalen Punkten in eine niedrigdimensionale Konfiguration umsetzen, so daß die Relationen zwischen den einzelnen Datenpunkten weitestgehend erhalten bleiben. Diese Verfahren werden unter dem Begriff *Multidimensional Scaling* zusammengefaßt. Eine der bekannten Methoden aus dieser Gruppe ist das SAMMON-Mapping [Sam69]. Die mathematischen Grundlagen dieses Verfahrens werden in Abschn. 5.4, ab S.138 vorgestellt. Es muß allerdings beachtet werden, daß alle diese Verfahren mit einem Informationsverlust verbunden sind. Auf der einen Seite ist nur dadurch eine Visualisierung der Information möglich, auf der anderen Seite muß diese Verringerung bzw. Beschränkung der dargestellten Daten bei der Interpretation der Ergebnisse berücksichtigt werden.

Die mehrdimensionale Visualisierung kann für die folgenden Aufgaben verwendet werden:

- Darstellung der Ähnlichkeit von Lösungen im Variablenraum und den korrespondierenden Zielfunktionswerten (Vergleich von einzelnen Optimierungslösungen untereinander),
- Darstellung des „Weges der sich verbessernden Lösung durch den Variablen-Suchraum“ während einer Optimierung,
- Darstellung des „Weges der sich verbessernden Lösung durch den Zielfunktions-Suchraum“ während einer Optimierung (bei mehrkriteriellen Problemen),
- Vergleich der „Wege durch den Suchraum“ von verschiedenen Optimierungen,
- Versuch der Erzeugung eines niedrigdimensionalen Abbildes des Gütegebirges der Zielfunktion.

Die ersten vier Aufgaben können unter Zuhilfenahme von Funktionen, die in [GEATbx] enthalten sind, berechnet und anschließend visualisiert werden. Die letzte Variante stößt im Moment noch auf rechentechnische Probleme und Visualisierungsprobleme, da selbst bei einer Zielfunktion mit nur zehn Dimensionen bzw. Variablen mit einer riesigen Datenmenge gearbeitet werden muß.

Beispiele für die Anwendung der ersten vier Varianten sind in Abschn. 5.4, ab S.138, dargestellt. Mit diesen Methoden lassen sich komplexe Zusammenhänge in den Daten anschaulich darstellen, die auf anderem Weg nicht oder nur mit sehr hohem Aufwand bzw. nach längerer Einarbeitung zugänglich sind.

## Beschränkung der Systemgröße

Ein wichtiger Aspekt der Voruntersuchung von neuen Systemen ist eine Analyse der Dimensionalität des Problems und mögliche Wege seiner Verringerung. Reale Systeme sind oftmals mit einer sehr hohen Dimension gegeben, die sich aus der Aufgabenstellung ergibt. Für die Optimierung ist es meistens möglich, diese Dimension zu verringern. Wenn eine Verringerung der Dimension nicht direkt möglich ist, sollte zumindest versucht werden, erste Versuche mit einem verkleinerten bzw. beschränkteren System vorzunehmen, um an diesem System erste Erfahrungen zu sammeln.

Welche Varianten einer Verringerung der Dimension oder Schwere eines Problems sollten im allgemeinen untersucht werden:

- **Verringerung der Anzahl der Variablen:**

Von den vorhandenen Variablen werden einige weggelassen (Verringerung der Dimension) oder auf feste Werte gesetzt (Reduzierung der Größe des Problems). Die Verringerung der Variablenanzahl stellt eine wichtige Variante dar, die immer so weit wie möglich ausgenutzt werden sollte.

- **Skalierbarkeit des Problems:**

Die Variablen eines Systems stellen oft diskrete Werte eines zu optimierenden Verlaufs dar. Je nach Feinheit der Diskretisierung verändert sich die Anzahl der Variablen des Optimierungsproblems. Für erste Optimierungen sollte mit einer groben Diskretisierung gearbeitet werden, die im weiteren Verlauf den tatsächlichen Erfordernissen angepaßt wird. Außerdem können manche Probleme zuerst mit einer geringeren Anzahl von Variablen betrachtet werden, ohne daß die Problemstellung verfälscht wird.

- **Unterteilung des Problems in mehrere Optimierungsaufgaben:**

Wenn es möglich ist, eine Optimierungsaufgabe in mehrere Aufgaben zu unterteilen, so stellt dies einen unbedingt zu nutzenden Weg dar, um die Gesamtaufgabe in kleinere, überschaubarere Teilaufgaben zu zerlegen. Ein Beispiel ist die Optimierung einer diskretisierten Steuerung eines Systems. Der gesamte Zeithorizont kann in mehrere Einzelstücke geteilt werden. Der Endzustand eines Abschnitts stellt jeweils den Anfangszustand des folgenden Abschnitts dar. Durch diese Methode können selbst Probleme gelöst werden, die sonst mit heutigen Methoden und Mitteln nicht beherrschbar sind. Allerdings ist die Anwendung problemspezifisch. Außerdem muß auf eventuelle Auswirkungen der Trennung der Einzelabschnitte in der Optimierung geachtet werden.

In Unterabschn. 8.5.4 wird auf die Anwendung einiger der angeführten Methoden eingegangen.

### 8.1.3 Festlegung des Optimierungsverfahrens

Mit den Kenntnissen über die Art eines zu behandelnden Systems bzw. den Erkenntnissen, die aus den Voruntersuchungen zum Systemverhalten gewonnen

wurden (Unterabschn. 8.1.2), muß ein Optimierungsverfahren zur Lösung des Problems ausgewählt werden. Dies kann ein einzelnes Verfahren sein, eine Anzahl von miteinander zu kombinierenden Verfahren bzw. Varianten einzelner oder mehrerer Verfahren.

Es gibt eine große Menge unterschiedlicher Problemklassen. Für jede Problemklasse ist ein anderes Verfahren bzw. eine andere Variante am besten geeignet.

In Kap. 7 werden Hinweise zur Auswahl von Verfahren und Operatoren sowie zur Festlegung der zugehörigen Optionen gegeben. Abschnitt 7.1 enthält Hinweise für Verfahren und Operatoren, die für die meisten zu lösenden Problemklassen in der gleichen Weise verwendet werden können. In den Abschn. 7.2 und folgende wird auf die Definition von Optimierungsverfahren für spezielle Problemklassen eingegangen. Diese Empfehlungen beziehen sich auf die Erfahrungen des Autors aus der bisherigen Arbeit an verschiedenen Systemen sowie Ergebnissen anderer Autoren, die in der Literatur berichtet wurden.

#### **8.1.4 Durchführung und Auswertung von Optimierungen**

Nachdem ein oder mehrere Optimierungsverfahren zur Lösung des Problems ausgewählt wurden, werden erste Optimierungen durchgeführt. Diese können darüber Aufschluß geben, wie schnell das Problem gelöst werden kann.

Wenn in den ersten Versuchen gleich die optimale Lösung (bei Kenntnis des Optimums) oder eine sehr gute bzw. ausreichend gute Lösung gefunden wird, kann das Problem aus ingenieurtechnischer Sicht als gelöst betrachtet werden. Je nach den weiteren Anforderungen kann aber noch mit den im folgenden beschriebenen Methoden nach einem Verfahren gesucht werden, welches das Problem noch besser bzw. mit geringerem Aufwand löst.

Meist kann das Problem nach ersten Läufen noch nicht als gelöst betrachtet werden. Entweder liefern die ersten Läufe keine befriedigenden Ergebnisse oder bei Unkenntnis über das Optimum ist die Unsicherheit bezüglich der erreichten Ergebnisse zu groß. An dieser Stelle beginnt die Suche nach Verfahren, die zur Lösung des Problems besser geeignet sind.

Dabei gibt es prinzipiell zwei Ansätze: es werden andere Verfahren, Operatoren bzw. Optionen verwendet oder die Anzahl der Individuen und/oder Interpolationen wird erhöht. Letzteres führt zu einer deutlichen Erhöhung der Rechenzeit, da die Anzahl der Individuen linear in die gesamte Rechenzeit eingeht (zumindest bei den meisten realen Problemen). Wenn die gesamte Rechenzeit aber noch unter einem Tag liegt, so kann damit erst einmal geklärt werden, in wieweit auf diesem Wege eine Lösung erreicht werden kann.

Wenn die bisher angewandten Verfahren nicht zum Erfolg führen, muß nach neuen Varianten gesucht werden. An diesem Punkt sollte über die Eigenschaften des Systems neu nachgedacht werden. Da das System nicht wie erhofft zu optimieren ist, muß das System Eigenschaften besitzen, die nicht in den Auswahlprozeß der Verfahren einbezogen wurden. Durch die Verwendung weiterer Verfahren können sich somit neue Erkenntnisse über das zu lösende Problem ergeben.

Eine wichtige Hilfe bei der Auswertung von durchgeführten Läufen sind die in Kapitel 5, ab S.115 vorgestellten Visualisierungsmöglichkeiten. Diese erlauben einen detaillierten Einblick in den Zustand der Population zu den verschiedensten Zeitpunkten einer Optimierung sowie in den Verlauf einer Optimierung. Damit lassen sich Fragen nach einer vorzeitigen Konvergenz der Population, dem Vorhandensein mehrerer Extremwerte oder dem Verlauf der Variablen guter Individuen beantworten. Die Anwendung dieser Verfahren erfordert eine gewisse Einarbeitung, um die dargebotenen Grafiken zu interpretieren. Die Visualisierungsverfahren bieten die beste Einsicht in den aktuellen Zustand und den Verlauf des Optimierungsprozesses und eröffnen neue Wege zur Anpassung der verwendeten Parameter der Optimierungsverfahren. Beispiele für die Anwendung der Visualisierungsverfahren werden in den folgenden Abschnitten gezeigt.

An dieser Stelle sei auf die gleichzeitige Anwendung verschiedener Strategien in einem Optimierungslauf verwiesen, die in Abschn. 4.5, ab S.96 vorgestellt wird. Dies eröffnet zum einen die Möglichkeit, gleichzeitig mehrere Strategien zur Problemlösung zu verwenden. Auf der anderen Seite tritt es sehr oft auf, daß in Abhängigkeit des Fortschritts der Optimierung zu unterschiedlichen Zeitpunkten in einem Optimierungslauf unterschiedliche Verfahren am besten geeignet sind. Damit unterstützen sich die Strategien untereinander, erst der Erfolg der einen Strategie ermöglicht den späteren Erfolg einer anderen Strategie.

Ein gutes Beispiel dafür ist der Einsatz von mehreren Mutationsvarianten, wobei eine Variante eine grobe Suche durchführt, eine weitere eine feinere Suche und eine andere eine sehr feine Suche. Ähnliches kann durch die gleichzeitige Anwendung verschiedener Rekombinationsoperatoren erreicht werden. In der anschließenden Auswertung kann dann sehr einfach überprüft werden, wann welche der eingesetzten Strategien erfolgreich war. Auf diesem Weg können erfolgversprechende Strategien gefunden werden, die ein Problem besser oder in kürzerer Zeit lösen.

Wenn selbst nach Einsatz der bekannten Verfahren die Lösung des Problems nicht zufriedenstellend ist, muß über das weitere Einbringen von problemspezifischem Wissen nachgedacht werden. Dies beinhaltet Dinge wie:

- die spezielle Initialisierung der Individuen der Anfangspopulation (Abschn. 3.6, ab S.60),
- das angepaßte Einschränken der Wertebereiche der Variablen und
- die Entwicklung spezieller Operatoren.

Die ersten beiden Punkte sollten immer angewendet werden, wenn dieses Wissen vorliegt. Dadurch kommt es meistens zu einer deutlichen Verkleinerung des Suchbereiches und damit zu einer Beschleunigung des Auffindens zufriedenstelender Lösungen. Die Entwicklung spezieller Operatoren ist in den meisten Fällen schwierig und für bisherige Aufgabenstellungen nicht erforderlich gewesen.

Mit dem in diesem Abschnitt vorgestellten Vorgehen zur Lösung von Optimierungsproblemen und den dargestellten Varianten sollte es möglich sein, sehr viele in der ingenieurtechnischen Arbeit auftretende Probleme zu lösen. Größere und komplexere Probleme erfordern eine längere Einarbeitungsphase sowie die intensive Auseinandersetzung mit dem Problem in verschiedenen Optimierungsversuchen. Am Ende, wenn Verfahren und Methoden zur Lösung des Problems gefunden wurden, zeigt sich sehr oft, daß in der Phase der Optimierung neue Er-

kenntnisse über das zu lösende System gefunden wurden, die vorher auch den entsprechenden Fachleuten nicht bekannt waren. Durch die Optimierung werden Bereiche des zugrunde liegenden Systems untersucht, die vorher gar nicht in Erwägung gezogen wurden. Dieser Erkenntniszuwachs für beide Seiten, die Fachleute für das zu optimierende System sowie den Systemtechniker, der die Optimierung durchführt, ist ein nicht zu unterschätzender Aspekt.

Wenn nach intensiver Beschäftigung mit dem System festgestellt wird, daß es durch ein anderes Optimierungsverfahren als Evolutionäre Algorithmen besser gelöst werden kann, dann sollte immer dieses Verfahren eingesetzt werden. Wenn ein Problem mit einem speziellen Verfahren gelöst werden kann (z.B. gradientenbasierte Verfahren), dann führt dieses fast immer schneller zu einer optimalen Lösung, als der Einsatz Evolutionärer Algorithmen. Allerdings stellen diese speziellen Verfahren oft Anforderungen an die Eigenschaften der zu lösenden Probleme, die von vielen realen Systemen nicht garantiert werden können (quadratische Form, Differenzierbarkeit usw.). In diesen Fällen sind Evolutionäre Algorithmen mit ihrem großen Spektrum an leistungsfähigen Verfahren und Operatoren das Verfahren der Wahl, um eine Lösung des Problems zu erreichen.

## 8.2 Optimierung mehrdimensionaler Funktionen

In diesem Abschnitt werden mehrdimensionale mathematische Funktionen als Beispiele für die Anwendung Evolutionärer Algorithmen verwendet. Diese mathematischen Funktionen haben bekannte Eigenschaften, sind gut zu überschauen und benötigen nur wenig Zeit zur Berechnung. Damit sind sie ideal als Testfunktionen geeignet.

Für die Veranschaulichung des Einsatzes Evolutionärer Algorithmen kommen 3 Testfunktionen (RASTRIGIN's Funktion, ROSENROCK's Funktion, *Moved axis parallel hyper-ellipsoid* Funktion) zum Einsatz, die jeweils spezifische Eigenschaften haben. Jede dieser Funktionen wird mit 3 verschiedenen Evolutionären Algorithmen optimiert. In der Auswertung der einzelnen Optimierungen werden die Ergebnisse miteinander verglichen und bewertet.

Unterabschnitt 8.2.1 erläutert die 3 verwendeten Evolutionären Algorithmen. Dabei wird nur kurz auf die verwendeten Operatoren und deren Optionen eingegangen. Die eingesetzten Evolutionären Algorithmen entsprechen den in Kap. 7, ab S.171, vorgeschlagenen Kombinationen von Verfahren. Eine ausführliche Motivation der einzelnen Werte erfolgt dort.

Jeder der nachfolgenden Unterabschnitte behandelt eine der eingesetzten Testfunktionen. Zuerst werden Definition und Eigenschaften der Funktion vorgestellt. Im Anschluß an die Beschreibung der Funktion erfolgt die Auswertung der durchgeführten Optimierungen und der Vergleich der Ergebnisse in Bezug auf die eingesetzten Optimierungsalgorithmen.

Die Optimierung der mehrdimensionalen Funktionen bietet einen leichten Zugang zum Einsatz Evolutionärer Algorithmen. Die Probleme sind einfach in ihrer Definition und gut zu verstehen. Die hier vorgestellten Experimente können leicht wiederholt werden. Damit bieten diese Testfunktionen einen idealen Startpunkt für die Anwendung Evolutionärer Algorithmen in der Optimierung.

### 8.2.1 Verwendete Optimierungsverfahren

Für die Optimierung der mathematischen Funktionen werden 3 Optimierungsalgorithmen eingesetzt. Diese orientieren sich an den Empfehlungen in Kap. 7.

Der erste Optimierungsalgorithmus ist ein global orientierter Evolutionärer Algorithmus mit mehreren Strategien für reelle Variablen (Abschn. 7.2, ab S.175), der in diesem Abschnitt als MPEA1 (*Multi subPopulation Evolutionary Algorithm*) bezeichnet wird:

- 4 Unterpopulationen mit je 25 Individuen (100 Individuen insgesamt),
- Fitneßzuweisung durch lineares Ranking mit Selektionsdruck von 1,7,
- Selektion durch *stochastic universal sampling, generation gap* von 0,9,
- diskrete Rekombination mit einer Rekombinationsrate von 1,
- Mutation reeller Variablen mit einer Mutationspräzision von 12 oder 16, der Mutationsbereich wird für jede Unterpopulation verschieden festgelegt: [0,1 0,01 0,001 0,0001], die Mutationsrate beträgt 1/Anzahl der Variablen (da alle Funktionen 10 Variablen haben, wird mit einer Mutationsrate von 0,1 gearbeitet),
- Wiedereinfügen aller Nachkommen, Ersetzen der schlechten Eltern,
- Verwendung des regionalen Populationsmodells, Migration alle 20 Generationen, 10% der besten Individuen werden in einer uneingeschränkten Netzstruktur ausgetauscht,
- Konkurrenz zwischen den Unterpopulationen findet zu jeder 4. Generation statt, 10% der Individuen müssen von schlechten Unterpopulationen abgegeben werden, das Unterpopulationsminimum beträgt 5 Individuen.

Der zweite Optimierungsalgorithmus ist ebenfalls ein global orientierter Evolutionärer Algorithmus für reelle Variablen. Im Gegensatz zum ersten Algorithmus wird eine Gesamtpopulation eingesetzt. Die Unterschiede des SPEA2 (*Single Population Evolutionary Algorithm*) zum MPEA1 sind:

- 1 Population mit 100 Individuen (globales Populationsmodell),
- Mutation reeller Variablen mit einer Mutationspräzision von 24 und einem Mutationsbereich von 0,01,
- keine Migration und keine Konkurrenz.

Beim dritten Optimierungsalgorithmus handelt es sich um einen lokal orientierten Evolutionären Algorithmus für reelle Variablen (Abschn. 7.3, ab S.176). Dieser wird hier als LEA3 (*Local Evolutionary Algorithm*) bezeichnet:

- 1 Population mit 12 Individuen,
- Fitneßzuweisung durch nichtlineares Ranking, Selektionsdruck von 6,
- Selektion durch Truncation-Selektion mit einem *generation gap* von 1,
- Wiedereinfügen von 12 Nachkommen in die Population, alle Eltern werden ersetzt,
- keine Rekombination,
- Mutation durch eine reelle Mutation mit Adaption der Schrittweiten und einer Richtung (Unterabschn. 3.4.2, ab S.51), Mutationsbereich von 0,01.

Die vorgestellten 3 Optimierungsalgorithmen werden auf die mathematischen Funktionen angewendet. Die im folgenden beschriebenen Resultate sind das Er-

gebnis einzelner typischer Optimierungsläufe. Da Evolutionäre Algorithmen stochastische Suchverfahren sind, sieht jeder Lauf etwas anders aus. Die grundsätzlichen Ergebnisse zeigen sich aber in allen Experimenten.

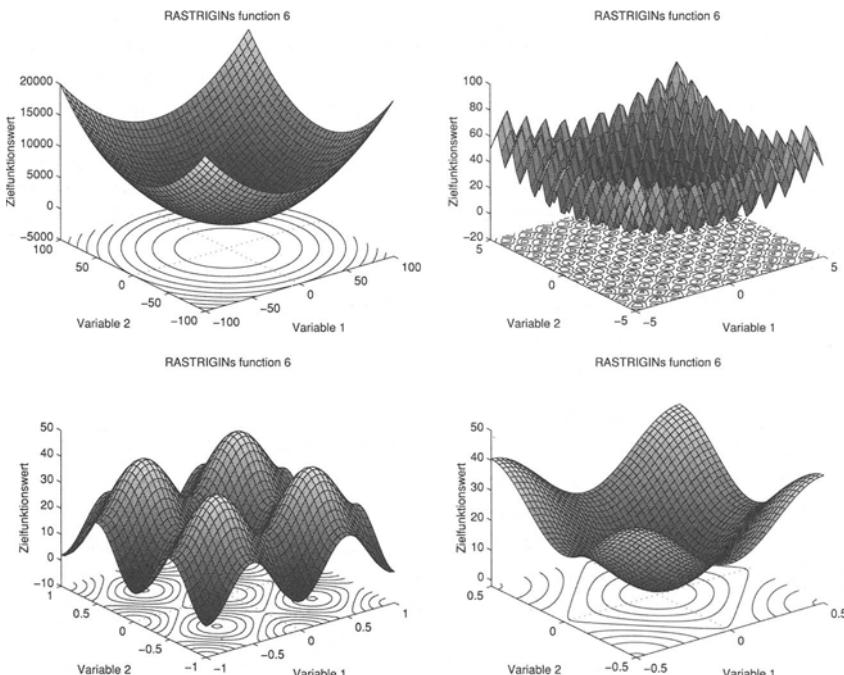
### 8.2.2 RASTRIGIN's Funktion

Eine häufig verwendete Testfunktion ist RASTRIGIN's Funktion. Die Funktion basiert auf einer quadratischen Funktion (DE JONG's Funktion 1 oder *hyper sphere*). Zu dieser werden Kosinus-Modulationen addiert, wodurch viele lokale Minima entstehen. RASTRIGIN's Funktion ist multimodal. Die Minima sind regelmäßig verteilt (Abstand von 1) und die Funktion ist separierbar. RASTRIGIN's Funktion kann in beliebigen Dimensionen (Anzahl von Variablen) verwendet werden.

Definition von RASTRIGIN's Funktion und globales Minimum:

$$f_6(x) = 10 \cdot n + \sum_{i=1}^n (x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i)) \quad (8-1)$$

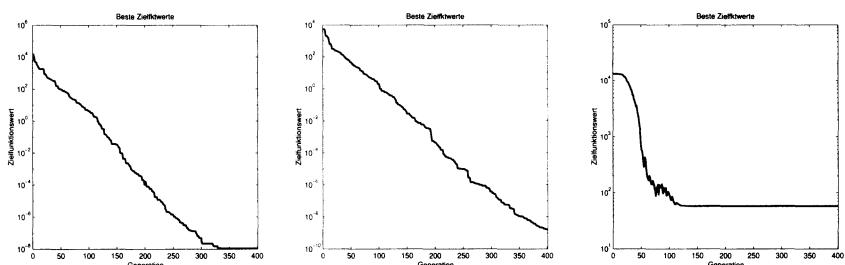
globales Minimum:  $f_6(x^*) = 0; \quad x_i^* = 0, \quad i = 1:n$



**Abb. 8-3.** Aussehen von RASTRIGIN's Funktion in zwei Dimensionen (Variable 1 und 2); oben links: gesamter Suchraum [-500 500], oben rechts: beschränkter Suchraum [-5 5], unten links: Umgebung des Optimums [-1 1], unten rechts: Einzugsbereich des Optimums [-0,5 0,5]

Einen guten Eindruck der Funktion vermitteln zweidimensionale Schnitte (Variationsdiagramme). In Abhängigkeit des verwendeten Definitionsbereiches der Variablen ergeben sich dabei ganz unterschiedliche Bilder.

In Abb. 8-3 links oben ist der gesamte Suchraum zu sehen. Dies entspricht einem Definitionsbereich von  $[-500 \ 500]$  für jede Variable. Aus dieser Entfernung sieht die Funktion sehr einfach aus, ähnlich DE JONG's Funktion 1. Erst bei einem verkleinerten Definitionsbereich von  $[-5 \ 5]$  ist die Multimodalität der Funktion zu erkennen, Abb. 8-3 rechts oben. Noch deutlicher zeigen sich die lokalen Minima (und Maxima) in einem kleinen Definitionsbereich von  $[-1 \ 1]$ , Abb. 8-3 links unten. Aus diesem Diagramm kann der Abstand zwischen den Minima mit 1 abgelesen werden. Sobald die Werte in einem Bereich kleiner als  $[-0,5 \ 0,5]$  liegen, Abb. 8-3 rechts unten, wird die Funktion wieder zu einer einfachen quadratischen Funktion.



**Abb. 8-4.** Optimierung von RASTRIGIN's Funktion, Zielfunktionswert des besten Individuums (Konvergenzdiagramm); links: MPEA1, Mitte: SPEA2, rechts: LEA3

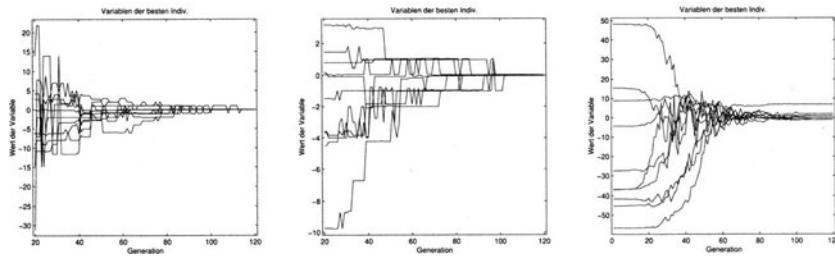
Die Optimierung wurde mit den 3 Verfahren über jeweils 400 Generationen durchgeführt. Einen ersten Einblick in das Ergebnis der Optimierungen geben die Konvergenzdiagramme in Abb. 8-4.

Die beiden global orientierten Verfahren MPEA1 und SPEA2 hatten keine Probleme, RASTRIGIN's Funktion zu lösen. Beide Verfahren erreichen minimale Zielfunktionswerte von  $10^{-8}$ . Dieser Wert liegt weit unter 1 und zeigt, daß das globale Minimum gefunden wurde. In Abb. 8-5 links und Mitte ist an den Variablen der besten Individuen zu erkennen, daß bereits nach reichlich 100 Generationen das Einzugsgebiet des globalen Minimums von beiden Verfahren erreicht wurde.

Im Gegensatz dazu hat das lokal orientierte Verfahren LEA3 das globale Minimum nicht erreicht. Wie nicht anders zu erwarten war, bleibt das Verfahren in einem lokalen Minimum stecken. In Abb. 8-4 rechts ist zu sehen, daß sich der Zielfunktionswert des besten Individuums ab Generation 120 nicht mehr verbessert. Der Wert liegt weit oberhalb von 1. Aus Abb. 8-5 rechts kann entnommen werden, daß die Variablen der besten Individuen nicht zu 0 werden.

Gegen Ende der Optimierung kommt es beim Verfahren MPEA1 zu einem Stillstand, der Zielfunktionswert des besten Individuums verbessert sich ab Generation 330 nicht mehr, Abb. 8-4 links. In dieser Stelle stößt die feinste Mutation (Unterpopulation 4) an ihre Grenzen. Der Abstand zum Optimum ist kleiner als

der minimal mögliche Mutationsschritt (Mutationspräzision von 12). Dadurch kann es zu keiner weiteren Verbesserung kommen. Durch die hohe Mutationspräzision von 24 beim Verfahren SPEA2 ist dieses noch nicht an seine Grenzen gestoßen. Bei weiteren Optimierungen dieser Funktion sollte die Mutationspräzision von MPEA1 auf 16 oder 20 erhöht werden.



**Abb. 8-5.** Optimierung von RASTRIGIN's Funktion, Variablen der besten Individuen (Beschränkung auf die Generationen 1 bis 120); links: MPEA1, Mitte: SPEA2, rechts: LEA3

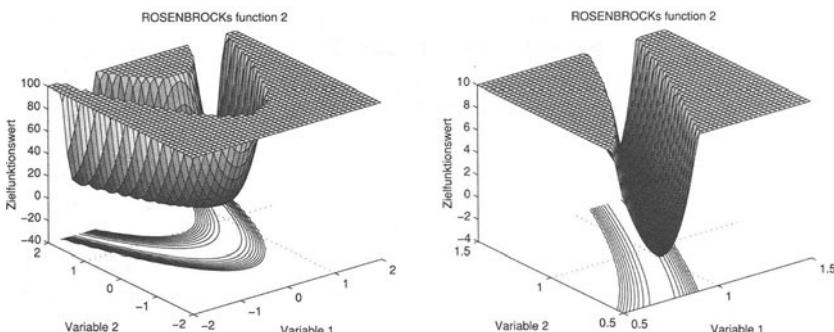
### 8.2.3 ROSENBROCK's Funktion

Eine bekannte Testfunktion ist ROSENBROCK's Funktion, oft als Bananenfunktion bezeichnet. Die ursprüngliche Funktion war nur für 2 Dimensionen angegeben, wurde in der Definition aber auf beliebige Dimensionen erweitert.

Definition von ROSENBROCK's Funktion und globales Minimum:

$$f_2(x) = \sum_{i=1}^{n-1} 100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2$$

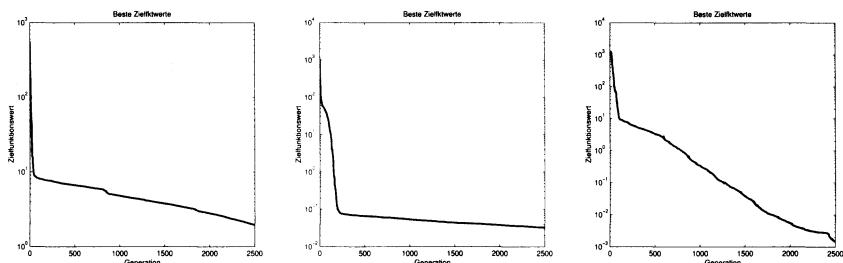
globales Minimum:  $f_2(x^*) = 0; \quad x_i^* = 1, \quad i = 1:n$



**Abb. 8-6.** Aussehen von ROSENBROCK's Funktion in zwei Dimensionen (Variable 1 und 2); links: gebogenes Tal im Suchraum [-2 2], rechts: Bereich um Optimum [0,5 1,5]

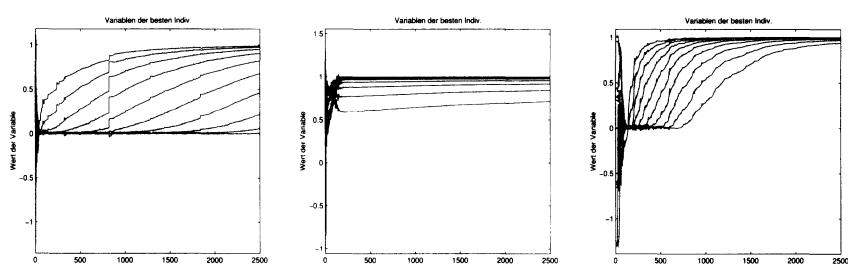
Die Variablen sind stark miteinander korreliert. Außer dem globalen Minimum existiert ein lokales Minimum. Das globale Minimum liegt in einem schmalen gekrümmten Tal. Die Variationsdiagramme in Abb. 8-6 vermitteln einen guten Eindruck davon.

Das linke Diagramm zeigt die Funktion im Definitionsbereich von [-2 2] für die ersten beiden Variablen. Das schmale und stark gekrümmte Tal ist gut zu erkennen. Im rechten Diagramm wurde der Definitionsbereich auf [0,5 1,5] eingeschränkt und zeigt den Bereich um das globale Minimum. In beiden Grafiken wurde der maximal dargestellte Wert beschränkt (links: 100, rechts: 10). Nur dadurch ergibt sich eine vernünftige Skalierung, aus der die Eigenschaften der Funktion abgelesen werden können. Die gezeigten Plateaus sind nicht Bestandteil der Funktion, sondern resultieren aus dieser Beschränkung.



**Abb. 8-7.** Optimierung von ROSENROCK's Funktion, Zielfunktionswert des besten Individuums (Konvergenzdiagramm); links: MPEA1, Mitte: SPEA2, rechts: LEA3

Die Optimierung von ROSENROCK's Funktion wurde über 2500 Generationen durchgeführt. Zu Beginn der Optimierung kommt es bei allen 3 Verfahren zu einer schnellen Verbesserung der Zielfunktionswerte. Diese Phase entspricht der Suche nach dem Grund des Tales. MPEA1 findet den Grund innerhalb weniger Generationen, Abb. 8-7 links. Dies dürfte seine Ursache mit in der von Unterpopulation 1 verwendeten groben Mutation haben. SPEA2 benötigt deutlich länger zum Finden des Grundes, Abb. 8-7 Mitte, landet aber (durch Zufall) in einem deutlich besseren Bereich.

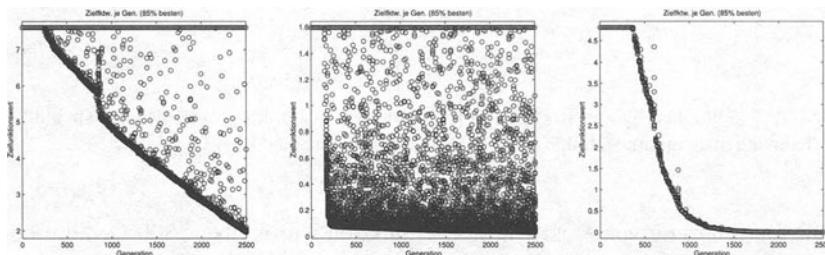


**Abb. 8-8.** Optimierung von ROSENROCK's Funktion, Variablen der besten Individuen; links: MPEA1, Mitte: SPEA2, rechts: LEA3

Der weitere Verlauf bei beiden global orientierten Verfahren ist sehr langsam. Während es bei MPEA1 immerhin noch zu einer gewissen Verbesserung der Zielfunktionswerte kommt, ist davon bei SPEA2 kaum etwas zu erkennen. Noch deutlicher wird dies bei einem Blick auf Abb. 8-8. Die Variablen des besten Individuums verändern sich bei MPEA1 kontinuierlich, Abb. 8-8 links. Für SPEA2 ist kaum eine Veränderung erkennbar, Abb. 8-8 Mitte.

Wie gut LEA3 für die Lösung von ROSENROCK's Funktion geeignet ist, zeigt sich deutlich in den Abbildungen. Über alle Generationen hinweg findet eine deutliche Verbesserung der Zielfunktionswerte statt, Abb. 8-7 rechts. Die Variablen des besten Individuums verändern sich kontinuierlich und erreichen innerhalb der Optimierung fast das globale Minimum, Abb. 8-8 rechts.

Sehr gut ist in Abb. 8-8 links und rechts zu erkennen, daß bei dieser Funktion eine gekoppelte Veränderung aller Variablenwerte stattfinden muß, um zu einer Verbesserung des Zielfunktionswertes zu gelangen. ROSENROCK's Funktion ist ein klassisches Beispiel für ein Problem mit korrelierten Variablen. Dafür ist ein lokal orientierter Evolutionärer Algorithmus wie LEA3, der eine Adaption der Mutationsparameter durchführt, am besten geeignet<sup>1</sup>. Die global orientierten Verfahren sind für eine solche Funktion nicht geeignet.



**Abb. 8-9.** Optimierung von ROSENROCK's Funktion, Zielfunktionswerte aller Individuen; links: MPEA1, Mitte: SPEA2, rechts: LEA3

In Abb. 8-9 wird die Verteilung aller Zielfunktionswerte während der Optimierung gezeigt. Bei MPEA1 sind die Unterpopulationen mit den kleinsten Mutationsschrittweiten am erfolgreichsten und arbeiten mit einem großen Teil der Individuen (hier nicht zu sehen). Dadurch sind viele Zielfunktionswerte dem aktuell besten Wert sehr ähnlich. Die wenigen schlechten Zielfunktionswerte sind durch die Begrenzung der schlechten Individuen nicht zu erkennen. SPEA2 verwendet nur eine Strategie, die viele nicht so schlechte Individuen erzeugt. Dadurch ergibt sich eine Verteilung der Zielfunktionswerte in einem größeren Bereich, Abb. 8-9 Mitte. Die wenigen Individuen in der Population von LEA3 sind sich sehr ähnlich, deutlich schlechtere Individuen können durch die Adaption der Mutationsschrittweiten nicht auftreten, Abb. 8-9 rechts.

<sup>1</sup> Noch besser läßt sich ROSENROCK's Funktion durch ein Gradientenverfahren lösen. Hier dient diese Funktion als Testbeispiel mit korrelierten Variablen.

### 8.2.4 Moved axis parallel hyper-ellipsoid Funktion

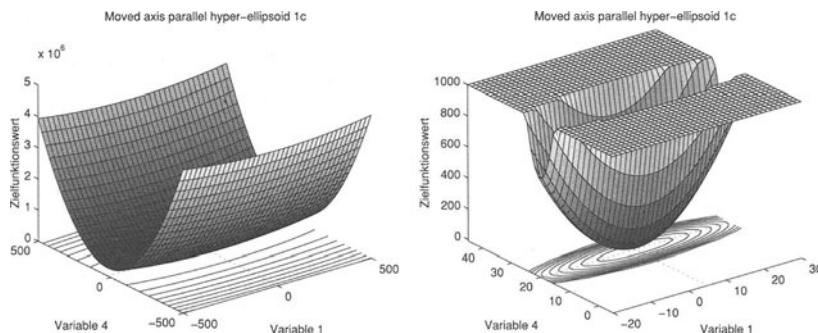
Eine weniger bekannte Testfunktion ist die *moved axis parallel hyper-ellipsoid* Funktion. Grundlage ist ein zu den Koordinatenachsen paralleles Hyperellipsoid. Die Funktion ist separierbar. Um das Optimum aller Variablen nicht bei identischen Werten zu haben, wurde das globale Minimum verschoben (*moved*). Damit eignet sich diese Funktion gut für Demonstrationen.

Definition von *moved axis parallel hyper-ellipsoid* Funktion und globales Minimum:

$$f_{1c}(x) = \sum_{i=1}^n (i \cdot (x_i - 5i))^2$$

$$\text{globales Minimum: } f_{1c}(x^*) = 0; \quad x_i^* = 5 \cdot i, \quad i = 1:n$$

Die Variationsdiagramme in Abb. 8-10 geben einen Eindruck des Aussehens der Funktion. Beide Diagramme zeigen eine Variation der Variablen 1 und 4 bei Verwendung der 4-dimensionalen Funktion. Die Variablen 2 und 3 sind fest auf 10 bzw. 15 (Wert der jeweiligen Variable im Optimum) gesetzt.



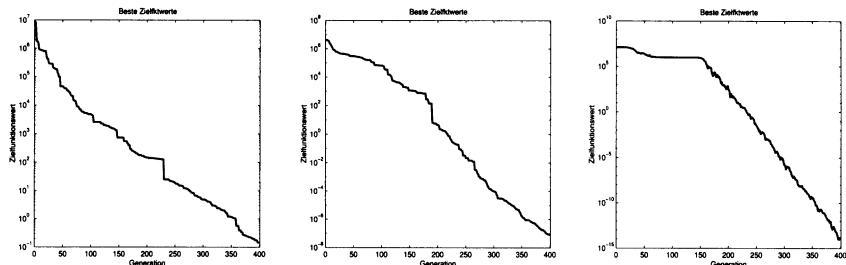
**Abb. 8-10.** Aussehen von *moved axis parallel hyper-ellipsoid* Funktion in zwei Dimensionen (Variable 1 und 4); links: gesamter Suchraum [-500 500], rechts: beschränkter Suchraum [-20 30] bzw. [-5 45]

Das linke Diagramm in Abb. 8-10 visualisiert den gesamten Suchraum von [-500 500]. Die stark elliptische Eigenschaft der Funktion ist gut zu erkennen. Das rechte Diagramm zeigt den Bereich in der Nähe des globalen Minimums, [-20 30] für Variable 1 und [-5 45] für Variable 4. Zusätzlich wurde das Maximum der dargestellten Zielfunktionswerte auf 1000 beschränkt. Die Plateaus sind nicht Bestandteil der Funktion, sondern resultieren aus dieser Beschränkung.

Die Optimierung wurde mit jedem Verfahren über 400 Generationen durchgeführt. Alle 3 Verfahren können diese Funktion lösen. Es zeigen sich allerdings einige Unterschiede.

Bei beiden global orientierten Verfahren MPEA1 und SPEA2 kommt es zu einer gleichmäßigen Verbesserung der Zielfunktionswerte, Abb. 8-11 links und

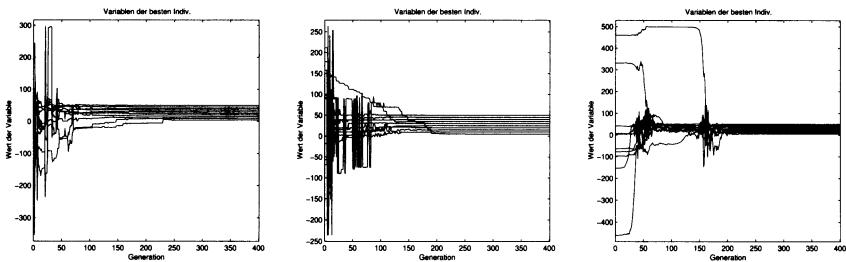
Mitte. SPEA2 erreicht in den hier gezeigten Läufen einen kleineren Zielfunktionswert als MPEA1.



**Abb. 8-11.** Optimierung von *moved axis parallel hyper-ellipsoid* Funktion, Zielfunktionswert des besten Individuums (Konvergenzdiagramm); links: MPEA1, Mitte: SPEA2, rechts: LEA3

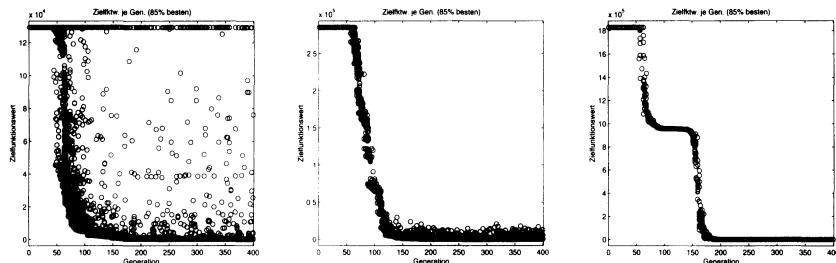
Das lokal orientierte Verfahren LEA3 benötigt eine hohe Anzahl von Generationen zur Adaption der Mutationsparameter. Sobald diese Adaption erreicht wird, werden die Zielfunktionswerte kontinuierlich und mit hoher Geschwindigkeit verbessert, Abb. 8-11 rechts.

Dies zeigt sich auch bei einem Blick auf Abb. 8-12 rechts. Erst nach Generation 150 erreicht die letzte Variable einen Wert in der Nähe des Optimums. Ab diesem Zeitpunkt kann LEA3 seine Stärke ausspielen.



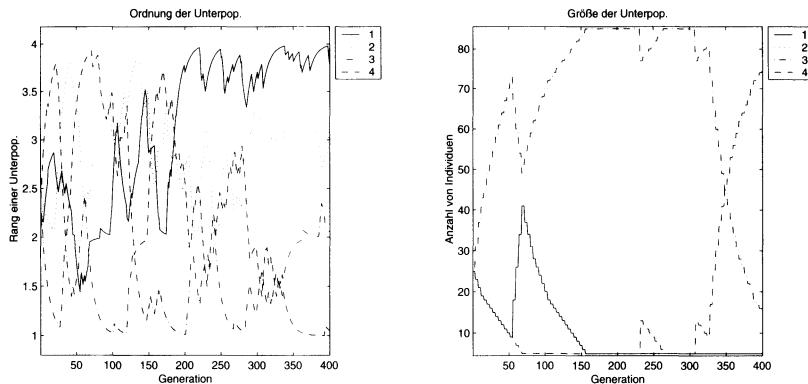
**Abb. 8-12.** Optimierung von *moved axis parallel hyper-ellipsoid* Funktion, Variablen der besten Individuen; links: MPEA1, Mitte: SPEA2, rechts: LEA3

Wenn diese Funktion als eine praktische Aufgabe betrachtet wird, ist das Problem allerdings bereits gelöst, wenn die Variablen in der Nähe des Optimums sind. In diesem Fall bedeutet dies ein Erreichen der Zielwerte [5 10 15 ... 50] für die einzelnen Variablen. Bei einer genauen Betrachtung der Grafiken in Abb. 8-12 wird dieser Bereich bei allen Verfahren nach etwa 200 Generationen erreicht. Zu diesem Zeitpunkt ist das Problem aus ingenieurtechnischer Sicht gelöst. Die in Abb. 8-11 scheinbar recht deutlichen Unterschiede zwischen den Verfahren verringern sich dadurch deutlich. Dies sollte bei einem Vergleich verschiedener Verfahren mit in Betracht gezogen werden.



**Abb. 8-13.** Optimierung von *moved axis parallel hyper-ellipsoid* Funktion, Zielfunktionswerte aller Individuen; links: MPEA1, Mitte: SPEA2, rechts: LEA3

Abbildung 8-13 zeigt die Zielfunktionswerte aller Individuen während der Optimierung. Die linke und die rechte Grafik sehen ähnlich wie in Abb. 8-9 aus. Ganz anders fällt dieser Vergleich für die mittlere Grafik aus. Im ersten Augenblick scheint dies ein Widerspruch zu sein. Bei einer genaueren Betrachtung der Charakteristik der zu optimierenden Funktion löst sich dieser aber auf. In den ersten 150 Generationen verbessern sich die Zielfunktionswerte in Abb. 8-13 Mitte in einem großen Wertebereich. In den weiteren Generationen ist die weitere Verbesserung durch die verwendete lineare Skalierung kaum noch zu erkennen. Dadurch fallen die Unterschiede in den Zielfunktionswerten einer Generation kaum auf. Zudem arbeitet SPEA2 mit relativ kleinen Mutationsschritten. Bei MPEA1 führen die durch Unterpopulation 1 erzeugten großen Mutationen zu relativ schlechten Individuen, die trotz der linearen Skalierung immer zu erkennen sind, Abb. 8-13 links.



**Abb. 8-14.** Optimierung von *moved axis parallel hyper-ellipsoid* Funktion, Anwendung verschiedener Strategien und Konkurrenz zwischen Unterpopulationen bei MPEA1; links: Ordnung der Unterpopulationen, rechts: Größe der Unterpopulationen

Bisher wurde nicht untersucht, wie sich die unterschiedlichen Strategien im Verfahren MPEA1 verhalten. Die linke Grafik in Abb. 8-14 zeigt die Ordnung

der Unterpopulationen. Daraus ist zu entnehmen, daß Unterpopulation 3 während der ersten 300 Generationen am erfolgreichsten ist. Dies bestätigt ein Blick auf Abb. 8-14 rechts. Unterpopulation 3 ist bis nach Generation 300 am größten und hat teilweise 85 Individuen. Nur während der Generationen 50 bis 70 ist Unterpopulation 1 erfolgreich und wächst kurzzeitig bis auf 40 Individuen an. Zum Ende der Optimierung wird die Strategie mit den kleinsten Mutationsschritten am erfolgreichsten, Unterpopulation 4 wächst kontinuierlich an.

### **8.2.5 Zusammenfassung mehrdimensionale Funktionen**

In diesem Abschnitt wurde die Optimierung von mehrdimensionalen mathematischen Funktionen gezeigt. Die verwendeten Funktionen sind bekannte Standardtestfunktionen. Die Darstellungen dienten der Einführung in die praktische Optimierung mit Evolutionären Algorithmen. Die Beispiele können vom Anwender leicht nachvollzogen werden. Außerdem wurde der Einsatz der Visualisierungsmethoden zur Auswertung und zum Vergleich von Optimierungsläufen gezeigt.

Sehr deutlich zeigten sich bei allen Optimierungen die Unterschiede zwischen den beiden global orientierten Verfahren MPEA1 und SPEA2 und dem lokal orientierten Verfahren LEA3. Ein Verfahren wie LEA3 ist immer dann besonders gut geeignet, wenn eine kontinuierliche Veränderung der Variablen zum Erfolg führt. Dies zeigt sich sehr deutlich bei der Optimierung von ROSENROCK's Funktion und der *moved axis parallel hyper-ellipsoid* Funktion. Für die Lösung einer multimodalen Funktion mit vielen lokalen Minima wie RASTRIGIN's Funktion ist LEA3 dagegen nicht geeignet.

Die global orientierten Verfahren MPEA1 und SPEA2 zeigen ein sehr ähnliches Verhalten. Dies ist nicht verwunderlich, da sie in den verwendeten Operatoren und Optionen ähnlich sind. Beide Verfahren sind nicht in der Lage, eine Funktion mit korrelierten Variablen vernünftig zu lösen. Dafür bereiten ihnen multimodale Funktionen nur geringe Probleme.

Bei den in diesem Abschnitt eingesetzten Testfunktionen konnte die Anwendung verschiedener Strategien in MPEA1 nicht so deutlich ihre Vorteile gegenüber SPEA2 zeigen. Dies liegt insbesondere daran, daß SPEA2 für die Testfunktionen gut genug ist. Bei der Lösung komplexer Probleme, bei denen nur wenig über das Verhalten der Zielfunktion bekannt ist, zeigen sich die Vorteile der Anwendung verschiedener Strategien deutlicher.

## **8.3 Parameteridentifikation eines Dieselmotormodells**

In diesem Abschnitt wird die Optimierung der Parameter eines Modells der Verbrennung für direkteinspritzende Dieselmotoren vorgestellt. Das Ziel ist die Ermittlung einer Parametrisierung des Modells, die eine sehr gute Übereinstimmung zwischen den im Versuch gemessenen und den in der Simulation ermittelten Zuständen für alle zur Verfügung stehenden Arbeitspunkte erreicht.

Für die Simulation der Verbrennungsvorgänge stand ein Simulationsprogramm zur Verfügung, das sich in bisherigen Untersuchungen als geeignet er-

wiesen hatte. Es bereitete allerdings Schwierigkeiten, die freien Parameter dieses Simulationsprogrammes an einen speziellen Motor anzupassen. Diese Aufgabe stellt ein hochkomplexes Optimierungsproblem dar, das mit deterministischen Verfahren bisher nicht bewältigt werden konnte.

In Voruntersuchungen wird das Verhalten des Simulationsprogrammes untersucht und die Ergebnisse grafisch dargestellt. Aus diesen Voruntersuchungen lassen sich Richtlinien für die Einsatzfähigkeit bestimmter Optimierungsverfahren ableiten.

Durch die Anwendung Evolutionärer Algorithmen als Optimierungsverfahren können alle für die Bewertung des Systemverhaltens relevanten Größen in die Zielfunktion aufgenommen werden.

Für die Optimierung standen 14 Datensätze (Szenarien) eines Motors zur Verfügung. Diese Szenarien decken den Arbeitsbereich des Motors ab.

Die in diesem Abschnitt beschriebenen Arbeiten wurden zusammen mit der DaimlerChrysler Forschung in Stuttgart durchgeführt. Eine ausführliche Darstellung der Untersuchungen und Ergebnisse erfolgte in [PS98].

### 8.3.1 Beschreibung des Systems

Das verwendete Simulationsmodell (eine ausführliche Beschreibung ist in [Sum95] enthalten) beschreibt den Verbrennungsvorgang in direkteinspritzenden Dieselmotoren. Die Modellierung bezieht sich auf die Wirkung der Wärmefreisetzung auf den Zylinderdruck während des Verbrennungsvorganges. Das Verbrennungsmodell stellt ein leicht zu handhabendes Computerprogramm dar, das mit besonderem Augenmerk auf kurze Rechenzeit entwickelt wurde.

Das Verhalten des Verbrennungsmodells wird durch die Vorgabe der folgenden Werte bzw. Datendateien beeinflusst:

1. Daten des verwendeten Motors (Einlesen aus Datei \*.dat),
2. Daten des Einspritzvorganges (Einlesen aus Datei \*.esp) und
3. Modellparametern zur Beeinflussung der einzelnen Prozesse des Verbrennungsvorganges (Übergabe als Programmparameter).

Die Dateien mit den Daten des Motors sowie des Einspritzvorganges werden von den Entwicklern des Verbrennungsmodells mitgeliefert (Datenmaterial aus Simulationen bzw. Experimenten, die für die Optimierung als Trainings- und Testdaten zur Verfügung stehen müssen). Für den Vergleich der Simulation mit dem Verhalten des realen Motors wird eine weitere Datei mitgeliefert, welche die Meßwerte mit dem Verlauf des Zylinderdrucks in Abhängigkeit vom Winkel enthält (Datei \*.mes). Die 3 Dateien stellen zusammengenommen jeweils einen Datensatz bzw. ein Szenario dar.

Die Modellparameter zur Beeinflussung der Prozesse des Verbrennungsvorganges sind die Werte, mit denen das Modell in seinem Verhalten an einen speziellen Datensatz bzw. eine Anzahl von Datensätzen (und damit an einen bestimmten Motor) angepaßt werden kann.

Die Bestimmung dieser Modellparameter ist Aufgabe der Optimierung. Als Grundlage des Vergleichs wird die Datei mit den Druckmeßwerten verwendet, die mit dem durch die Simulation ermittelten Verlauf des Zylinderdruckes vergli-

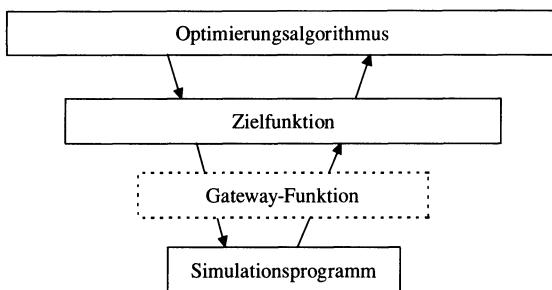
chen wird. Aus diesem Vergleich und unter Einbeziehung weiterer Größen wird eine Zielfunktion (Gütfunktion) aufgestellt, die eine Bewertung der Übereinstimmung zwischen realen Meßwerten und simulierten Werten ermittelt. Die Aufstellung und Definition der Zielfunktion wird in Unterabschn. 8.3.3 detailliert beschrieben.

Das verwendete Verbrennungsmodell wird über 7 Parameter gesteuert, die jeweils einen konstanten Faktor zur Beeinflussung eines Teilprozesses darstellen:

- Parameter 1: Gemischbildung (Basiskonstante),
- Parameter 2: Drallabhängigkeit,
- Parameter 3: Gemischbildungsgrad (Einspritzung),
- Parameter 4: Kolbengeschwindigkeit,
- Parameter 5: Wandanlagerung (Wandwechselwirkung),
- Parameter 6: Zündverzug,
- Parameter 7: Wandwärme.

Bei der Präsentation und Diskussion der Ergebnisse in den folgenden Kapiteln wird jeweils nur von der entsprechenden Parameternummer gesprochen.

Das Modell für die Verbrennung in Dieselmotoren liegt in Form eines FORTRAN-Programms vor. Dieses Programm ist ein Simulationsprogramm. Für einen vorgegebenen Satz der 7 Parameter (1. Eingabeparameter für die Simulation) führt es unter Zuhilfenahme des ausgewählten Datensatzes (2. Eingabeparameter<sup>2</sup>) eine Simulation des Verbrennungsvorganges durch. Der Verlauf und die Endzustände dieses Verbrennungsvorganges (Ausgabeparameter der Simulation) können im Anschluß an die Simulation untersucht werden.



**Abb. 8-15.** Struktur des Programmaufrufs

Wie bei jeder Optimierung ist vom Anwender eine Zielfunktion zu schreiben, welche die Bewertung von Lösungen durchführt. Innerhalb dieser Zielfunktion erfolgt der Aufruf des Simulationsprogrammes für die Verbrennung. Abbildung 8-15 stellt diesen mehrstufig strukturierten Aufruf dar und zeigt die Ab-

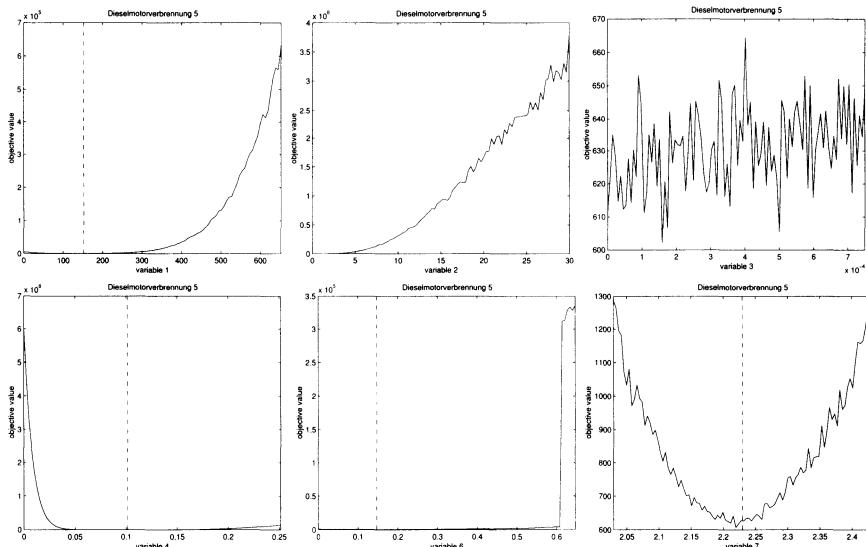
<sup>2</sup> String mit dem Namen der zu verwendenden Datendatei: z.B. '0850\_04' → Verwendung der Dateien 0850\_04.dat, 0850\_04.mes und 0850\_04.esp; die ersten vier Ziffern stehen für die Umdrehungen des Motors pro Minute, hier 850 U/min; die letzten 2 Ziffern für den (Einspritz-)Druck, hier 4 bar.

hängigkeiten zwischen den einzelnen Ebenen. Eine Einbindung des Simulationsprogramms unter MATLAB kann durch Verwendung einer FORTRAN-Gateway-Funktion geschehen. Das Simulationsprogramm wird von der Gateway-Funktion als Unterprogramm aufgerufen. Dadurch ist keine Übergabe von Parametern auf der Kommandozeile oder über eine Dateischnittstelle notwendig. Die Steuerung der Simulation erfolgt von innerhalb des Optimierungsprogramms. Genauso werden die Ergebnisse nicht in Textdateien ausgegeben, sondern in Matrizen an die Gateway-Funktion und nachfolgend an die Zielfunktion im Optimierungsprogramm übergeben, in der die Auswertung der Ergebnisse und die Berechnung der Zielfunktionswerte erfolgt.

Durch die direkte Ankopplung des FORTRAN-Programms an MATLAB kann das originale Simulationsprogramm eingesetzt werden. Damit ist eine Neuimplementierung des Modells nicht notwendig, sondern das Originalprogramm, das sich in bisherigen Tests bewährt hat, wird direkt für die Optimierung verwendet. Dies ist ein wichtiger Aspekt für die Akzeptanz der Optimierung und ihrer Ergebnisse.

### 8.3.2 Untersuchungen des Systemverhaltens

Zu Beginn der Arbeit mit dem Verbrennungsmodell und im weiteren Verlauf der durchgeführten Optimierungen wurden verschiedene Untersuchungen zum Systemverhalten des Verbrennungsmodells durchgeführt. Diese dienten dazu, mehr Information über das Verhalten des Modells zu erhalten.



**Abb. 8-16.** Eindimensionale Schnitte durch das Gütegebirge des Modells der Verbrennung mit einer Variation der Parameter um 10% des entsprechenden Suchbereiches (ohne Parameter 5)

In den für die Verbrennungssimulation durchgeführten Untersuchungen wurde meist mit eindimensionalen Schnitten gearbeitet, nur an einigen Punkten wurden zusätzlich zweidimensionale Schnitte berechnet. Hier werden die Ergebnisse für Schnitte um einen Punkt präsentiert. Bei anderen Punkten wurden qualitativ gleichwertige Ergebnisse erhalten.

In den Abb. 8-16 und 8-17 sind für die einzelnen Parameter jeweils eindimensionale Schnitte durch das Gütegebirge um den Punkt [152 0,00097 0 1014 648 0,015 2,23] zu sehen. Dieser Punkt ist der Mittelpunkt für die folgenden eindimensionalen Schnitte und war das Ergebnis einer der durchgeführten Optimierungen. In Abb. 8-16 wurden die Parameter jeweils um 10% des Suchbereiches variiert, in Abb. 8-17 dagegen nur um 1%. Die senkrechte gestrichelte Linie zeigt den Mittelpunktwert des variierten Parameters an, der für die jeweils anderen Grafiken verwendet wurde<sup>3</sup>.

Durch die relativ starke Variation der Parameter in Abb. 8-16 wird ein größerer Teil des Suchbereiches abgetastet. Die einzelnen Grafiken zeigen sehr deutlich die unterschiedliche Empfindlichkeit der Parameter. In fast allen Grafiken ist eine sehr starke Änderung der Zielfunktionswerte zu sehen, wenn die Parameter an die Grenzen des dargestellten Intervalls kommen. Außerdem ist bei den Parametern 1, 2, 4, 6 und 7 gut zu erkennen, daß der Zielfunktionswert um so größer wird, je weiter sich der Parameter vom Mittelpunkt entfernt. Eine Ausnahme bildet Parameter 3 (Gemischbildungsgrad (Einspritzung)). Bei diesem ist der Zusammenhang zwischen Entfernung vom Mittelpunkt und Zielfunktionswert nicht zu erkennen. Der Verlauf sieht wie eine starke Störung bzw. stark verrauscht aus.

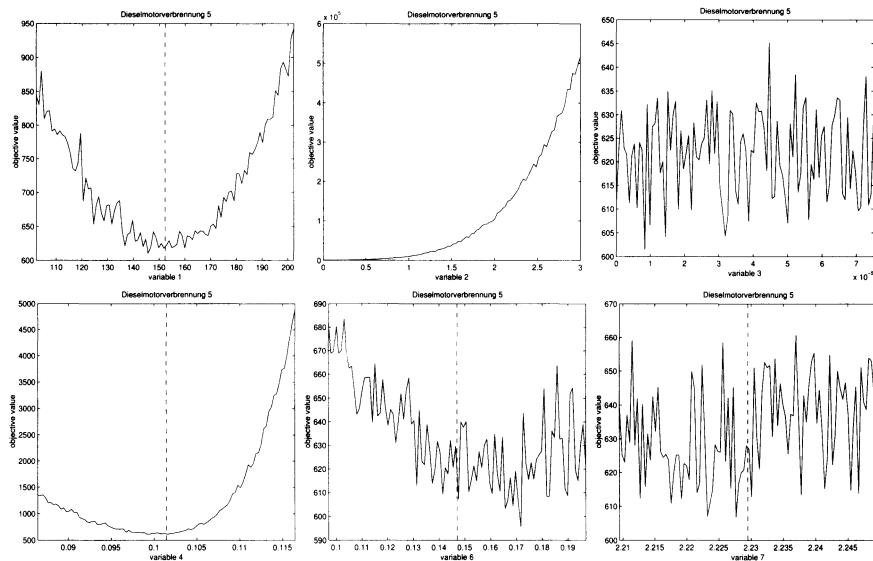
Eine Sonderstellung nimmt Parameter 5 (Wandanlagerung) ein. Dieser Parameter hatte bei den hier verwendeten Datensätzen keinen Einfluß auf die Zielfunktionswerte. Eine mögliche Konsequenz wäre, daß bei dem hier verwendeten Beispiel die Dimension des Problems auf sechs Parameter verringert werden könnte.

In Abb. 8-17 sind eindimensionale Schnitte um denselben Punkt wie in Abb. 8-16 dargestellt. Allerdings wurden die Parameter hier nur um 1% des Suchbereiches variiert. Diese Grafiken stellen damit ein feiner aufgelöstes Bild des Zentrums der Grafiken aus Abb. 8-16 dar und zeigen den unmittelbaren Nahbereich des verwendeten Mittelpunktes.

Neben der aus diesen Grafiken ableitbaren Metastruktur für die Parameter 1, 2 und 4, sieht es für die Parameter 3, 6 und 7 im Nahbereich etwas anders aus. Ein Zusammenhang zwischen Entfernung vom Mittelpunkt und größer werdendem Zielfunktionswert ist nicht oder kaum erkennbar. Eine gleichmäßige Veränderung dieser Parameter führt nicht zu einer korrespondierenden gleichmäßigen Änderung der Zielfunktionswerte. Statt dessen ist der Zusammenhang stochastischer Natur. Kleine Änderungen der Parameter können starke Änderungen der Zielfunktionswerte zur Folge haben. Weiter voneinander entfernt liegende Parameterwerte können dagegen ähnliche Zielfunktionswerte besitzen. Diese chaoti-

<sup>3</sup> Normalerweise liegt der Mittelpunktwert in der Mitte des variierten Bereiches. Wenn allerdings die Grenze des Variationsbereiches außerhalb des Suchbereiches für diese Variable liegt, so wird der Variationsbereich entsprechend dem Suchbereich beschränkt. Dadurch kommt es vor, daß der Mittelpunktwert gleichzeitig an der Grenze des Variationsbereiches liegt.

sche Natur der Abhangigkeit zwischen Zielfunktionswert und einigen Parametern erschwert die Optimierung.



**Abb. 8-17.** Eindimensionale Schnitte durch das Gutegebirge des Modells der Verbrennungssimulation mit einer Variation der Parameter um 1% des entsprechenden Suchbereiches (ohne Parameter 5)

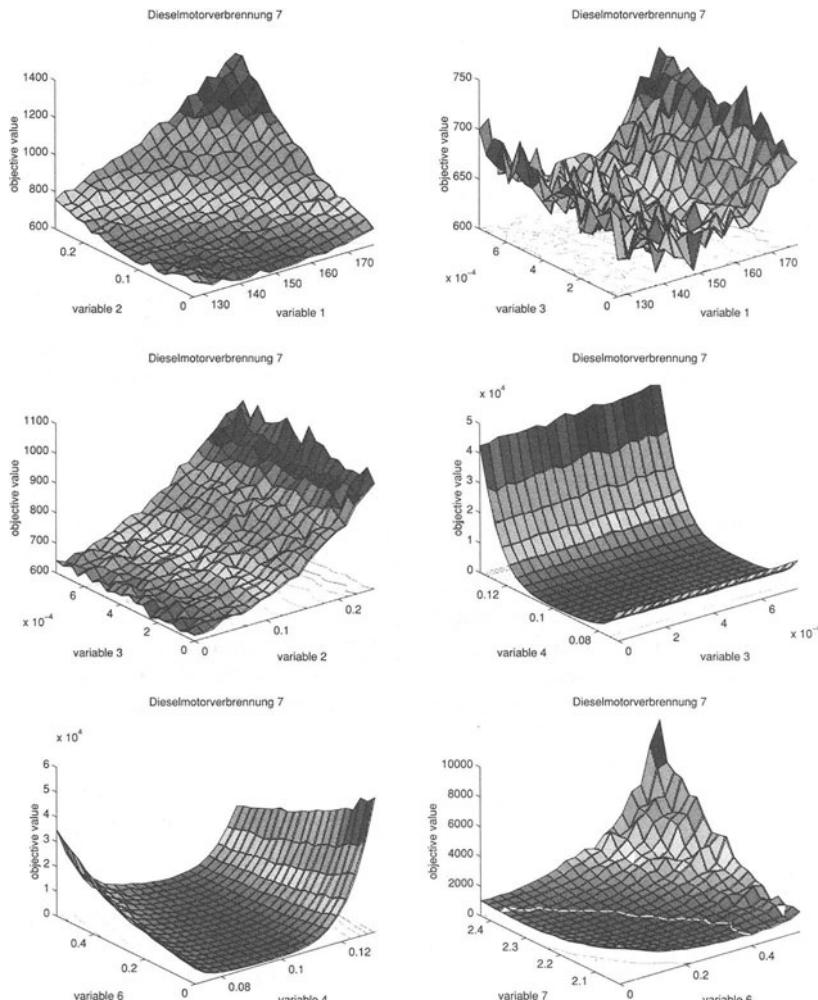
Zum Abschlu dieser Untersuchungen wurden zweidimensionale Schnitte durch das Gutegebirge angefertigt. Unter Verwendung der Ergebnisse der eindimensionalen Schnitte wird der Variationsbereich der verwendeten Variablen unterschiedlich festgelegt, um eine gute Skalierung der Ergebnisse zu erreichen. Zur Erzeugung der zweidimensionalen Schnitte werden jeweils zwei Parameter im Variationsbereich um den Mittelpunkt gleichmig verandert und die Zielfunktionswerte fur die zugehorigen Parameterwerte berechnet. Die Ergebnisse sind in einem 3-D Oberflachendiagramm mit darunterliegendem Konturdiaagramm dargestellt.

Bei der Auswertung der Grafiken in Abb. 8-18 sind einige interessante Ergebnisse erkennbar. Zwischen den Parametern 1 und 2, 6 und 7 sowie in geringerem Mae zwischen den Parametern 4 und 6 scheint eine Korrelation zu bestehen. Parameter 3 ist auch in diesen Grafiken relativ unempfindlich, so da keine Auswirkungen durch die Anderung dieses Parameters zu erkennen sind. Parameter 5 wurde hier nicht dokumentiert, da eine Anderung dieses Parameters keine Auswirkungen hatte. Da fur die Erstellung der Grafiken in Abb. 8-18 die Parameter nur grob gerastert wurden (20 Punkte je Parameter, dadurch 400 Punkte pro Grafik), ergibt sich eine relativ glatte Flache in den Grafiken.

Die Grafiken in Abb. 8-17 haben sehr deutlich gezeigt, da das Gutegebirge im Nahbereich chaotisch bzw. rauh ist. Im Fernbereich dagegen, der in den Abb. 8-16 und 8-18 dargestellt ist, lsst sich eine Metastruktur erkennen, die auf

ein zwar relativ tiefes Tal hindeutet, aber keine großen Probleme für ein entsprechend angepaßtes Optimierungsverfahren darstellen dürfte. Die Störungen im Nahbereich sind deutlich kleiner als die Unterschiede der Zielfunktionswerte im Fernbereich.

Eine grob bis mittel genaue Suche (dies entspricht einer stärker global orientierten Suche) müßte demnach zu guten Ergebnissen gelangen. Im Nahbereich und damit im Bereich um ein gefundenes Optimum, kann es dagegen zu sichtbaren Unterschieden in den Zielfunktionswerten kommen, obwohl die Parameter nur wenig verändert wurden.



**Abb. 8-18.** Zweidimensionale Schnitte durch das Gütegebirge des Modells der Verbrennungssimulation (Variationsbereich = [5% 0,5% 10% 2% 10% 8% 10%])

Die Untersuchungen des Modells der Verbrennung für direkteinspritzende Dieselmotoren konnten einen Ausschnitt aus den Eigenschaften des Systems beleuchten. Aus diesen Teilen lassen sich einige Schlußfolgerungen bzw. Vermutungen über die Art und das Aussehen des Gütegebirges ableiten:

- Im Nahbereich zeigt die Gütfunktion stark chaotisches Verhalten. Die Ursache dieser Eigenschaft liegt in der chaotischen Reaktion des Verbrennungsmodells auf sehr kleine Variationen in den Parametern (hier nicht näher gezeigt).
- Die Größe der chaotischen Reaktionen im Nahbereich ist klein verglichen zu den Unterschieden in den Zielfunktionswerten bei größeren Variationen der Parameter.
- Im Fernbereich konnten keine großen lokalen Minima gefunden werden.

Aus diesen Ergebnissen lassen sich die folgenden Forderungen an die Eigenschaften des zu verwendenden Optimierungsalgorithmus ableiten:

- Es muß eine global orientierte Suche durchgeführt werden.
- Der Optimierungsalgorithmus darf nicht empfindlich gegenüber einer rauen Oberfläche des Gütegebirges sein.
- Die Verwendung eines approximierten Gradienten ist nicht möglich.
- Es muß nicht verstärkt nach mehreren möglichen Minima gesucht werden.

Auf die Umsetzung dieser Forderungen und ihren Einfluß auf den verwendeten Algorithmus für die Optimierung wird in Abschn. 8.3.4 eingegangen.

### 8.3.3 Zielfunktion für die Optimierung

Für die Bewertung einer Lösung stehen die folgenden Größen für jeden simulierten Meßdatensatz zur Verfügung:

- Verlauf des gemessenen Zylinderdruckes  $P_{zM}$  (aus der Datendatei \*.mes),
- Verlauf des durch die Simulation ermittelten Zylinderdruckes  $P_{zs}$  (Ergebnis der Simulation, wird vom Simulationsprogramm in einer Matrix zurückgegeben),
- Differenz zwischen eingespritzter,  $m_s$ , und verbrauchter Brennstoffmenge,  $m_b$  (Ergebnis der Simulation, wird als zusätzlicher Zielfunktionswert zurückgegeben).

Aus diesen zur Verfügung stehenden Größen werden mehrere Zielfunktionswerte gebildet. Dabei wird von einer Minimierung ausgegangen (kleinere Werte sind besser als größere Werte):

- Integral der Differenz zwischen den Verläufen des gemessenen und des durch die Simulation ermittelten Zylinderdruckes (integrale Druckdifferenz),
- Maxima der Differenz der obigen Druckverläufe (maximale Druckdifferenz),
- Differenz zwischen eingespritzter,  $m_s$ , und verbrauchter Brennstoffmenge,  $m_b$  (Kraftstoffdifferenz).

Der Verlauf des Zylinderdrucks liegt jeweils als abgetastete Größe mit einem Abtastabstand von  $1^\circ$  vor.

Berechnung und Bewertung der Abweichung zwischen den Verläufen des gemessenen und des durch die Simulation ermittelten Zylinderdruckes (integrale Druckdifferenz):

$$SumDiff_{P_z} = \sum_{\varphi=\varphi_{Start}}^{\varphi=\varphi_{Ende}} \left| P_{z_M}(\varphi) - P_{z_S}(\varphi) \right|^{Expo_{Druck}} \quad \varphi \in [0^\circ, 720^\circ], \quad Expo_{Druck} \in [1, 2, 3, \dots] \quad (8-2)$$

$Expo_{Druck} = [1, 2, 6]$  wurden verwendet

Durch die Wahl von  $Expo_{Druck}$  können die Differenzen mit unterschiedlichen Potenzen bewertet werden (höhere Potenzen bewerten große Differenzen stärker als kleine Differenzen). Eine Normierung der Differenz der Drücke auf den aktuellen Druck wird nicht vorgenommen. Diese hätte zur Folge, daß Differenzen bei hohen Drücken weniger stark berücksichtigt werden. Dieser Fall ist aber nicht erwünscht, da gerade bei hohen Drücken eine gute Übereinstimmung erreicht werden soll. Die direkte Bewertung der absoluten Differenzen führt zu einer gleichbleibenden absoluten Gewichtung auch bei hohen Drücken.

Berechnung der maximalen Differenz zwischen den Verläufen des gemessenen und des durch die Simulation ermittelten Zylinderdruckes (maximale Druckdifferenz):

$$MaxDiff_{P_z} = \max \left( \left| P_{z_M}(\varphi) - P_{z_S}(\varphi) \right| \right) \quad \varphi \in [0^\circ, 720^\circ] \quad (8-3)$$

Eine Einbeziehung von  $MaxDiff_{P_z}$  ermöglicht eine direkte Bewertung der maximalen Differenz zwischen gemessenem und simuliertem Verlauf des Zylinderdruckes. Dies ist besonders dann notwendig, wenn eine große Abweichung zwischen den beiden Verlaufskurven für kurze Zeit auftritt.

Berechnung der Differenz zwischen eingespritzter,  $m_s$ , und verbrauchter Brennstoffmenge,  $m_b$ , (Kraftstoffdifferenz):

$$Diff_{m_{sb}} = m_s - m_b \quad (8-4)$$

Die Kraftstoffdifferenz ist ein wichtiges Kriterium für die Bewertung der Qualität der Verbrennung. Am Ende der Verbrennung sollte nur noch eine kleine Menge oder kein Brennstoff verblieben sein.

Für die Bewertung des Verbrennungsvorganges stehen damit 3 Größen zur Verfügung:

1. Abweichung zwischen den Verläufen des gemessenen und des durch die Simulation ermittelten Zylinderdruckes,  $SumDiff_{P_z}$ , Gl. 8-2,
2. maximale Differenz zwischen den Verläufen des gemessenen und des durch die Simulation ermittelten Zylinderdruckes,  $MaxDiff_{P_z}$ , Gl. 8-3,
3. Differenz zwischen eingespritzter und verbrauchter Brennstoffmenge,  $Diff_{m_{sb}}$ , Gl. 8-4.

Für die eigentliche Bewertung in der Optimierung wird allerdings nur die erste dieser drei Größen verwendet. Die Verwendung der maximalen Druckdifferenz,  $MaxDiff_{P_z}$ , ist nicht notwendig, da bei den Optimierungen meist mit einem  $Expo_{Druck}$  von 6 für die Berechnung der integralen Druckdifferenz,  $SumDiff_{P_z}$ , ge-

arbeitet wird. Der hohe Wert für  $Expo_{Druck}$  führt zu einer verstärkten und in diesem Falle ausreichenden Bewertung großer Abweichungen. Nähere Erläuterungen zur Wahl von  $Expo_{Druck}$  und den Auswirkungen verschiedener Werte für  $Expo_{Druck}$  werden auf S.215 gegeben. Auch die Differenz zwischen zugeführtem und verbranntem Kraftstoff (Kraftstoffdifferenz) wird nicht in die Bewertung für die Optimierung einbezogen, da es für die vorgenommenen Untersuchungen nicht notwendig ist. Allerdings wird die Kraftstoffdifferenz als zusätzliche Information mit ausgegeben und ist in der Ergebnisdokumentation aufgeführt.

Zum Abschluß erfolgt eine Zusammenfassung der zu bewertenden Größen durch eine gewichtete Summenfunktion. Alternativ dazu könnte eine mehrkriterielle Bewertung entsprechend Unterabschn. 3.1.3 verwendet werden. Allerdings ist diese nur wenig besser zu überblicken, da Zielbereiche (*goals*) für die einzelnen Werte hier nicht direkt definierbar sind.

Da im Endeffekt nur eine der Größen pro Meßdatensatz,  $SumDiff_{P_z}$ , verwendet wird, erhält diese Größe einen Gewichtungswert von 1. Die anderen beiden Größen werden mit 0 gewichtet und sind dadurch nicht in die Bewertung einbezogen<sup>4</sup>:

$$\text{Güte} = W_1 \cdot SumDiff_{P_z} + W_2 \cdot MaxDiff_{P_z} + W_3 \cdot Diff_{m_{ab}} \quad W = [1, 0, 0] \quad (8-5)$$

Mit Gl. 8-5 wird der Zielfunktionswert für einen Meßdatensatz berechnet. Die Optimierung wird über mehrere Meßdatensätze durchgeführt, so daß mehrere Gütwerte erhalten werden (*multiple objective values*). Eine Zusammenfassung dieser einzelnen Werte erfolgt durch Addition der einzelnen Werte. Eine weitere Gewichtung der Gütwerte für die einzelnen Datensätze untereinander findet dabei nicht statt, da alle Meßdatensätze dieselbe Wertigkeit besitzen.

### 8.3.4 Verwendetes Optimierungsverfahren

Auf Grund der in den Voruntersuchungen erhaltenen Ergebnisse, muß ein global orientiertes Optimierungsverfahren eingesetzt werden. Es gelten die Hinweise, die in Abschn. 7.2 gegeben werden.

Im einzelnen wurde mit den folgenden Operatoren und Optionen gearbeitet:

- 3 Unterpopulationen mit je 40 Individuen über 150 Generationen,
- *generation gap* von 0,8 (pro Generation werden  $3 \cdot 40 \cdot 0,8 = 96$  Nachkommen produziert), alle Nachkommen werden in die Population eingefügt,
- Fitneßzuweisung durch lineares Ranking, Selektionsdruck von 1,5, Selektion durch *stochastic universal sampling*,
- diskrete oder Linien-Rekombination,
- reelle Mutation mit Range von 0,1 / 0,01 und Präzision von  $2^{-12} / 2^{-8}$ ,
- Anwendung von verschiedenen Kombinationen der Rekombinations- und Mutationsoperatoren in den einzelnen Unterpopulationen.

<sup>4</sup> Bei der Einbeziehung mehrerer Größen in die Zielfunktion werden die Gewichtungsfaktoren  $W$  für jede der Größen so festgelegt, daß die Größenordnung der einzelnen Werte aufeinander abgestimmt ist bzw. deren relativen Wichtigkeit entspricht.

Der Vorteil dieser Operatoren und Optionen liegt in der großen Robustheit bei der Suche. Dieser Evolutionäre Algorithmus ist deshalb gut für die grobe Suche geeignet, wobei er durch den Einsatz verschiedener Strategien am Ende auch gut bei der lokalen Suche ist. Die Verwendung einer größeren Anzahl von Individuen und mehrerer voneinander zeitweise isolierter Unterpopulationen stellt eine weitgehende Durchmusterung des Suchraumes sicher und verhindert gleichzeitig die zu schnelle Einschränkung auf ein kleines Gebiet des Suchraumes zu Beginn der Optimierung.

### **8.3.5 Problemspezifische Visualisierung**

Für eine komfortable Auswertung des Verlaufs und der Ergebnisse der Optimierungen wurde eine grafische Oberfläche unter Matlab geschaffen, s. Abb. 8-19. Sie stellt ein Beispiel der problemspezifischen Visualisierung dar, auf die schon an mehreren Stellen eingegangen wurde.

Im einzelnen sollen damit die folgenden Aufgaben erfüllbar sein:

- Auswertung des Verlaufs der Optimierung durch Darstellung von:
  - Verlauf der Zielfunktionswerte,
  - Verlauf der Kraftstoffdifferenz.
- Bewertung der Qualität einzelner Lösungen durch Darstellung der folgenden Größen (interaktive Simulation):
  - Verlauf des gemessenen Zylinderdruckes,
  - Verlauf des simulierten Zylinderdruckes,
  - Verlauf der Differenz zwischen den beiden Druckverläufen,
  - Angabe der einzelnen Zielfunktionswerte inklusive Kraftstoffdifferenz.

Für beide Gruppen von Ausgaben muß die Darstellung für alle verwendeten bzw. möglichen Datensätze erfolgen oder auf eine Anzahl von ausgewählten Datensätzen beschränkt werden können.

Die für die Optimierung verwendete GEATbx bietet die Option, während der Optimierung alle für eine spätere Auswertung wichtigen und notwendigen Daten in einer binären Datei (mat-Datei: Matlab spezifisches Format) zu speichern. Auf diese Datei kann in der nachfolgenden Auswertung zugegriffen werden. Mit den Daten aus dieser binären Ergebnisdatei sowie der für die Optimierung verwendeten Zielfunktion können alle benötigten Ergebnisse und Daten eines Optimierungslaufes erzeugt bzw. reproduziert werden.

Obwohl die Optimierung des Dieselmotormodells nur 7 Parameter umfaßt, ergibt sich in der Auswertung ein komplexes Problem. Der Verlauf mehrerer Größen ist zu bewerten. Alle diese Informationen sollten und wurden in der grafischen Darstellung untergebracht. Deshalb an dieser Stelle einige Informationen zum Inhalt der Grafiken.

Die Simulationsgrafiken zeigen jeweils den Verlauf des gemessenen und simulierten Zylinderdruckes. Um bei einer guten Übereinstimmung beider Kurven trotzdem einen vernünftigen Einblick in die Abweichungen zwischen beiden Kurven zu erhalten, wird die Differenz beider Kurven zusätzlich in die Grafik mit eingezeichnet. Da die Größenordnung der Differenz zwischen beiden Kurven vom Betrag meist klein ist, wird eine Skalierung auf den Wertebereich der

Druckkurven vorgenommen. Die genaue Skalierung ist für jede Simulationsgrafik anders und muß der jeweiligen Legende entnommen werden. Dabei wird die Skalierung so durchgeführt, daß möglichst der gesamte zur Verfügung stehende Bereich ausgenutzt wird. Die Differenzkurve selbst darf damit nur zur qualitativen Bewertung verwendet werden. Die Größenordnung der Differenz zwischen den Druckkurven kann nur aus der Legende abgelesen werden. Eine Legendenbeschriftung von: Dif. \*5+30 (Abb. 8-19 linke obere Grafik) bedeutet, daß die Differenzwerte mit 5 multipliziert (skaliert) und dann 30 dazu addiert wurden (verschieben). Um also den wahren Differenzwert zu ermitteln muß zuerst 30 abgezogen und dann durch 5 dividiert werden.

Der Titel jeder Simulationsgrafik enthält den Namen des zugrunde liegenden Datensatzes, danach die ersten drei Zielfunktionswerte (für  $Expo_{Druck}$ : 1, 2 und 6) und am Ende den Wert der Kraftstoffdifferenz (vierter Zielfunktionswert)<sup>5</sup>. Als Beispiel soll der Titel der linken oberen Grafik verwendet werden: 1200\_04 : 4.7/1.47/324 KSD: 2.16. Es wurde der Datensatz 1200\_04 verwendet, die Zielfunktionswerte lauten 4,7 (für  $Expo_{Druck} = 1$ ), 1,47 (für  $Expo_{Druck} = 2$ ) und 324 (für  $Expo_{Druck} = 6$ ) und die Kraftstoffdifferenz KSD beträgt 2,16%. Damit sind alle wichtigen Informationen in dieser Grafik enthalten.

Mit der erstellten Benutzeroberfläche ist eine komfortable Ergebnisdarstellung für die Bewertung von Lösungen möglich. Gleichzeitig kann damit eine Dokumentation der Ergebnisse auf grafischem Weg erfolgen. Alle Grafiken in Unterabschn. 8.3.6 wurden mit dieser Oberfläche erzeugt. Zusätzlich sei erwähnt, daß dieselbe Oberfläche Verwendung findet, wenn während einer Optimierung eine grafische Bewertung der aktuell besten Lösung stattfinden soll. Außerdem ist eine Steuerung der Oberfläche aus Matlabskripten heraus möglich (Öffnen der Oberfläche, Laden einer Datei, Anzeige der Grafiken, Speichern der Ergebnisse), die bei der Erstellung von Demos genutzt wird.

### 8.3.6 Ergebnisse der Optimierungen

Dieses Kapitel zeigt ausgewählte Ergebnisse, die durch die Optimierungen der Parameter des Verbrennungsmodells ermittelt wurden. Ziel dieser Arbeiten war mittels Optimierung eine Lösung für die Parametrisierung des Verbrennungsmodells für eine größere Anzahl von Datensätzen zu finden.

Für die Optimierungen standen eine Anzahl von Datensätzen zur Verfügung: 0850\_04, 0850\_09, 0850\_12, 1200\_04, 1200\_09, 1200\_12, 1200\_15, 1600\_04, 1600\_09, 1600\_13, 1600\_15, 1900\_04, 1900\_09, 1900\_12.

Die in Abschn. 8.3.3 beschriebene Zielfunktion kam bei der Optimierung zur Anwendung. Es wurde allgemein mit einem  $Expo_{Druck} = 6$  gearbeitet. In Voruntersuchungen hatte sich gezeigt, daß kleinere Werte für  $Expo_{Druck}$  zu einer zu ge-

<sup>5</sup> Nach der Simulation eines Datensatzes werden alle 4 Zielfunktionswerte (3 Werte für unterschiedliche  $Expo_{Druck}$ , 1 Wert für die Kraftstoffdifferenz) berechnet. Alle 4 Werte werden auch in der Visualisierung ausgegeben bzw. in die Ergebnisdateien geschrieben. Für die Optimierung wird aber nur einer dieser Zielfunktionswerte, der Wert für  $Expo_{Druck} = 6$  berücksichtigt. Die zusätzlichen Werte dienen nur einer ausführlicheren Dokumentation.

ringen Bewertung größerer Abweichungen führt. Erst mit einem  $Expo_{Druck}$  von 6 konnten diese Probleme überwunden werden.

Diese Problematik der Wahl von  $Expo_{Druck}$  lässt sich anhand der Grafiken in Abb. 8-19 anschaulich zeigen. Bei der Bewertung der Übereinstimmung zwischen den beiden Zylinderdruckkurven in jeder der Grafiken kann eine Reihenfolge entsprechend der Qualität der Übereinstimmung aufgestellt werden. Die Zielfunktion muß in der Art gewählt werden, daß die durch die Zielfunktionswerte aufgestellte Reihenfolge mit der durch den Anwender vorgenommenen Rangfolge übereinstimmt bzw. zumindest ähnlich ist. Dabei muß beachtet werden, daß hier mit einer Minimierung gearbeitet wird (kleine Zielfunktionswerte sind besser als große Werte).

Abbildung 8-19 enthält 5 Grafiken (1. Grafik: oben links, 2. Grafik: oben rechts, 3. Grafik: Mitte links, 4. Grafik: Mitte rechts, 5. Grafik: unten links). Eine Bewertung ergibt, daß die 2. und 3. Grafik die beste Übereinstimmung zwischen den Kurven zeigen, die 1. Grafik ist am schlechtesten und die 4. und 5. Grafik sind jeweils nicht sehr gut, aber trotzdem besser als die 1. Grafik. Durch die Zielfunktionswerte sollte also die folgende Reihenfolge (Nummer der Grafik, beste zuerst) erreicht werden: 2, 3, 4/5, 1.

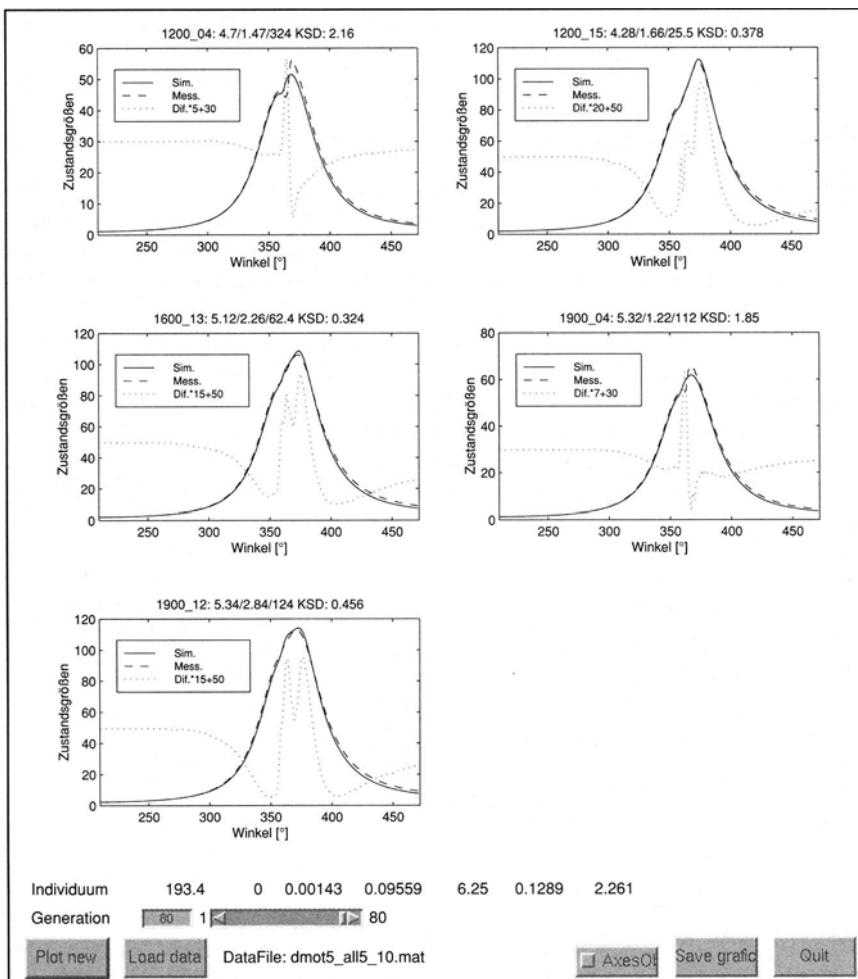
In jeder der Grafiken sind die Zielfunktionswerte für drei verschiedene Werte von  $Expo_{Druck}$  enthalten. Hier soll nur ein Vergleich zwischen den Zielfunktionswerten für  $Expo_{Druck} = 2$  (1,47 1,66 2,26 1,22 2,84) und  $Expo_{Druck} = 6$  (324 25,5 62,4 112 124) vorgenommen werden. Für  $Expo_{Druck} = 2$  ergibt sich die Reihenfolge zu: 4, 1, 2, 3, 5. Dieses Ergebnis entspricht nicht der erwarteten Reihenfolge. Bei der Verwendung von  $Expo_{Druck} = 6$  ergibt sich die Reihenfolge zu: 2, 3, 4, 5, 1. Diese Reihenfolge entspricht den vorab aufgestellten Erwartungen. Die Ursache dieser besseren Wiedergabe der erwarteten Bewertung liegt darin, daß durch einen höheren Wert für  $Expo_{Druck}$  große Differenzen, wie sie in den Grafiken 1 und 4 auftreten, stärker bewertet werden.

Die zur Wahl von  $Expo_{Druck}$  dargelegten Überlegungen enthalten die Bewertung, welche Art der Übereinstimmung zwischen zwei Kurven als gut und welche als nicht so gut betrachtet wird. Hier wird davon ausgegangen, daß kleine Abweichungen (auch wenn sie über einen längeren Bereich auftreten) nicht so schlecht sind wie stärkere Abweichungen (auch wenn sie nur kurz auftreten). Eine stärkere Abweichung deutet oft darauf hin, daß qualitativ bestimmte Bereiche noch nicht gut modelliert sind. Kleinere Abweichungen dagegen deuten auf geringfügige quantitative Abweichungen hin, die nicht so gravierend sind. Aus diesen Gründen wird für alle Optimierungen der hohe Wert für  $Expo_{Druck} = 6$  gewählt, da damit die oben aufgeführten Probleme sehr gut gelöst werden können.

## **Erste Optimierungen**

Um einen Eindruck von den Problemen bei der Optimierung zu bekommen, wurden die ersten Optimierungen mit 5 Datensätzen durchgeführt: 1200\_04, 1200\_15, 1600\_13, 1900\_04, 1900\_12. Das heißt, fünf Datensätze wurden für jede Lösung (für jedes Individuum) simuliert und einzeln entsprechend der Zielfunktion bewertet. Diese fünf Werte werden addiert und ergeben den für die Optimierung verwendeten Zielfunktionswert.

In Abb. 8-19 sind die Simulationsergebnisse für diese Datensätze unter Verwendung des erreichten Optimierungsergebnisses dargestellt. Auf den ersten Blick ist die Übereinstimmung zwischen den Meß- und Simulationskurven in allen Grafiken gut. Bei einer genaueren Betrachtung lassen sich allerdings zwei Klassen unterscheiden. Bei den Datensätzen 1200\_15 (oben rechts), 1600\_13 (Mitte links) und 1900\_12 (unten links) ist die Übereinstimmung wirklich sehr gut, was sich auch nach einem Blick auf die Skalierung der Differenzkurve bestätigt (Faktoren von 20 bis 15). Bei den beiden Datensätzen 1200\_04 (oben links) und 1900\_04 (Mitte rechts) zeigen sich deutliche Abweichungen im Bereich des maximalen Druckes. Diese Unterschiede zeigen sich auch in der Differenzkurve mit deutlich kleineren Skalierungsfaktoren von 5 und 7.



**Abb. 8-19.** Simulation der ersten 5 Datensätze, Lösung ist das Ergebnis einer Optimierung über die ersten 5 Datensätze

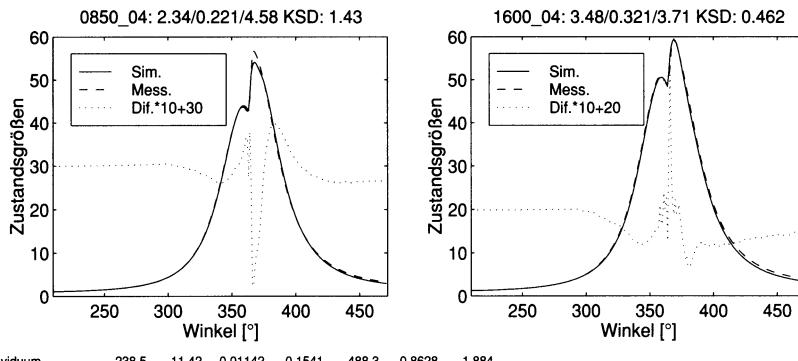
In Schlußfolgerung dieser (und weiterer hier nicht gezeigter) Optimierungen scheint sich abzuzeichnen, daß die Parametrisierung für Datensätze mit einem Druck von 4 bar Probleme bereitet. Ob dies ein prinzipielles Problem des Simulationsprogrammes oder nur im Zusammenhang mit der Verwendung anderer Datensätze auftritt, muß in weiteren Optimierungen durch die Beantwortung der folgenden Fragen bzw. Probleme geklärt werden:

- Gibt es eine sehr gute Parametrisierung des Modells für die Datensätze mit einem Druck von 4 bar?
- Gibt es eine sehr gute Parametrisierung des Modells für die Datensätze mit einem anderen Druck als 4 bar (höherer Druck)?
- Gibt es vielleicht doch eine sehr gute Parametrisierung des Modells für alle Datensätze?

Auf die Optimierungen, die bei der Beantwortung dieser Fragen helfen sollen, wird im folgenden eingegangen.

### **Optimierung mit Datensätzen gleichen Drucks**

Mit den Beobachtungen aus den ersten Optimierungen bot sich eine Unterscheidung der verwendeten Datensätze nach dem Druck an (zweiter Teil des Dateinamens). Deshalb wurden in einer Optimierung Datensätze mit einem Druck von 4 bar verwendet (1200\_04, 1900\_04, 0850\_04, 1600\_04) und in einer zweiten Optimierung Datensätze mit einem anderen Druck (1200\_15, 1600\_13, 1900\_12, 0850\_09, 1200\_09, 1600\_09). Mit Optimierungen anhand aufgeteilter Datensätze soll untersucht werden, wie gut die Ergebnisse, die mit einer solch beschränkten Klasse von Datensätzen erreicht werden, auf die jeweils anderen Datensätze übertragbar sind. Die Darstellungen in den Abbildungen werden aus Platzgründen jeweils auf die letzten 2 Datensätze (0850\_04, 1600\_04 bzw. 1200\_09, 1600\_09) beschränkt. Die Ergebnisse sind bei den anderen Datensätzen qualitativ gleich.



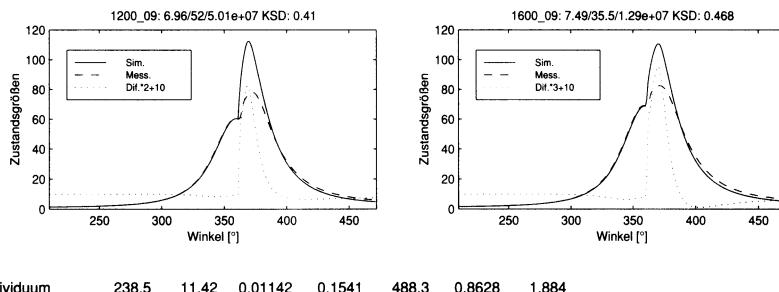
**Abb. 8-20.** Optimierung über Datensätze mit einem Druck von 4 bar; Simulation dieser Datensätze (Auswahl)

In den Abb. 8-20 und 8-21 sind jeweils Simulationsergebnisse mit einer Parametrisierung dargestellt, die aus einer Optimierung über Datensätze mit einem Druck von 4 bar stammen.

In Abb. 8-20 sind die Simulationsergebnisse für die Datensätze mit einem Druck von 4 bar zu sehen – dieselben Datensätze, die für die Optimierung verwendet wurden. Die Kurven für den Zylinderdruck von Simulation und Messung stimmen gut überein, die Unterschiede sind nur an der Differenzkurve ablesbar. Die Größenordnung der Differenz liegt für alle Kurven etwa gleich niedrig, der maximale Fehler bei etwa 2.

Mit einer Optimierung nur für die Datensätze mit einem Druck von 4 bar gelang es, für diese eine sehr gute Parametrisierung zu finden. Ein Blick auf die Parameter dieser Lösung im Vergleich mit den bisherigen Lösungen zeigt größere Unterschiede nur im 2. Parameter (Drallabhängigkeit). Bei den bisherigen Lösungen lag dieser Parameter um 0. Bei der Optimierung für einen Druck von 4 bar in Abb. 8-20 hatte dieser Parameter einen Wert von etwa 11.

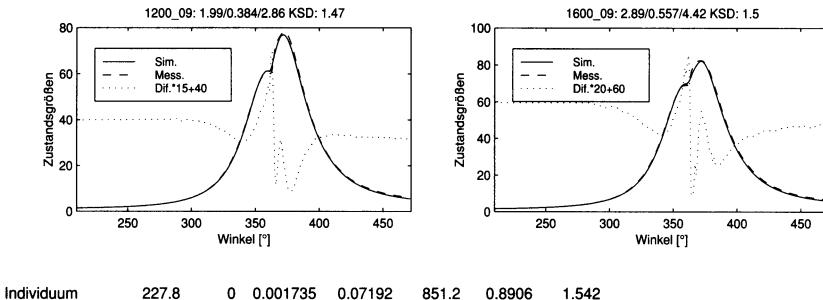
Es muß jedoch überprüft werden, wie die Simulationsergebnisse mit dieser (auf die Datensätze mit einem Druck von 4 bar zugeschnittenen) Parametrisierung für die anderen Datensätze aussehen. Diese Simulationsergebnisse sind in Abb. 8-21 dargestellt.



**Abb. 8-21.** Optimierung über Datensätze mit einem Druck von 4 bar; Simulation mit Datensätzen, die einen anderen (höheren) Druck als 4 bar haben (Auswahl)

In allen Grafiken in Abb. 8-21 ist der Unterschied zwischen Simulations- und Meßkurve deutlich zu erkennen. Die Differenzkurve hat für alle Grafiken einen ähnlichen Verlauf und die Größenordnung ist ebenfalls ähnlich. Im Bereich eines hohen Zylinderdruckes gibt es keine Übereinstimmung, die Kurven verlaufen qualitativ deutlich unterschiedlich. Damit ist diese speziell auf einen Druck von 4 bar (allgemeiner: einen niedrigen Druck) zugeschnittene Parametrisierung nicht als allgemeine Parametrisierung auf die anderen Datensätze anwendbar.

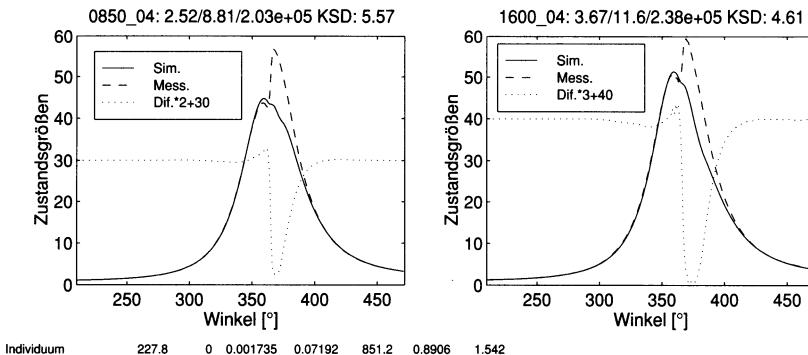
In einer weiteren Optimierung wurden nur Datensätze mit einem anderen Druck als 4 bar verwendet (1200\_15, 1600\_13, 1900\_12, 0850\_09, 1200\_09, 1600\_09), die Druckwerte liegen zwischen 9 und 15 bar. Die Simulationsergebnisse für die Parametrisierung aus dieser Optimierung sind in den Abb. 8-22 und 8-23 dargestellt.



**Abb. 8-22.** Optimierung über Datensätze mit einem anderen (höheren) Druck als 4 bar; Simulation dieser Datensätze (Auswahl)

Wie erwartet ergibt sich in Abb. 8-22 eine sehr gute Übereinstimmung zwischen den Druckkurven. In der Optimierung konnte damit eine Parametrisierung für eine sehr gute Übereinstimmung bei diesen Datensätzen ermittelt werden. Die Differenzen sind vom Betrag her sehr klein.

Es muß jedoch überprüft werden, inwieweit diese Parametrisierung, die auf Datensätze mit einem hohen Druck zugeschnitten ist, ebenfalls auf die Datensätze mit einem niedrigen Druck von 4 bar anwendbar ist. Die Simulationsergebnisse für diese Datensätze sind in Abb. 8-23 dargestellt.



**Abb. 8-23.** Optimierung über Datensätze mit einem anderen (höheren) Druck als 4 bar; Simulation mit Datensätzen, die einen Druck von 4 bar haben

Schon der erste Blick auf die Grafiken in Abb. 8-23 zeigt, daß die Parametrisierung für die Datensätze mit 4 bar Druck ungeeignet ist. In allen Grafiken sind deutliche Abweichungen zwischen Meß- und Simulationskurve erkennbar. Im Bereich eines hohen Zylinderdruckes wird die Druckspitze nicht modelliert, eine systematische Abweichung liegt vor.

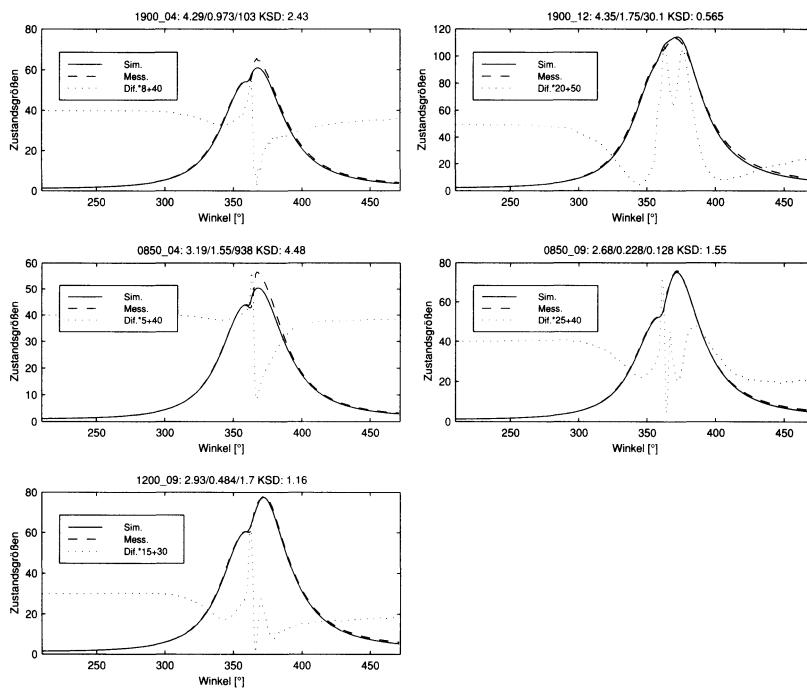
In Auswertung der Ergebnisse in diesem Abschnitt kann festgestellt werden, daß sich die zur Verfügung stehenden Datensätze in Abhängigkeit des Druckes

in zwei Klassen unterscheiden lassen, die Datensätze mit niedrigem Druck (4 bar) und die Datensätze mit einem höheren Druck (9 bis 15 bar). Die jeweils auf eine dieser Klassen zugeschnittenen Optimierungsergebnisse lassen sich nicht auf die andere Klasse übertragen.

### Optimierung entsprechend der Drehzahl

In weiteren Optimierungen wurde untersucht, ob sich eine Abhängigkeit der Optimierungsergebnisse von der für die Erstellung der Datensätze verwendeten Drehzahl (erster Teil des Dateinamens) erkennen lässt. Deshalb wurden einige Optimierungen für Datensätze gleicher Drehzahl durchgeführt. Die Ergebnisse dieser Optimierungen wurden dann an allen Datensätzen getestet und auf eine vorhandene Abhängigkeit untersucht.

Diese Untersuchungen werden hier nur an einem Beispiel dargestellt. Dabei wurden für eine Optimierung Datensätze mit einer Drehzahl von 1600 U/min verwendet: 1600\_13, 1600\_04, 1600\_09 und 1600\_15. In Abb. 8-24 werden nur die Simulationsergebnisse für einige der Datensätze gezeigt, die bei der Optimierung nicht verwendet wurden.



Individuum	210.9	0.07549	0.0004095	0.08296	695	0.5446	1.866
------------	-------	---------	-----------	---------	-----	--------	-------

**Abb. 8-24.** Optimierung über Datensätze mit einer Drehzahl von 1600 U/min; Simulation mit Datensätzen, die eine andere Drehzahl (als 1600 U/min) haben (Auswahl)

Eine sehr gute Übereinstimmung konnte nur für die Datensätze erreicht werden, die einen von 4 bar verschiedenen Druck haben. Dabei spielt die Drehzahl keine Rolle. Es zeichnet sich damit ab, daß auch bei einer Optimierung nach der Drehzahl wieder die schon weiter oben erkannte Klassifikation nach dem Druck von Bedeutung ist. Die Datensätze mit einem Druck von 4 bar zeigen deutliche Abweichungen. Bei den anderen Datensätzen dagegen ist kaum ein Unterschied zwischen Meß- und Simulationskurve zu erkennen.

### 8.3.7 Auswertung der Ergebnisse

Aus einem Vergleich der Ergebnisse der Optimierungen für die verschiedenen Kombinationen von Datensätzen und die anschließende Darstellung der Simulationsergebnisse ergeben sich die folgenden Schlußfolgerungen:

- Eine Parametrisierung des Simulationsmodells für eine große Anzahl von Datensätzen eines Motors ist möglich.
- Die Übereinstimmung zwischen Meß- und Simulationskurven ist für fast alle Datensätze sehr gut.
- Für Datensätze mit einem niedrigen Druck (4 bar) ergeben sich bei der gemeinsamen Parametrisierung aller Datensätze signifikante Abweichungen zwischen Meß- und Simulationskurve.
- Bei einer separaten Optimierung für Datensätze mit niedrigem Druck läßt sich auch für diese Datensätze eine gute Parametrisierung finden.
- Eine Abhängigkeit der Parametrisierung von der Drehzahl des Datensatzes konnte nicht festgestellt werden.

**Tabelle 8-1.** Vergleich der Werte der Parameter des Verbrennungsmodells und der Gütwerte für gut an den DaimlerChrysler 6-Zylinder-NFZ-Motor angepaßte Parametersätze; Gegenüberstellung der allgemeinen Lösung mit den nur für speziellere Bereiche geltenden Lösungen

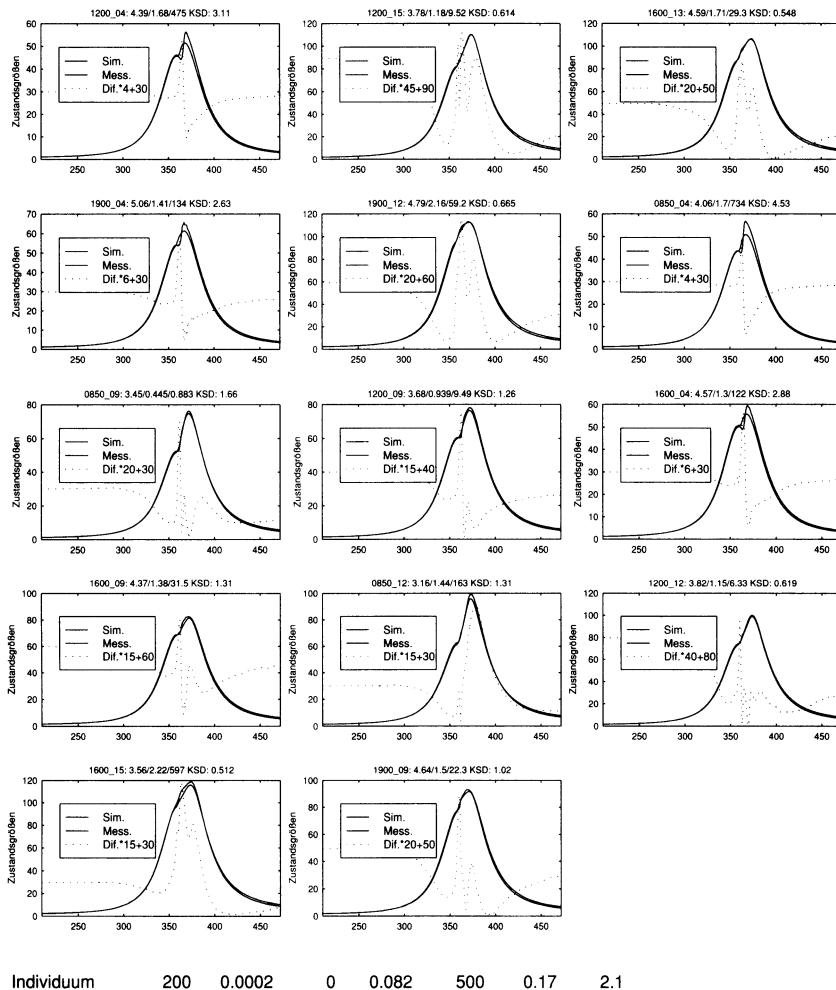
Bezeichnung der Parameter	Wert der Parameter		
	für alle Datensätze gut s. Abb. 8-25	für Druck 04 sehr gut	für Druck nicht-04 sehr gut
Gemischbildung	200	230	172
Drallabhängigkeit	0,0002	11,5	0,01
Gemischbildungsggrad	0	0,011	0,00017
Kolbengeschwindigkeit	0,82	0,15	0,068
Wandanlagerung	500	550	140
Zündverzug	0,17	0,9	0,96
Wandwärme	2,1	1,8	1,4

<b>Gütwerte:</b>			
alle Datensätze (14 Stück)	2393	513100000	953990
nur 04 Datensätze (4 Stück)	1465	254	953238
nur nicht-04 Datensätze (10)	928	513100000	752

An dieser Stelle sollen Werte für die Parameter des Modells der Verbrennung genannt werden. Diese in der 2. Spalte von Tabelle 8-1 gezeigten Werte der Parameter stellen die Lösung für das in diesem Abschnitt gestellte Optimierungsproblem dar. Am Ende der Tabelle wird der Gütwert angegeben, der sich aus der Simulation aller Datensätze mit diesem Parametersatz ergibt. Die Simulationsergebnisse für diesen Parametersatz sind in Abb. 8-25 dargestellt.

Die aufgeführten Werte der Parameter sind als gerundete Werte angegeben. Eine exakte Angabe ist auf Grund der mit einer leichten Variation der Parameter verbundenen Probleme nicht möglich und auch nicht sinnvoll, s. Abschn. 8.3.2.



**Abb. 8-25.** Simulationsergebnisse für eine Lösung des Optimierungsproblems, alle Datensätze wurden mit einem Parametersatz simuliert, s. Tabelle 8-1, 2. Spalte, der für alle Datensätze gute Ergebnisse bringt

Die gezeigten Werte stellen eine gute Lösung für das Optimierungsproblem bei einer Verwendung aller Datensätze dar. Damit ist dieser Parametersatz eine Lösung für das gesamte Optimierungsproblem.

In Beachtung der am Anfang dieses Abschnittes aufgeführten Schlußfolgerungen soll an dieser Stelle noch auf zwei spezielle Ergebnisse eingegangen werden. In den Untersuchungen wurde gezeigt, daß bei einer separaten Optimierung für Datensätze mit einem niedrigen Druck (4 bar) für diese eine sehr gute Parametrisierung gefunden werden konnte. Dasselbe gilt für eine separate Optimierung der anderen Datensätze (nicht-04).

Deshalb werden zwei weitere Lösungen für einen eingeschränkten Bereich des Optimierungsproblems gezeigt:

1. eine Parametrisierung, die sehr gut für alle Datensätze mit einem Druck von 4 bar ist (04 sehr gut), s. Tabelle 8-1 dritte Spalte,
2. eine Parametrisierung, die sehr gut für alle Datensätze mit einem anderen Druck als 4 bar ist (nicht-04 sehr gut), s. Tabelle 8-1 vierte Spalte.

Diese Lösungen gelten nur für die jeweils angegebenen Datensätze bzw. Eigenschaften der Datensätze.

Aus einem Vergleich der Gütwerte für die drei Lösungen lassen sich die folgenden Schlußfolgerungen ziehen:

1. Bei der allgemeinen Lösung des Optimierungsproblems (für alle Datensätze gut) ist der Gütwert für die 4 Datensätze mit niedrigem Druck von 4 bar größer als der Gütwert für die restlichen 10 Datensätze.
2. Die beiden speziellen Lösungen sind für ihren jeweiligen Bereich besser als die allgemeine Lösung des Optimierungsproblems.
3. Die beiden speziellen Lösungen sind als Lösung für alle Datensätze ungeeignet.

Der in der 2. Spalte von Tabelle 8-1 präsentierte Parametersatz stellt eine gute Lösung für das gestellte Optimierungsproblem dar. Mit den weiteren zwei aufgeführten Lösungen, die nur für spezielle Bereiche bzw. Datensätze gelten, konnte gezeigt werden, daß prinzipiell noch bessere Ergebnisse erreichbar sind. Dafür könnte einmal eine Beschränkung auf bestimmte Arbeits- bzw. Datenbereiche erfolgen, für die das Simulationsprogramm gut angepaßt bzw. validiert ist. Andererseits könnte das Verbrennungsmodell so verändert werden, daß es für alle in den Datensätzen vorkommenden Arbeitsbereiche gut geeignet ist.

Alle in diesem Abschnitt dargestellten Ergebnisse wurden aus den Untersuchungen mit Datensätzen eines einzigen Motors abgeleitet. Eine Aussage, ob diese Schlußfolgerungen auch auf andere Motoren übertragbar sind, kann an dieser Stelle nicht getroffen werden.

### **8.3.8 Zusammenfassung Parameteridentifikation Dieselmotor**

In diesem Abschnitt wurde die Optimierung der Parameter eines Verbrennungsmodells für direkteinspritzende Dieselmotoren durchgeführt. Die Simulation des Modells erfolgte direkt mit dem FORTRAN-Programm, das sich in bisherigen Simulationstests als erfolgreich herausgestellt hat.

In Vorarbeiten wurde das Verhalten des Systems untersucht. Diese zusätzlichen Untersuchungen, s. Abschn. 8.3.2, erbrachten neue Erkenntnisse über das Verhalten des Simulationsprogrammes sowie über die Art und das Aussehen des für die Optimierung verwendeten Gütegebirges. Dabei konnten wesentliche Eigenschaften des Optimierungsproblems ermittelt werden. Es stellte sich heraus, daß das Systems im Nahbereich (kleine Änderungen von Parametern) ein chaotisches Verhalten zeigt. Dieses chaotische Verhalten im Nahbereich war aber klein verglichen zu den Änderungen der Gütwerte bei größeren Änderungen der Parameter. Damit ließ sich erklären, warum die Optimierungsaufgabe bisher nicht mit anderen (nichtlinearen gradientenorientierten) Verfahren gelöst werden konnte.

Auf der Grundlage der Ergebnisse der Voruntersuchungen zu den Eigenschaften des Optimierungsproblems konnten geeignete Evolutionäre Algorithmen ausgewählt werden, die für die Lösung dieser Optimierungsaufgabe sehr gut geeignet sind. Der Einsatz mehrerer Suchstrategien parallel zueinander führte zu einer großen Robustheit der Suche kombiniert mit einer hohen Auflösung beim Auffinden des Optimums. Damit konnten durch die Optimierung mit Evolutionären Algorithmen Lösungen für ein Problem erarbeitet werden, die bisher so nicht möglich waren.

Bei der Optimierung des Verbrennungsmodells zeigte sich ein Vorteil des Einsatzes Evolutionärer Algorithmen deutlich. In der Definition der Zielfunktion konnten alle für den Anwender relevanten Größen und Meßwerte zur Bewertung des Verhaltens des Systems berücksichtigt werden. Dabei mußte nicht auf eine bestimmte Form der Zielfunktion oder die Definiertheit von Gradienten geachtet werden. Diese direkte Aufstellung der Zielfunktion entsprechend den Erfordernissen des Anwenders erleichtert die Lösung von Problemen deutlich.

Im Ergebnis der Optimierungen wurde für einen Dieselmotor eine Parametrisierung des Simulationsmodells ermittelt, die für fast alle zur Verfügung stehenden Meßdatensätze zu einer sehr guten Übereinstimmung zwischen Messung und Simulation führte. Abweichungen zeigten sich für eine Klasse von Meßdatensätzen, bei denen der Druck sehr niedrig ist (Druck von 4 bar). Bei einer separaten Optimierung läßt sich für diese Datensätze eine sehr gute Parametrisierung des Modells finden. Wenn das Ziel aber, wie in diesem Fall, eine gemeinsame Parametrisierung für alle Datensätze ist, läßt sich für die Datensätze mit einem niedrigen Druck keine gute Übereinstimmung erreichen. Dies deutet auf eine nicht vollständige Modellierung dieser Abhängigkeit im Simulationsprogramm hin.

In der weiteren Arbeit können die Ergebnisse auf weitere Motormodelle ausgedehnt werden. Mit den dabei gewonnenen Erkenntnissen ist einerseits eine Verifikation des Simulationsprogrammes möglich. Andererseits kann darauf aufbauend eine Erweiterung des Simulationsprogrammes vorgenommen oder der Einsatz erweiterter Simulationsprogramme in Erwägung gezogen werden.

## 8.4 Optimierung der Parameter eines Reglers (Fahrzeuglenkung)

Wenn das Modell eines Systems hinreichend bekannt ist, steht oft die Aufgabe der Festlegung und Dimensionierung eines Reglers. Damit soll ein gewünschtes Verhalten des Gesamtsystems erreicht werden. In diesem Abschnitt wird als ein Beispiel die Querlenkung eines autonomen Straßenfahrzeugs (*lateral control of an autonomous road vehicle*) vorgestellt.

Das Modell zur Querlenkung basiert auf einem linearisierten Modell des Fahrzeugverhaltens auf die Lenkung. Der verwendete Regler ist als Übertragungsfunktion implementiert. Durch den Regler müssen mehrere sich widersprechende Anforderungen erfüllt werden (Sicherheit, Komfort der Fahrzeuginsassen, Schnelligkeit). Außerdem muß das Verhalten des ermittelten Reglers unter unterschiedlichen Bedingungen (Szenarien) getestet werden.

Die Optimierung erfolgt auf der Grundlage einer Zielfunktion, die den Verlauf der Zustands- und Ausgangsgrößen des Gesamtsystems in Betracht zieht. Damit können sehr spezifische und komplexe Bewertungen des Verhaltens unter verschiedenen Szenarien durchgeführt werden. Diese Zielfunktion könnte auch in weiteren Untersuchungen unter Verwendung erweiterter und komplexerer Modelle Verwendung finden, da keine besonderen Anforderungen an bestimmte Eigenschaften des Systems (z.B. Linearität) gestellt werden.

In früheren Arbeiten werden unter anderem Neuronale Netze als Regler eingesetzt, die durch modellbasiertes Training das gewünschte Verhalten erlernen (*neural-network control with model-based training* [MHF92]). Eine weitere Arbeit befaßt sich mit der Anwendung von Neuro-Fuzzy-Reglern auf dieses System [HH94]. Die Eingangssignale für den Regler (z.B. relative Position des Fahrzeugs, Winkel des Fahrzeugs zur Straße – *yaw angle*) wurden in diesen Arbeiten aus den Aufnahmen von Videokameras entnommen [MHF92].

### 8.4.1 Modell der Querlenkung

Das verwendete Querlenkungsmodell basiert auf einem linearisierten Modell der Reaktion des Fahrzeugs auf die Lenkung auf einer geraden Straße ([MHF92]). Das Modell hat 4 Zustandsgrößen, einen Steuereingang und wird durch eine Reihe von variablen und festen Systemvariablen bestimmt.

Die 4 Zustandsvariablen des Modells sind:

- $\beta$  Schwimmwinkel (*sideslip angle*),
- $\gamma$  Abstand zur Straßenmitte (Abweichung von gewünschter Position),
- $\psi$  Differenz zwischen dem Drehwinkel (*yaw angle*) des Fahrzeugs und dem Kurswinkel (*course angle*) der Straße,
- $\lambda$  Lenkungswinkel (*steer angle*).

Das Modell kann durch einen Steuereingang geregelt werden:

- $u$  Änderung des Lenkungswinkel (*control action*).

Die folgenden Systemvariablen beeinflussen das Modell (in eckigen Klammern sind die hier verwendeten festen Werte angegeben):

- $v$  Geschwindigkeit in m/s (*velocity, speed*), [veränderlich],
- $m$  Fahrzeuggmasse in kg (*vehicle mass*), [fest: 5800 kg],
- $a$  Spurabstand in m (*wheel base*), [fest: 4,25 m],
- $k$  Reibungskoeffizient in Querrichtung in kN/rad (*lateral friction coefficient*), [150 kN/rad].

Die 4 Zustandsgleichungen des Modells sind in Gl. 8-6 gezeigt. Neben der ausführlichen Darstellung wird eine verkürzte Form mit zusammengefaßten Variablen aufgeführt, mit der in den folgenden Umstellungen weiter gearbeitet wird.

$$\begin{aligned}\dot{\beta} &= \left(-2\frac{k}{mv}\right)\beta + \left(\frac{v}{a} - \frac{k}{mv}\right)\lambda = -a_1\beta + \left(a_2 - \frac{a_1}{2}\right)\lambda \\ \dot{y} &= v(\psi + \beta) = a_3(\psi + \beta) \\ \dot{\psi} &= \left(\frac{v}{a}\right)\lambda = a_2\lambda \\ \dot{\lambda} &= u = \\ \text{mit: } a_1 &= 2\frac{k}{mv}, \quad a_2 = \frac{v}{a}, \quad a_3 = v\end{aligned}\tag{8-6}$$

Aus den Zustandsgleichungen in Gl. 8-6 läßt sich das Zustandsmodell ableiten:

$$\begin{aligned}\dot{\underline{x}} &= \underline{A}\underline{x} + \underline{b}u = \begin{pmatrix} -a_1 & 0 & 0 & a_2 - \frac{a_1}{2} \\ a_3 & 0 & a_3 & 0 \\ 0 & 0 & 0 & a_2 \\ 0 & 0 & 0 & 0 \end{pmatrix} \underline{x} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} u \\ \tilde{y} &= \underline{c}'\underline{x} = (0 \ 1 \ 0 \ 0)\underline{x}\end{aligned}\tag{8-7}$$

Die Übertragungsfunktion des Modells kann durch Einsetzen der entsprechenden Größen (s. Gl. 8-7) in die allgemeine Form und umstellen ermittelt werden:

$$\begin{aligned}\tilde{y} &= \underline{c}'[\underline{E}s - \underline{A}]^{-1}\underline{b}u \\ \Rightarrow G_p(s) &= a_3\left(2a_2 - \frac{a_1}{2}\right) \frac{s + \frac{2a_1a_2}{4a_2-a_1}}{s^4 + a_1s^3}\end{aligned}\tag{8-8}$$

Damit sind alle Darstellungsformen gegeben, die für die weitere Arbeit mit dem Modell benötigt werden.

Bei der Ermittlung der Polstellen für das Nennerpolynom von Gl. 8-8 ergeben sich 3 Polstellen im Ursprung und 1 Polstelle in der linken s-Halbebene:

roots(...)= [ 0, 0, 0, -0,0026], für eine Geschwindigkeit  $v = 20$  m/s.

Das System ist damit nicht bibostabil (*bounded in – bounded out*).

## 8.4.2 Anforderungen an den Regler

Die grundlegende Aufgabe des zu entwerfenden Reglers ist:

- das Fahrzeug in der Mitte der Straße (bzw. Fahrbahnspur) zu halten.

Daneben sind eine Anzahl von weiteren Anforderungen zu erfüllen:

- große Abweichungen von der Fahrbahnmitte ( $y$ ) sind so schnell wie möglich zu korrigieren (Sicherheitsanforderung),
- kleine Abweichungen von der Fahrbahnmitte können toleriert werden, die Grenze zwischen großen und kleinen Abweichungen wird im weiteren mit  $y_c = 0,5$  m festgelegt,
- Lenkbewegungen ( $u$ ) sollen so klein wie möglich gehalten werden (Komfortanforderungen), es darf nicht zu plötzlichen großen Lenkbewegungen kommen (*no lateral jolt*), außerdem sollte kein Hin- und Herlenken auftreten,
- die Querbeschleunigung soll in einem großen Geschwindigkeitsbereich konstant gehalten werden.

Die beiden Anforderungen, große Abweichungen sollen schnell korrigiert werden und die Lenkbewegungen sind klein zu halten, widersprechen sich. Dieser Konflikt sowie die Nichtlinearität der letzten Forderung (Querbeschleunigung konstant) sind die Hauptschwierigkeiten dieses Regelungsproblems.

Alle oben aufgestellten Forderungen müssen in der Zielfunktion berücksichtigt und untereinander in ihrer Wertigkeit gewichtet werden. Mehr dazu in Unterabschn. 8.4.4.

Für die Festlegung der Struktur des zu verwendenden Reglers muß zuerst das zu regelnde System untersucht werden. Das Modell der Querlenkung stellt ein System 4. Ordnung dar, wie aus der Übertragungsfunktion in Gl. 8-8 zu ersehen ist. Um mit einem analytischen Verfahren eine eindeutige Lösung zu erhalten, muß ein Regler 3. Ordnung eingesetzt werden.

In den durchgeföhrten Experimenten wird zuerst ein Regler verwendet, der als Übertragungsfunktion entsprechend Gl. 8-9 implementiert werden kann:

$$G_R(s) = \frac{r_1 s^2 + r_2 s + r_3}{s^3 + r_4 s^2 + r_5 s + r_6} \quad (8-9)$$

Ein Regler mit dieser Struktur hat 6 freie Parameter,  $r_1$  bis  $r_6$ . Diese Parameter sind durch den Optimierungsalgorithmus zu bestimmen.

Wenn mit diesem Regler keine zufriedenstellenden Ergebnisse zu erreichen sind, kann die Struktur zu einem *pole-placement*-Regler 3. Ordnung erweitert werden:

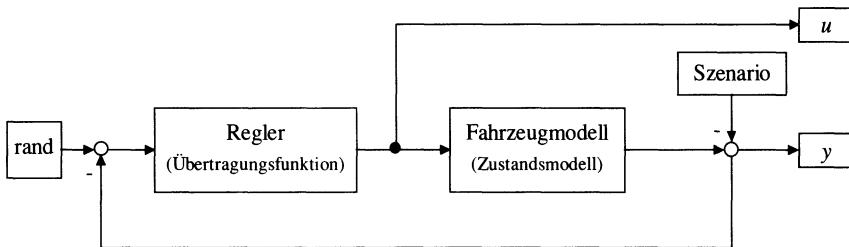
$$G_R(s) = \frac{r_0 s^3 + r_1 s^2 + r_2 s + r_3}{s^3 + r_4 s^2 + r_5 s + r_6} \quad (8-10)$$

Dieser Regler hat 7 freie Parameter,  $r_0$  bis  $r_6$ , die durch den Optimierungsalgorithmus zu bestimmen sind.

### 8.4.3 Simulation des Gesamtsystems

Aus dem vorliegenden Modell und dem Regler kann das Gesamtsystem zusammengestellt werden. Dieses Gesamtsystem stellt die Grundlage für die nachfolgenden Simulationen dar.

Für die Simulation ist zu beachten, daß die Anfangszustände des Modells gesetzt (in welchem Zustand befindet sich das Fahrzeug zu Beginn der Simulation) und die Daten der Szenarien (Verlauf der Fahrbahn) eingebunden werden müssen. Außerdem werden die Meßwerte der Abweichung von der Fahrbahnmitte ( $y$ ) durch ein zusätzliches Rauschen in gewissem Maße verfälscht (höhere Realitätsnähe der Simulation).



**Abb. 8-26.** Struktur des Gesamtsystems aus Fahrzeugquerlenkung und Regler unter Einbeziehung des Fahrbahnverlaufs

Diese Struktur kann im verwendeten Simulationswerkzeug (hier SIMULINK) definiert werden. Für jede Simulation innerhalb der Optimierung müssen dann die Parameter des Reglers, die Daten des Szenario und die Anfangszustände des Fahrzeugmodells in die entsprechenden Blöcke „geladen“ werden und die Simulation kann durchgeführt werden. Anhand der erhaltenen Ausgangswerte (Steuerung  $u$  und Ausgang Fahrbahnabweichung  $y$ ) kann die Güte der verwendeten Reglerparameter für dieses Szenario bewertet werden (Güte- bzw. Zielfunktion).

Da es sich bei dem vorliegenden System um ein lineares System handelt, kann mit einfachen Mittel ein Stabilitätstest durchgeführt werden. Dies wird vor jeder Simulation getan. Nur stabile Systeme werden überhaupt simuliert. Instabile Systeme bekommen sehr hohe (schlechte) Zielfunktionswerte zugewiesen. Nähere Hinweise dazu in Unterabschn. 8.4.4.

Das Gesamtsystem wurde bisher als kontinuierliches System betrachtet. Wenn das System im Einsatz allerdings abgetastet wird, dann sollte dies in der Simulation berücksichtigt werden. Dazu wird die Simulation mit dem zeitdiskreten Systemmodell und der entsprechenden Abtastzeit (*sample time*) durchgeführt.

Da sich das prinzipielle Vorgehen für beide Varianten, kontinuierliches oder diskretes System, nicht unterscheidet, wird im weiteren das kontinuierliche System verwendet.

#### 8.4.4 Aufstellung der Zielfunktion

Die Aufgaben des Reglers wurden bereits in Unterabschn. 8.4.2 vorgestellt. Inwieweit diese Zielstellungen durch einen einzelnen Regler erreicht werden, muß durch die Zielfunktion bewertet werden.

Jede einzelne Zielstellung stellt ein zu bewertendes Kriterium dar. Da hier mehrere Kriterien gleichzeitig bewertet werden, handelt es sich um eine mehrkriterielle Zielfunktion. Für die Behandlung dieser Kriterien könnten alle Verfahren eingesetzt werden, die in Unterabschn. 3.1.3 vorgestellt wurden.

In diesem Beispiel ist in etwa bekannt, wie die Kriterien miteinander verknüpft werden können. Deshalb werden alle Kriterien durch Gewichtung zu einem einzigen Gesamtkriterium zusammengefaßt (gewichtete Summe - *weighted sum*). Nur dieses Gesamtkriterium wird durch den Optimierungsalgorithmus als Bewertungsgrundlage verwendet.

Die folgenden Gleichungen zeigen zuerst die Umsetzung der einzelnen Zielstellungen. Danach erfolgt eine Kombination der Teilkriterien zur Gesamtbewertung durch eine gewichtete Summe.

Abweichung von der Fahrbahnmitte:

- Abweichungen größer als die kritische Abweichung ( $y_c = 0,5$  m) sind schnell zu verlassen (sehr hohe/schlechte Bewertung), geringere können toleriert werden. Diese beiden Zielstellungen lassen sich gut in einem Kriterium vereinen.

$$J_y = \int_{t_0}^{t_{end}} \left( \frac{y_t}{y_c} \right)^6 dt \quad (8-11)$$

- Durch die Division der aktuellen Abweichung ( $y_t$ ) durch die kritische Abweichung ( $y_c$ ) und die anschließende hohe Potenzierung erfolgt ein fließender Übergang zwischen hoher Bewertung großer Abweichungen und der geringen Bewertung kleiner Abweichungen. Durch die Größe der Potenz (hier 6) kann entschieden werden, wie stark die großen Abweichungen bewertet werden im Vergleich zu den unkritischen Abweichungen. Je größer die Potenz ist, um so größer ist der Unterschied zwischen Werten oberhalb und unterhalb der kritischen Grenze.
- Der Wert der kritischen Abweichung kann als eine Zielstellung angesehen werden, die unbedingt erreicht werden muß. Werte größer als dieses Ziel (*goal*) müssen sehr schlecht bewertet werden, kleinere Werte dagegen fallen kaum ins Gewicht. Es findet aber trotz allem noch immer eine Unterscheidung zwischen Werten statt, die unterhalb der kritischen Grenze liegen. Damit wird dem Optimierungsalgorithmus auch in diesem Bereich ein Anhaltspunkt für eine weitere Verbesserung geliefert.
- Es sollte beachtet werden, daß für die Potenz eine gerade Zahl verwendet wird. Nur dann ist sichergestellt, daß das Teilkriterium immer eine nicht-negative Zahl ergibt. Bei der Verwendung anderer Potenzen muß vorher der Absolutwert der entsprechenden Werte berechnet werden.

Größe der Querbeschleunigung:

- Die Querbeschleunigung soll möglichst klein gehalten werden. Hohe Werte der Querbeschleunigung sind für die Fahrzeuginsassen unangenehm und verringern den Komfort deutlich.

$$J_\lambda = \int_{t_0}^{t_{end}} (\dot{\lambda}_t v)^2 dt \quad (8-12)$$

- Hier wird der Wert der Querbeschleunigung einfach quadriert. Dies ist der klassische Weg und man erhält immer nichtnegative Werte für dieses Teilkriterium.
- Wenn die Querbeschleunigung möglichst klein gehalten wird, kann damit gleichzeitig die Forderung nach einer weitgehenden Konstanz der Querbeschleunigung erreicht werden.

Kein Kreuzen der Fahrbahnmitte:

- Eine weitere Zielstellung, die durch den Regler erreicht werden sollte, zeigte sich erst im Verlauf einiger Experimente. Wenn eine Fahrbahnabweichung ausgeregelt wird, sollte es zu keinem Überkreuzen der Fahrbahnmitte kommen (*overshot over lane center*). Dieses Pendeln um die Fahrbahnmitte ist für die Fahrzeuginsassen nicht angenehm und außerdem mit zusätzlichem Aufwand verbunden. Die bereits vorgenommene Bewertung der Größe der Querbeschleunigung erwies sich hierfür als nicht ausreichend.

$$J_o = \int_{t_0}^{t_{end}} y_t \left( \text{sign}(y_t) \neq \text{sign}(y_{t_0}) \right)^2 dt \quad (8-13)$$

- Zu diesem Zweck werden die Bereiche ermittelt, in denen es zu einem Überkreuzen der Fahrbahnmitte gekommen ist (Wechsel des Vorzeichens von  $y$ ). In den entsprechenden Bereichen werden die Werte von  $y$  quadriert und das Integral gebildet. Der erhaltene Wert dient als ein weiteres Kriterium.

Ein Zusammenfügen der Kriterien zu einem einzigen Kriterium erfolgt durch eine gewichtete Summe. Dazu wird jedes der Kriterien mit einem festzulegenden Gewichtungsfaktor multipliziert und alle Werte miteinander addiert.

$$\text{ObjVal} = W_y \cdot J_y + W_\lambda \cdot J_\lambda + W_o \cdot J_o \quad (8-14)$$

Die Gewichtungsfaktoren für die einzelnen Kriterien,  $W_y$  (Abweichung von Fahrbahnmitte),  $W_\lambda$  (Querbeschleunigung) und  $W_o$  (Überkreuzen der Fahrbahnmitte) wurden wie folgt festgelegt:

$$W_y = 1 \quad W_\lambda = 625 \quad W_o = 5000 \quad (8-15)$$

Der erhaltene Wert dient als einziges Kriterium für die Bewertung eines Reglers durch den Evolutionären Algorithmus.

Es ist möglich, ein weiteres Kriterium zu definieren, daß nur die Abweichungen von der Fahrbahnmitte nach einer bestimmten Zeit bewertet. Damit hätte der Regler eine gewisse Zeit, um die Abweichung auszuregeln. Erst wenn es bis dahin nicht geschafft wurde, findet eine weitere Bewertung statt. Bei einer Simulation über 10 Sekunden könnte diese Bewertung nach 4 Sekunden beginnen. Alle zu diesem Zeitpunkt noch vorliegenden Abweichungen werden dann wesentlich

stärker bestraft, als dies mit Gl. 8-11 geschieht. Dies führt zu einer strikten Minimierung der Abweichungen von der Fahrbahnmitte zum Ende der Simulation hin. Hier wird dieses Kriterium meistens nicht verwendet.

Wie in Unterabschn. 8.4.3 bereits angedeutet, wird vor jeder Simulation des Gesamtsystems ein Stabilitätstest durchgeführt. Ist das System stabil, wird die Simulation durchgeführt und die oben vorgestellte Zielfunktion kommt zum Einsatz. Stellt sich dagegen heraus, daß das Gesamtsystem aus Modell und Regler instabil ist (was insbesondere zu Beginn einer Optimierung sehr oft vorkommt), wird keine Simulation durchgeführt. Statt dessen kommt eine modifizierte Zielfunktion zum Einsatz.

Zielfunktionswert bei instabilem System:

- Der Zielfunktionswert für instabile Systeme muß höher sein als der für jedes stabile System.
- Zusätzlich sollte ein Unterschied zwischen „sehr“ instabilen Systemen und „nicht so“ instabilen Systemen enthalten sein. Damit erhält der Optimierungsalgorithmus zumindest einen kleinen Hinweis, welches von zwei instabilen Systemen besser ist.
- Kontinuierliches System:

$$ObjVal(\text{instable}) = 10^{10} \cdot \max(\text{real}(\text{roots}(\text{system}))) + 10^7 \quad (8-16)$$

- Der Summand  $10^7$  sorgt für sehr große Zielfunktionswerte für alle instabilen Systeme.
- Zur Unterscheidung der „Instabilität“ der Gesamtsysteme wird die Polstelle des jeweiligen Nennerpolynoms verwendet und dieser Wert mit  $10^{10}$  multipliziert. Damit wird eine Unterscheidung der Güte zwischen zwei instabilen Systemen und damit eine Reihenfolgebildung ermöglicht.

Wenn bei einer Optimierung am Anfang nur instabile Systeme vorliegen, würde es ohne eine Unterscheidung zwischen den instabilen Systemen zu einer zufälligen Suche (*random search*) kommen. Mit der hier vorgenommenen Bewertung der größten vorliegenden Polstelle wird zumindest eine Richtung der weiteren Suche geboten. Ob dies wirklich funktioniert, kann nur von Fall zu Fall entschieden werden.

#### 8.4.5 Durchführung der Optimierung

Die Optimierung der Reglerparameter der Fahrzeuglenkung stellt ein Parameteroptimierungsproblem dar. Da keine spezielle Information über das Aussehen der Zielfunktion vorliegt, wird mit einer global orientierten Suche gearbeitet.

Die folgenden Operatoren und Optionen werden verwendet:

- 120 Individuen (4 Unterpopulationen mit je 40 Individuen) über 100 Generationen,
- lineares Ranking mit Selektionsdruck von 1,7, *stochastic universal sampling, generation gap* von 0,9,
- Rekombination: diskrete Rekombination und Linien-Rekombination mit Rekombinationsrate von 1,

- Mutation: reelle Mutation, Mutationsbereich von [0,1, 0,03, 0,01, 0,003] und Mutationspräzision von 12,
- Migration zwischen den Unterpopulationen alle 20 Generationen, Austausch von 10% der besten Individuen der Unterpopulation,
- Anwendung verschiedener Strategien und Konkurrenz zwischen Unterpopulationen.

Die Hintergründe zur Auswahl der Operatoren und ihrer Optionen werden ausführlich in Abschn. 7.2 erläutert.

Alle Daten der Optimierung einschließlich der Ergebnisse werden in einer Textdatei und einer binären Datei aufgezeichnet, um später komfortabel die Auswertung vornehmen zu können.

Während der Optimierung wird alle 5 Generationen eine problemspezifische Ergebnisvisualisierung des besten Individuums vorgenommen. Damit kann während einer laufenden Optimierung geprüft werden, ob die berechneten Regler die erhofften Ergebnisse erreichen. Dieselbe Visualisierung wird in der Auswertung in Unterabschn. 8.4.6 zur visuellen Veranschaulichung der Qualität des jeweiligen Reglers verwendet.

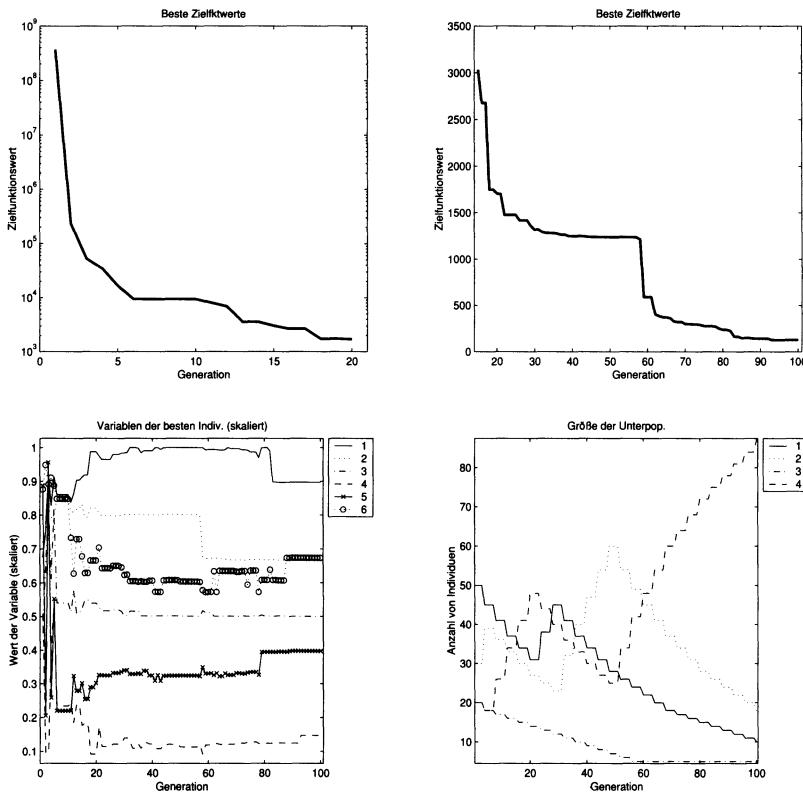
Neben der problemspezifischen Visualisierung kam auch die Visualisierung des Zustandes und Verlaufs der Optimierung zur Anwendung, die in Kap. 5 ausführlich vorgestellt wurde.

In Abb. 8-27 ist ein Beispiel für die Visualisierung des Verlaufs einer Optimierung der Fahrzeuglenkung dargestellt. Es werden der Verlauf des besten Zielfunktionswertes jeder Generation, die Variablen des besten Individuums jeder Generation und die Position der Unterpopulationen gezeigt. Dieser Lauf zur Optimierung des Reglers in Gl. 8-9 (6 Parameter) bei einer Geschwindigkeit des Fahrzeugs von  $v = 20 \text{ m/s}$  korrespondiert mit den dargestellten Ergebnissen in Abb. 8-28.

Aus dem linken oberen Diagramm in Abb. 8-27 ist zu erkennen, daß in den ersten Generationen nur Regler mit sehr großen Zielfunktionswerten gefunden wurden. Allerdings wurde in diesem Lauf schon in Generation 1 mindestens ein stabiler Regler gefunden (in anderen Läufen waren dazu manchmal 3-4 Generationen notwendig). Im weiteren Verlauf der Optimierung verbessern sich die Zielfunktionswerte kontinuierlich (Diagramm oben rechts), wobei es um Generation 18 und 58 zu größeren Sprüngen kommt.

Die Veränderung der Reglerparameter im Verlauf der Optimierung kann aus dem Diagramm unten links in Abb. 8-27 abgelesen werden. Dabei ist zu beachten, daß die Darstellung die skalierten Variablenwerte zeigt (relativer Wert innerhalb des Suchbereiches der Variablen). Nur dadurch sind die Veränderungen der in unterschiedlichen Größenordnungen vorliegenden Variablen zu erkennen.

Durch Auswertung der Größe der Unterpopulationen, Abb. 8-27 unten links, kann der Erfolg der einzelnen Strategien bewertet werden. Zu Beginn der Optimierung dominieren Unterpopulation 1 und 4 (diskrete Rekombination mit großen bzw. sehr kleinen Mutationsschritten). Ab Generation 30 ist Unterpopulation 2 (diskrete Rekombination mit mittleren Mutationsschritten) am erfolgreichsten und dominiert bis Generation 50. Zu diesem Zeitpunkt wird Unterpopulation 4 besser und bleibt dies bis zum Ende des Laufs. Unterpopulation 3 (Linienrekombination) ist im gesamten Lauf nicht erfolgreich.



**Abb. 8-27.** Verlauf der Optimierung für die Fahrzeuglenkung, oben links: bester Zielfunktionswert der Generationen 1–20, oben rechts: wie links, Generationen 15–100, unten links: Variablen des besten Individuums über alle Generationen, unten rechts: Größe der Unterpopulationen

Aus diesem Vergleich der Strategien kann abgeleitet werden, daß die Linien-Rekombination für dieses Problem nicht so gut geeignet ist. Die diskrete Rekombination mit unterschiedlichen Mutationsschritten war deutlich erfolgreicher.

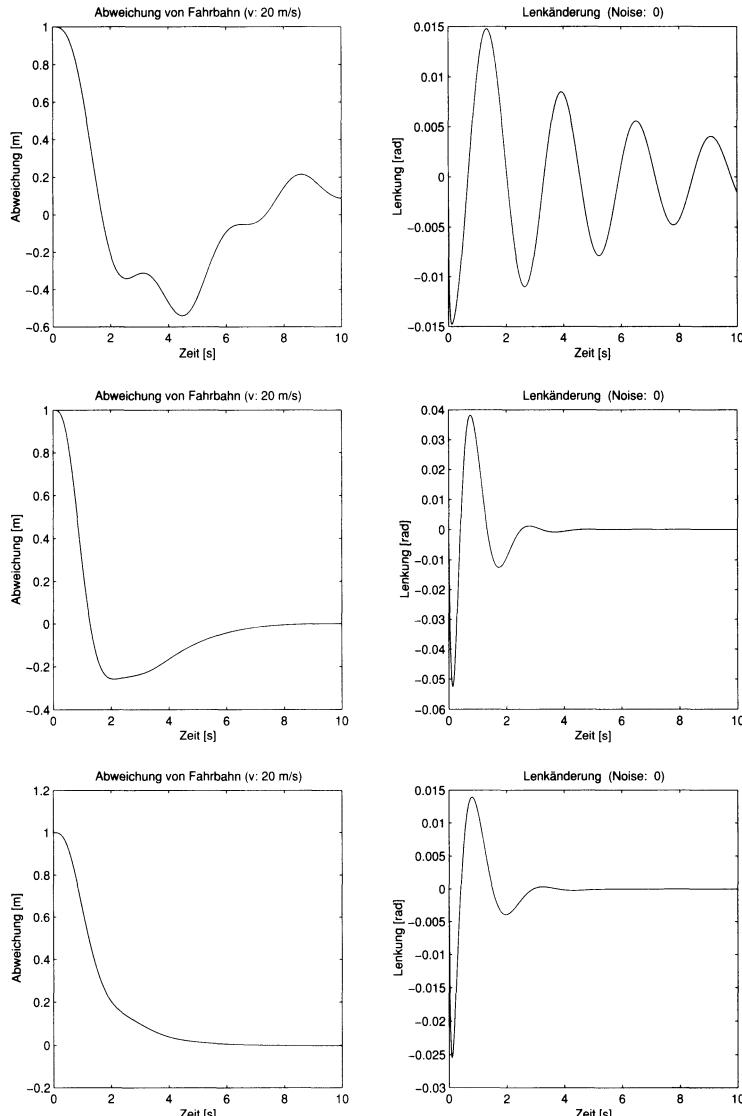
Für die anderen durchgeföhrten Optimierungen zur Fahrzeuglenkung sehen die Visualisierungsergebnisse ähnlich aus. Deshalb wird auf deren Darstellung verzichtet.

#### 8.4.6 Ergebnisse der Optimierung

In diesem Unterabschnitt werden einige der ermittelten Ergebnisse der Regleroptimierung für die Fahrzeuglenkung dokumentiert. Dabei wird sich jeweils auf die Darstellung des Reglers und des damit ermittelten Systemverhaltens konzentriert.

Die erste Optimierung wurde mit einer Reglerstruktur gemäß Gl. 8-9 und der Zielfunktion aus Unterabschn. 8.4.4 für eine Geschwindigkeit von  $v = 20 \text{ m/s}$  durchgeführt. In allen folgenden Optimierungen kam immer eine Anfangsabweichung

chung von der Fahrbahnmitte von 1 m zur Anwendung. Die Simulationen wurden über einen Zeitraum von  $t_{end} = 10$  s durchgeführt. Im Gegensatz zu den Optimierungen wurde bei den Simulationen der Ergebnisse kein zusätzliches Rauschen (Noise: 0) verwendet.



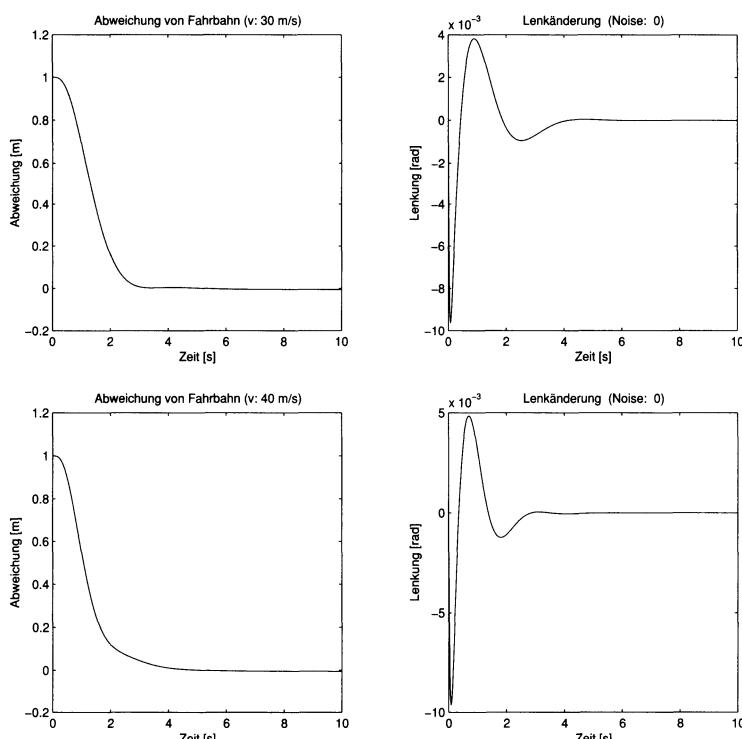
**Abb. 8-28.** Problemspezifische Visualisierung der Autolenkung: Verlauf der Abweichung von der Fahrbahnmitte (links) und zugehörige Änderung des Lenkwinkels (rechts) im Verlauf einer Optimierung ( $v = 20$  m/s); oben: Generation 10 (ZFW: 9400), Mitte: Generation 50 (ZFW: 1207), unten: Generation 100 (ZFW: 98); alle Simulationen ohne zusätzliches Rauschen

In Abb. 8-28 wird gezeigt, wie sich die Qualität der optimierten Regler im Verlauf einer Optimierung bei einer Geschwindigkeit von  $v = 20 \text{ m/s}$  verbessert. Dazu wird die problemspezifische Visualisierung verwendet, welche den Verlauf der Abweichung von der Fahrbahnmitte (links) und die zugrundeliegende Änderung des Lenkwinkels (rechts) darstellt. Die oberen Diagramme zeigen den Beginn der Optimierung (Generation 10), die mittleren Diagramme den Stand in Generation 50 und in den unteren Diagrammen ist das Ergebnis des Laufes (Generation 100) zu sehen. Der Zielfunktionswert (ZFW), der entsprechend Unterabschn. 8.4.4 berechnet wurde, verbesserte sich von 9400 in Generation 10 über 1207 in Generation 50 auf nur noch 98 zum Ende der Optimierung.

Die visuelle Bewertung des Ergebnisses der Optimierung ist gut. Die Lenkänderungen sind nicht sehr hoch, es findet kein Überschwingen statt und nach etwa 4 s ist die Abweichung von der Fahrbahnmitte ausgeregelt. Die Parameter des Reglers sind in Gl. 8-17 angegeben.

Regler 1, Geschwindigkeit  $v = 20 \text{ m/s}$ :

$$G_{R_1}(s) = \frac{0,69 s^2 + 0,0017 s + 0,000023}{s^3 + 21,0 s^2 + 73,5 s + 199} \quad (8-17)$$



**Abb. 8-29.** Problemspezifische Visualisierung der Autolenkung: Verlauf der Abweichung von der Fahrbahnmitte (links) und zugehörige Änderung des Lenkwinkels (rechts); oben:  $v = 30 \text{ m/s}$  (ZFW: 64), unten:  $v = 40 \text{ m/s}$  (ZFW: 69)

Um den Einfluß der variablen Größe Geschwindigkeit zu ermitteln, wurden zusätzlich die Reglerparameter für Geschwindigkeiten von  $v = [30 \text{ m/s}, 40 \text{ m/s}]$  optimiert.

Regler 2, Geschwindigkeit  $v = 30 \text{ m/s}$ :

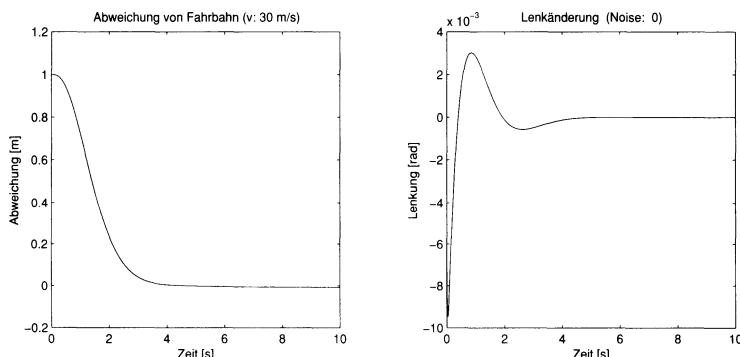
$$G_{R_2}(s) = \frac{0,41s^2 + 0,0010s + 0,00013}{s^3 + 35,8s^2 + 127s + 243} \quad (8-18)$$

Regler 3, Geschwindigkeit  $v = 40 \text{ m/s}$ :

$$G_{R_3}(s) = \frac{0,31s^2 + 0,0010s + 0,000082}{s^3 + 25,6s^2 + 107s + 298} \quad (8-19)$$

Der Verlauf der Abweichung von der Fahrbahnmitte und die entsprechende Änderung des Lenkwinkels sind für die obigen Regler in Abb. 8-29 dargestellt.

Die Ergebnisse für die höheren Geschwindigkeiten sehen dem der Geschwindigkeit von 20 m/s sehr ähnlich und sind genauso zufriedenstellend. Mit einigen weitergehenden Überlegungen und entsprechend angepaßten Optimierungen sollte es keine große Schwierigkeit sein, einen Regler zu entwerfen und zu optimieren, der für einen größeren Geschwindigkeitsbereich einsetzbar ist. Dazu müssen einige der Parameter in Abhängigkeit der Geschwindigkeit berechnet werden.



**Abb. 8-30.** Ergebnis der Optimierung der Autolenkung für die erweiterte Reglerstruktur: Verlauf der Abweichung von der Fahrbahnmitte (links) und zugehörige Änderung des Lenkwinkels (rechts),  $v = 30 \text{ m/s}$  (ZFW: 74)

Die gezeigten Ergebnisse waren zufriedenstellend. Um zu untersuchen, ob eine weitere Verbesserung möglich ist, wurde die Reglerstruktur gemäß Gl. 8-10 mit 7 freien Parametern verwendet. Bei den Optimierungen dieser erweiterten Reglerstruktur für die verschiedenen Geschwindigkeiten wurde keine deutliche Verbesserung der Qualität der Lösungen erreicht. Die Ergebnisse liegen in derselben Größenordnung wie in den obigen Abbildungen. Deshalb wird an dieser Stelle nur das Ergebnis für die Geschwindigkeit von  $v = 30 \text{ m/s}$  gezeigt.

Regler 4, Geschwindigkeit  $v = 30 \text{ m/s}$ :

$$G_{R_4}(s) = \frac{0,0063s^3 + 0,59s^2 + 0,0023s + 0,00019}{s^3 + 54,5s^2 + 214s + 379} \quad (8-20)$$

Im optischen Vergleich ist kein Unterschied zwischen den Diagrammen in Abb. 8-30 und Abb. 8-29 oben zu erkennen, die Zielfunktionswerte unterscheiden sich kaum. Deshalb kann im weiteren mit der einfachen Reglerstruktur aus Gl. 8-9 gearbeitet werden, da mit der erweiterten Reglerstruktur keine weitere Verbesserung zu erreichen ist.

#### 8.4.7 Zusammenfassung Fahrzeuglenkung

In diesem Abschnitt wurde die Optimierung der Parameter eines Reglers am Beispiel der Fahrzeugquerlenkung gezeigt. Ausgehend vom Modell der Fahrzeuglenkung (Unterabschn. 8.4.1) und den Anforderungen an den Regler (Unterabschn. 8.4.2) wurde eine Zielfunktion (Unterabschn. 8.4.4) aufgestellt. Diese bewertet alle notwendigen Aspekte der Zustandsgrößen. Auf den ersten Blick scheint die Zielfunktion zwar sehr komplex zu sein, aber nur durch die Einbeziehung aller wichtigen Größen kann ein Ergebnis erreicht werden, daß optimal in Bezug auf die vorgegebenen Anforderungen ist.

Unter Verwendung Evolutionärer Algorithmen für eine global orientierte Parameteroptimierung reeller Variablen (Abschn. 7.2) konnten in fast allen Optimierungen sehr gute Ergebnisse erreicht werden. Die Auswahl der vorgestellten Ergebnisse zeigt dies deutlich. Dabei fand in allen Läufen eine wirkliche Suche nach sehr guten Lösungen statt, wie dies in Abb. 8-27 zu sehen ist. Bis zum Ende der Optimierung wurden immer wieder bessere Lösungen gefunden. Einen genaueren Einblick in 3 dieser Lösungen ist in Abb. 8-28 gegeben.

In der Auswertung wurde bewußt die problemspezifische Visualisierung betont. Der Zielfunktionswert stellt nur einen Wert dar, in der Visualisierung können dagegen mehrere Verläufe gleichzeitig und im Vergleich dargestellt werden. Nur damit ist eine wirklich problembezogene Auswertung dieses Systems möglich. Gleichzeitig erhält man auf diesem Weg ein leistungsfähiges Werkzeug zur Dokumentation der Ergebnisse.

Das Problem der Fahrzeuglenkung kann in mehreren Bereichen erweitert werden:

- Zum einen müssen weitere Szenarien definiert und eingesetzt werden. In den bisherigen Beispielen wird nur mit einer Anfangsabweichung von der Fahrbahnmitte gearbeitet. In erweiterten Szenarien, die über einen längeren Zeithorizont verlaufen, verändert sich z.B. die Fahrbahnmitte während des Szenarios (Kurve). Zudem werden die Szenarien deutlich länger.
- Außerdem wurde die Geschwindigkeit als konstant angesetzt. Bei längeren Szenarien kann die Geschwindigkeit auf einzelnen Streckenabschnitten unterschiedlich sein (z.B. Kurven oder Landstraßen/Ortschaften).

Bei allen Erweiterungen muß überprüft werden, ob das Modell der Fahrzeuglenkung weiterhin gültig ist. Definiert wurde es für die Fahrt auf gerader Strecke.

Nicht umsonst gibt es umfangreiche Arbeiten zur Fahrzeuglenkung, die über das Modell aus Unterabschn. 8.4.1 hinausgehen (z.B. [Mit90] und [Ack93]).

Mit der ausführlichen Darstellung in diesem Abschnitt werden viele Hinweise gegeben, welche direkt auf eigene Optimierungen von Reglern übertragen werden können. Die Zielfunktion gibt einen Einblick in die Möglichkeiten, die mit problembezogenen Zielfunktionen möglich sind und geht deutlich über das hinaus, was im allgemeinen bei der Regleroptimierung zur Bewertung angewendet wird. Zusammen mit der problemspezifischen Visualisierung erhält der Regelungstechniker damit ein leistungsfähiges Werkzeug zum Aufstellen und Erreichen von komplexen Anforderungen, die auf anderem Wege nur schwer erreichbar wären.

## 8.5 Steuerung eines komplexen Systems (Gewächshausklima)

Die Verwendung Evolutionärer Algorithmen für die Berechnung der optimalen Steuerung eines komplexen dynamischen Systems wird am Beispiel der Steuerung der Zustandsgrößen in einem Gewächshaus dargestellt. Diese Arbeit wurde zusammen mit Dr. Adolf Heißner vom Institut für Gemüse- und Zierpflanzenbau Großbeeren/Erfurt in Großbeeren durchgeführt. Dr. Heißner entwickelte die verwendeten Modelle für Gewächshaus und Pflanze.

Die Modelle werden nur so kurz wie für das weitere Verständnis notwendig vorgestellt. Das Hauptaugenmerk liegt auf der Beschreibung der verwendeten Zielfunktion, dem Einsatz problemspezifischen Wissens sowie einer Auswahl der Optimierungsergebnisse.

Die Grundlage der Optimierung ist ein integriertes Modell (Gewächshausklima, Wachstum und Transpiration des Pflanzenbestandes) für die Vorhersage von Temperatur, Luftfeuchtigkeit und CO<sub>2</sub>-Konzentration im Gewächshaus in einem Zeitbereich von 15-60 Minuten (Kurzfristmodell).

Die Optimierung der Steuerung des Gewächshausklimas wurde mit dem Ziel einer Maximierung des Gewinns (Produktion von Gemüse) unter Berücksichtigung von Beschränkungen (z.B. Verhinderung von Stress für die Pflanzen) durchgeführt. Durch die Einbeziehung von aufgabenspezifischem Wissen in den Evolutionären Algorithmus gelang es, wesentlich schneller zu besseren Ergebnissen zu gelangen, als dies mit einem normalen Evolutionären Algorithmus möglich war. Es werden Ergebnisse der Optimierung für durchschnittliche Tage verschiedener Jahreszeiten, bei sich ändernden Preisen und unter Verwendung realer Wetterdaten gezeigt.

### 8.5.1 Einleitung

Die Effektivität der Pflanzenproduktion im Gewächshaus hängt entscheidend von der Einstellung optimaler klimatischer Wachstumsbedingungen zur Erzielung hoher Erträge bei niedrigem Aufwand, guter Qualität und geringer Umweltbelastung ab. Um dies zu erreichen, müssen gleichzeitig mehrere Einflußgrößen wie

Temperatur, Luftfeuchte und CO<sub>2</sub>-Konzentration in jedem Zeitpunkt entsprechend vorgegebener Kriterien durch Heizen, Lüften, CO<sub>2</sub>-Zufuhr u.a. optimal gesteuert werden. Da dieser sich ständig ändernde optimale Zustand in der gesamten Vegetationsperiode einzuhalten ist, ergibt sich eine hochdimensionale Optimierungsaufgabe.

Auf der Basis nichtlinearer Optimierungen mit Ertragsmodellen (Gurke, Tomate) werden für Diskretisierungsintervalle von mehreren Tagen beispielsweise in [Arn87], [Mar90] und [Sch85] Lösungen vorgestellt. Es handelt sich hierbei um dynamische Langfriststeuerungen, die durch entsprechende Steuerungen im Kurzfristbereich (Minuten, Stunden) zu ergänzen sind. Die Langfriststeuerung ist dabei die übergeordnete Steuerung, welche den pflanzenbaulich zulässigen Steuerungsbereich für die untergeordnete Kurzfriststeuerung vorgibt.

In diesem Abschnitt wird eine auf einem einfachen Modell des Gewächshausklimas [Hei96] beruhende Optimierung der Temperatur, Luftfeuchte und CO<sub>2</sub>-Konzentration im Kurzfristbereich (15-60 Minuten) durch Anwendung Evolutionärer Algorithmen vorgestellt.

### 8.5.2 Kurzbeschreibung des Gewächshausklimamodells

Das Gewächshausklimamodell, Abb. 8-31, beschreibt die Abhängigkeit der Temperatur, der Luftfeuchte und der CO<sub>2</sub>-Konzentration im Gewächshaus von den Stellgrößen sowie den mikrometeorologischen Bedingungen im Freien mit einem System von drei nichtlinearen Differentialgleichungen 1. Ordnung.

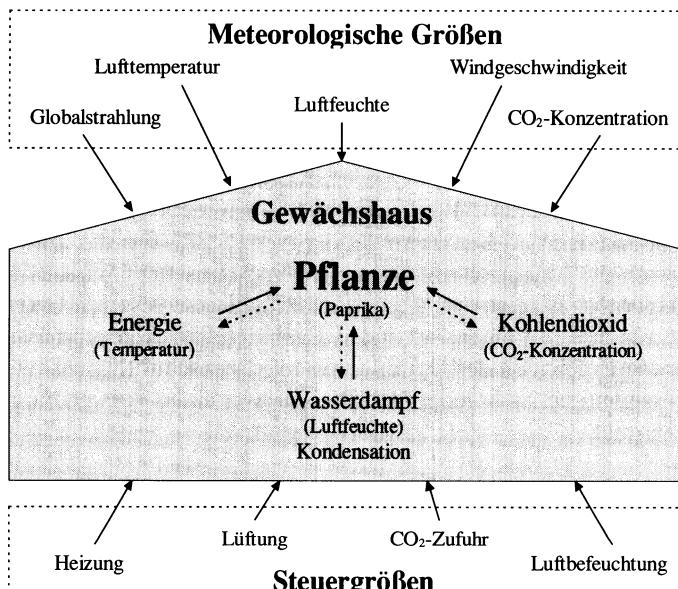


Abb. 8-31. Schema des Gewächshausklimamodells

Die drei Differentialgleichungen des Modells sind die Bilanzgleichungen im Gewächshausinnenraum für 1) Energie, Gl. A2-1, 2) Wasserdampf, Gl. A2-2, und 3) Kohlendioxid, Gl. A2-3.

Das Gewächshausklima kann durch 4 Stellgrößen beeinflußt werden: 1) Heizung,  $Q$  [ $\text{W}/\text{m}^2$ ], 2) Lüftung,  $LR$  [ $\text{m}^3/\text{m}^2$ ], 3)  $\text{CO}_2$ -Zufuhr,  $W$  [ $\text{g}/(\text{m}^2 \cdot \text{h})$ ], und 4) Luftfeuchtigkeit,  $RM$  [ $\text{g}/(\text{m}^2 \cdot \text{h})$ ].

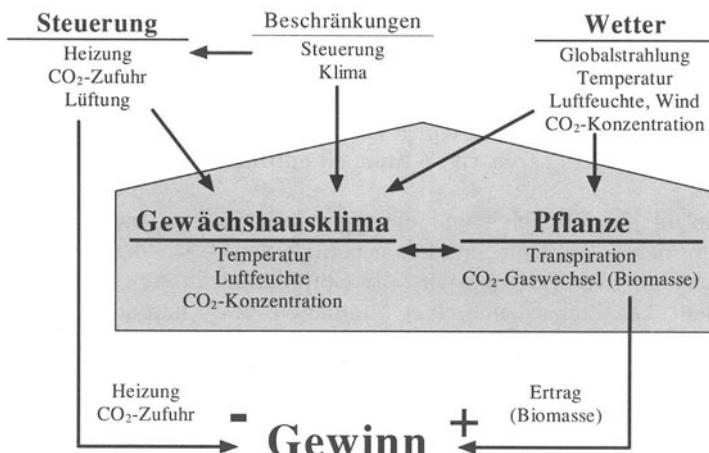
Die zu messenden mikrometeorologischen Bedingungen im Freien sind 1) Globalstrahlung,  $IGLOB$  [ $\text{W}/\text{m}^2$ ], 2) Lufttemperatur außen,  $TEMA$  [ $^\circ\text{C}$ ], 3) Luftfeuchte außen,  $FA$  [% r.F.], 4)  $\text{CO}_2$ -Konzentration außen,  $CA$  [ppm] und 5) Windgeschwindigkeit,  $U$  [m/s].

Im Gewächshausmodell wird nur ein Kompartiment betrachtet (Gewächshausinnenraum mit Pflanzenbestand). Deshalb müssen neben den Bedingungen im Freien weitere Eingangsgrößen berücksichtigt werden: 1) Temperatur der Bodenoberfläche,  $TEMB$  [ $^\circ\text{C}$ ], und 2) Temperatur der Gewächshaushülle,  $TEMG$  [ $^\circ\text{C}$ ].

Die Pflanzen im Gewächshaus werden durch zwei Größen modelliert: 1)  $\text{CO}_2$ -Gaswechsel, Gl. A2-4, und 2) Transpiration, Gl. A2-5. Aus dem  $\text{CO}_2$ -Gaswechsel wird die Biomasseproduktion berechnet, Tabelle A2-1. Für die Simulationen wurden die Modelle für Paprika verwendet [Hei97].

Die in den Bilanzgleichungen verwendete Temperaturgröße ist durch die spezifische Wärmekapazität von Luft und Pflanzen (Volumen pro Grundfläche und Wasseräquivalent des Pflanzenbestandes) im Gewächshaus definiert und stellt damit ein Mittel aus Luft- und Pflanzentemperatur dar.

Die Berechnungen mit dem Modell benutzen eine Reihe von physikalischen Konstanten, Gewächshausparametern und -hilfsgrößen sowie Koeffizienten für die Darstellung der Transpiration und des  $\text{CO}_2$ -Gaswechsels durch Modelle und Pflanzenbestandsparameter ([Hei96], [PH96a]).



**Abb. 8.32.** Abhängigkeiten der Zielfunktion

Eine ausführlichere Darstellung des Gewächshausklimamodells und der Pflanzenmodelle wird in Abschn. A.2, S.276, sowie in [Hei96] und [Hei97] gegeben.

Für die Optimierung sind insbesondere das Zusammenspiel und die gegenseitigen Abhängigkeiten der verschiedenen Größen und ihre Auswirkungen auf die Bewertung, hier den Gewinn, wichtig. In Abb. 8-32 sind diese Abhängigkeiten der Zielfunktion dargestellt. Im Gewächshaus befinden sich die Pflanzen. Deren Wachstum wird durch das Klima im Gewächshaus und einen Teil der äußeren Wetterbedingungen beeinflußt. Das Gewächshausklima wiederum wird durch die äußeren Wetterbedingungen, die Steuerung und die vorgegebenen Beschränkungen beeinflußt. Zusätzlich wirken auch die Pflanzen auf das Gewächshausklima zurück. Aus der Biomasseproduktion der Pflanzen kann ein Ertrag errechnet werden, der vermindert um die Aufwendungen für die Steuerung den Gewinn ergibt. Diese immer noch vereinfachende Übersicht zeigt recht deutlich die komplexen Zusammenhänge und Abhängigkeiten der Größen des Gewächshausklimamodells.

### 8.5.3 Repräsentation der Individuen und Zielfunktion

Jedes Individuum im Evolutionären Algorithmus repräsentiert die Steuerung für eine Simulationsperiode. Wie in Unterabschn. 8.5.2 gezeigt wurde, kann das Gewächshausklima durch 4 Größen gesteuert werden. Im Moment werden 3 Größen verwendet: Heizung, Lüftung und CO<sub>2</sub>-Zufuhr. Die Luftbefeuhtung wird zur Zeit nicht verwendet und ist die gesamte Zeit auf 0 gesetzt.

Die Steuerungen werden zu äquidistanten Zeitpunkten diskretisiert. Für die Simulation wird ein Halteglied 1. Ordnung verwendet, um Werte der Steuerungsgrößen zwischen den Diskretisierungspunkten zu erhalten. Die Anzahl der Variablen kann mittels Gl. 8-21 berechnet werden:

$$NumVar = \left( \frac{SimTime}{ControlStep} + 1 \right) \cdot NumControl; \quad ControlStep = 0.25 \text{ h} \\ NumControl = 3 \quad (8-21)$$

Ein Steuerungsschritt alle 15 Minuten ist für die Simulation klein genug und hält gleichzeitig die Anzahl der Variablen so klein wie vertretbar. Damit ergibt sich, daß für eine Simulationszeit von 4 Stunden ein Individuum aus 51 Variablen besteht.

Um die Anzahl der Variablen und die Optimierungszeiten handhabbar zu halten, wird normalerweise eine Simulationsperiode von 4 Stunden verwendet. Zur Optimierung längerer Perioden wird die Optimierung in Abschnitte von 4 Stunden unterteilt. Die Endzustände einer Simulation bzw. Optimierung dienen dann als Startwerte für die folgende Periode. Damit ist es möglich, selbst sehr lange Simulationszeiträume auf Standardhardware (PC Pentium-200, 64 MB RAM) zu berechnen. Sobald leistungsfähigere Hardware zur Verfügung steht, kann die Simulationsperiode entsprechend verlängert werden.

Die im Modell verwendete Zielfunktion, *Cost*, Gl. 8-22, dient zur Maximierung des Gewinns, Gl. A2-6, unter der Bedingung der Erfüllung der Beschränkungen, s. Unterabschn. A.2.4, S.280. Um die Einhaltung der Beschränkungen zu erreichen, wird den Individuen bei Verletzung der Beschränkungen ein Be-

strafungswert (*Penalty*) zugewiesen. Da bei der Optimierung mit einer Minimierung gearbeitet wird, muß in Gl. 8-22 der Gewinn mit -1 multipliziert werden.

$$Cost = - \int_{T_S}^{T_E} GEWI \, dt + Penalty \quad (8-22)$$

$T_S$ : Startzeit,  $T_E$ : Endzeit der Simulation

Für die Bestrafungsfunktion *Penalty*, Gl. 8-23, kommt eine gewichtete Summe zur Anwendung. *Val* entspricht entweder einer der Zustandsvariablen des Gewächshausklimas oder einer der Steuerungsgrößen. *Constr* ist die jeweils dazugehörige Beschränkung. Durch Wahl der Gewichte *W* kann die Wichtigkeit der Einhaltung dieser Beschränkung definiert werden.

$$Penalty = \sum_{i=1}^{NumConstr} \left( W_i \cdot \int_{T_S}^{T_E} \left| (Val_i - Constr_i) > 0 \right| dt \right) \quad (8-23)$$

*NumConstr*: Anzahl der Beschränkungen

Die Verwendung einer gewichteten Summe zur Berechnung des Bestrafungswertes in Gl. 8-23 gewährleistet, daß Lösungen mit einer stärkeren Verletzung der Beschränkungen einen höheren Bestrafungswert und Individuen mit einer geringeren Verletzung der Beschränkungen einen kleineren Bestrafungswert erhalten. Dadurch sind Lösungen, welche die Beschränkungen verletzen, nicht einfach schlecht, sondern es gibt eine Reihenfolge unter diesen Lösungen. Dies ist besonders dann wichtig, wenn gute Lösungen sehr nah an den Beschränkungen liegen bzw. leichte Verletzungen der Beschränkungen noch tolerierbar sind. Bei der Optimierung der Steuerung des Gewächshausklimas ist dies für mehrere der Beschränkungen der Fall.

#### 8.5.4 Einbeziehung problemspezifischen Wissens

Für die Steuerung des Gewächshausklimas gibt es Vorwissen, das in die Optimierung einbezogen werden kann und sollte. Dieses problemspezifische Vorwissen kann in zwei Bereiche unterteilt werden:

- Vorgabe von guten Steuerstrategien, die auf durchschnittlichen bzw. typischen Klimadaten beruhen, (benutzt für die Initialisierung der Suche) und
- Einschränkung des Suchbereiches der Variablen durch bekannte Beschränkungen der Steuergrößen.

Aus der Optimierung der Gewächshausklimasteuerung unter Verwendung durchschnittlicher Tage, s. Unterabschnitt 8.5.5, ab S.244, ergeben sich Steuerstrategien für die jeweiligen Monate, die auf den durchschnittlichen Klimadaten basieren. Diese Strategien für die Steuervariablen als Ergebnis von Optimierungen bilden eine Grundlage für die Initialisierung der Anfangspopulation.

Für die Optimierung unter Verwendung realer Wetterdaten, s. Unterabschn. 8.5.7, ab S.247, wurde jeweils eine gute Steuerstrategie für jeden Monat berechnet. Jede dieser Steuerstrategien besteht aus einem kompletten Tagsgang der Steuergrößen Heizung, Lüftung und Kohlendioxidzufuhr. Für die

Initialisierung der Anfangspopulation werden nicht nur die Steuerstrategie des aktuellen Monats verwendet, sondern auch die Strategien der zwei vorhergehenden und der zwei folgenden Monate. Die Überlegung dabei ist, daß dadurch auch Strategien für kühlere bzw. wärmere Tage im Vergleich zum Durchschnitt des aktuellen Monats in die Initialisierung aufgenommen werden. Außerdem werden neben der reinen Verwendung dieser Strategien weitere Strategien gebildet, indem die Werte der einzelnen Steuergrößen zwischen den Monaten ausgetauscht werden. Am Ende werden alle Vorgaben in einem geringen Maß verrauscht.

Mit dieser Initialisierung konnten sehr gute Erfahrungen gesammelt werden. Meistens ist eine der in der Anfangspopulation enthaltenen Strategien schon recht gut, zumindest werden viele der Beschränkungen eingehalten. Für den Optimierungsalgorithmus steht „nur“ noch die Aufgabe, in der näheren Umgebung dieser Lösung nach besseren Varianten zu suchen. Wenn in der Anfangspopulation noch keine sehr gute Lösung enthalten ist, so sind in dieser doch viele sinnvolle Elemente enthalten, die durch den Evolutionären Algorithmus innerhalb einiger Generationen zu einem sehr guten Ergebnis kombiniert werden.

Ein weiterer Bereich der Verwendung problemspezifischen Wissens ist die Beschränkung des möglichen Wertebereiches der Steuervariablen in Abhängigkeit des Monats und der Tageszeit. Aus den Ergebnissen von Optimierungen und aus der Praxis wurden für jede Stunde eines Tages in jedem der zwölf Monate ein Maximal- und ein Minimalwert für jede der drei Steuervariablen definiert. Dies bedeutet z.B., daß im Winter die Lüftung einen deutlich kleineren Maximalwert hat als im Sommer, dasselbe gilt für die Nacht. Im Sommer ist dafür der Maximalwert der Heizung kleiner als im Winter. Ähnliche Unterschiede wurden zwischen Tag und Nacht festgelegt.

Bei der Festlegung von Beschränkungen für die Steuervariablen muß beachtet werden, daß dadurch der Suchraum für die Optimierung nicht so stark eingeschränkt wird, daß eine Einhaltung der Beschränkungen der Gewächshausklimagrößen nicht mehr möglich ist. Beispiele dafür sind ein kühler Tag im Sommer, ein sonnenreicher Tag im Frühjahr bzw. eine sehr kalte Nacht im Herbst.

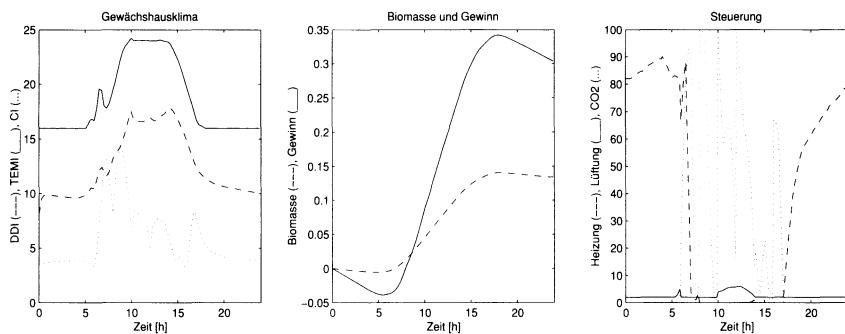
Unter Verwendung des in diesem Unterabschnitt vorgestellten problemspezifischen Vorwissens kann die Suche nach Lösungen robuster durchgeführt und stark beschleunigt werden. Dies zeigt sich insbesondere bei den unter Verwendung realer Wetterdaten durchgeföhrten Optimierungen.

### **8.5.5 Optimaler Steuerung für durchschnittliche Tage**

Dieser Unterabschnitt zeigt die Ergebnisse der Optimierungen des integrierten Gewächshausklimamodells für durchschnittliche Tage im April und Juni. Für die Berechnung des Gewinns wurden die Preise aus Tabelle A2-1, S.280, benutzt. Abweichend zu den in Unterabschn. A.2.4, S.280, aufgeführten Beschränkungen, wurde in diesem Unterabschnitt mit einer maximalen Temperatur im Gewächshaus von 24°C gearbeitet.

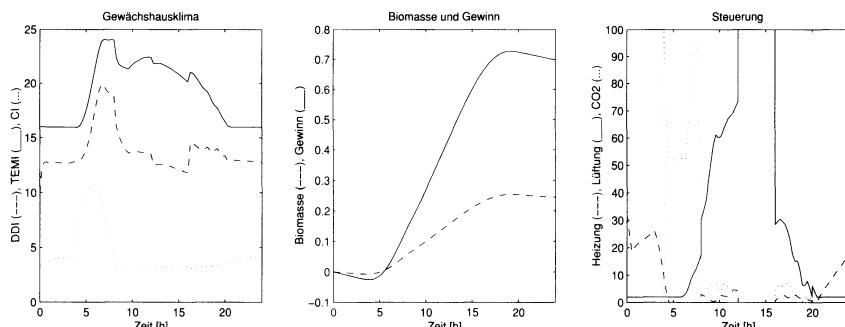
Die verwendeten meteorologischen Wetterdaten in diesem und dem nächsten Unterabschnitt sind Durchschnittswerte über 10 Jahre für Potsdam/Deutschland. Deshalb sind diese Daten glatt und ohne Störungen. Die Blattfläche pro Fläche ist im Februar am kleinsten ( $0,5 \text{ m}^2/\text{m}^2$ ) und im Juni am größten ( $2,5 \text{ m}^2/\text{m}^2$ ).

Das jeweils linke Diagramm der Abbildungen 8-33 – 8-36 zeigt die Verläufe der Zustandsvariablen des Gewächshauses: Luftfeuchtigkeit,  $DDI$ , Gl. A2-2, Temperatur,  $TEMI$ , Gl. A2-1, und  $\text{CO}_2$ -Konzentration,  $CI$ , Gl. A2-3. Die  $\text{CO}_2$ -Konzentration wurde zur Skalierung auf den Wertebereich der anderen beiden Variablen durch 100 dividiert. Die Verläufe von Biomasse,  $BIOM$ , und Gewinn,  $GEWI$ , Gl. A2-6, sind jeweils im mittleren Diagramm dargestellt (summiert über die Zeit – *cumulative sum*). Die Biomasse wurde durch 100 dividiert, um mit dem Gewinn in einem Diagramm dargestellt werden zu können. Das jeweils rechte Diagramm zeigt die Verläufe der 3 Stellgrößen: Heizung,  $Q$ , Lüftung,  $LR$  und  $\text{CO}_2$ -Zufuhr,  $W$ , wobei die Werte der  $\text{CO}_2$ -Zufuhr mit 10 multipliziert wurden.



**Abb. 8-33.** Ergebnisse der Optimierung durchschnittlicher Tage für den Monat April

Im April, Abb. 8-33, ist die Heizung die gesamte Nacht an. Dadurch wird die Einhaltung der unteren Temperaturschwelle von  $16^\circ\text{C}$  gesichert. Während des Tages ist die Heizung nicht notwendig. Die Lüftung wird nur über die Mittagszeit in geringem Maße benutzt, wodurch die Temperatur unter der oberen Schwelle von  $24^\circ\text{C}$  gehalten wird. Da die Lüftung den gesamten Tag über recht gering oder ausgeschaltet ist, wird ständig Kohlendioxid zugeführt, wodurch die  $\text{CO}_2$ -Konzentration tagsüber hoch gehalten wird.

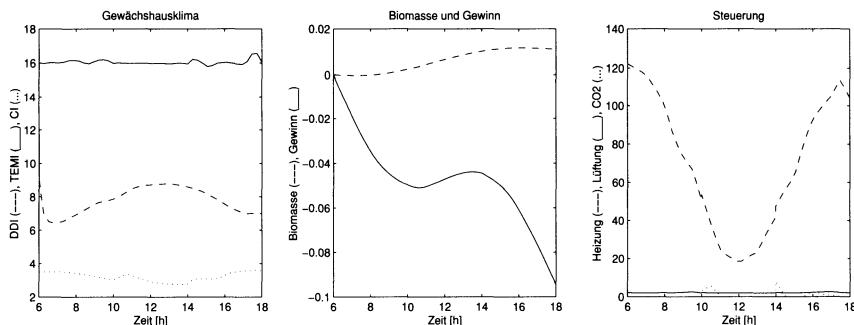


**Abb. 8-34.** Ergebnisse der Optimierung durchschnittlicher Tage für den Monat Juni

Im Juni, Abb. 8-34, ist das Ergebnis ein ganz anderes. Die Lüftung ist den gesamten Tag an, um die Temperatur niedrig zu halten. Am Morgen, wenn die Lüftung noch ausgeschaltet ist, aber die Sonne bereits scheint (Sonnenaufgang etwa gegen 4 Uhr), wird Kohlendioxid zugeführt, um die CO<sub>2</sub>-Konzentration zu erhöhen. Dies führt zu einer stärkeren Biomasseproduktion. Sobald die Lüftung zur Einhaltung der oberen Temperaturschwelle geöffnet wird, ist CO<sub>2</sub>-Anreicherung nicht länger sinnvoll. Die Heizung wird nur während der Nacht in geringem Maße benutzt, damit die untere Temperaturschwelle von 16°C eingehalten wird.

### 8.5.6 Optimale Steuerung bei veränderten Preisen

Dieser Unterabschnitt zeigt die Ergebnisse von Optimierungen, bei denen im Vergleich zum vorigen Abschnitt veränderte Preise zugrunde gelegt wurden. Für die folgenden Berechnungen wurde der Preis für Heizung (Energie) und die CO<sub>2</sub>-Zufuhr gegenüber dem Standardpreis verändert. Abhängig von der Größe der Änderung kann sich der Verlauf der Stell- und Zustandsgrößen nur etwas oder stark verändern. Da die verwendeten Wetterdaten Durchschnittswerte und damit glatt und ohne Störungen sind, wurden hier zwei drastische Änderungen der Preise eingesetzt, um die Auswirkungen zeigen zu können.

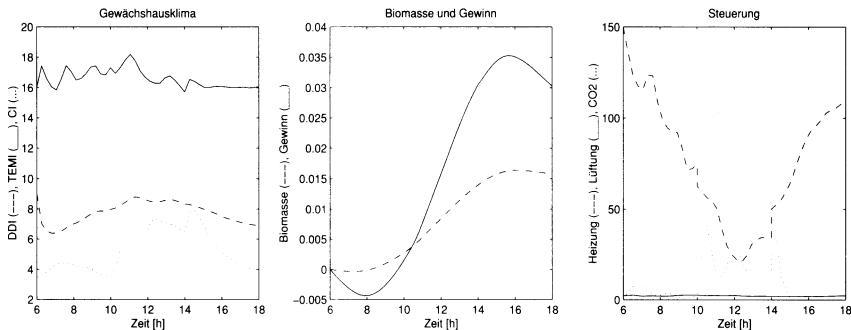


**Abb. 8-35.** Ergebnisse der Optimierung durchschnittlicher Tage für den Monat Februar unter Zugrundelegung eines hohen Energie- und Kohlendioxidpreises

Bei den in Abb. 8-35 gezeigten Ergebnissen waren der Energie- und Kohlendioxidpreis 3 mal so hoch wie der Standardpreis aus Tabelle A2-1. Im Ergebnis ist die Heizung so gering wie möglich, um trotzdem noch die untere Temperaturbeschränkung einzuhalten. Während des Tages wird so gut wie kein Kohlendioxid zugeführt, die Lüftung wird nie eingeschaltet.

Unter Zugrundelegung eines niedrigen Energie- und Kohlendioxidpreises (<sup>1</sup>/<sub>3</sub> des Standardpreises aus Tabelle A2-1), Abb. 8-36, ergibt sich ein anderes Bild. Hier ist die Heizung höher, als es die Einhaltung der unteren Temperaturbeschränkung erfordert. Zusätzlich wird während des Tages Kohlendioxid zugeführt. Die im Verlauf des Tages produzierte Biomasse (BIOM=1,5 g/m<sup>2</sup>) ist hö-

her, verglichen mit der unter erhöhten Preisen erreichten Biomasseproduktion ( $\text{BIOM}=1,1 \text{ g/m}^2$ ). Durch die niedrigeren Preise ist der Gewinn viel größer und positiv.



**Abb. 8-36.** Ergebnisse der Optimierung durchschnittlicher Tage für den Monat Februar unter Zugrundelegung eines niedrigen Energie- und Kohlendioxidpreises

### 8.5.7 Optimale Steuerung unter Verwendung realer Wetterdaten

Dieser Unterabschnitt zeigt eine Auswahl der Ergebnisse von Optimierungen des integrierten Gewächshausklimamodells unter Verwendung realer Wetterdaten des Jahres 1995 in Großbeeren bei Berlin. Für die Berechnung des Gewinns wurden die Preise aus Tabelle A2-1, S.280, benutzt.

Die Darstellung der Ergebnisse weicht etwas von der in den vorigen Unterabschnitten ab. Einmal werden Ergebnisse für jeweils mehr als einen Tag in einer Grafik dargestellt. Außerdem enthalten die Abbildungen zusätzlich eine Grafik mit den verwendeten Klimagrößen. Dadurch ist schnell zu erkennen, worin die Ursachen bestimmter Steuerstrategien liegen. Alle Grafiken enthalten Legenden, die zusätzlich zu den Namen der Größen die teilweise verwendeten Skalierungsfaktoren der einzelnen Größen enthalten. Die Zeitachse aller Grafiken bezieht sich auf den entsprechend fortlaufend nummerierten Tag des Jahres.

Die Optimierung des integrierten Gewächshausklimamodells wurde für den Zeitraum April – September 1995 durchgeführt. Die folgenden Abbildungen enthalten Ausschnitte aus diesem Zeitraum. Es wurden solche Gruppen von aufeinanderfolgenden Tagen ausgewählt, die typisch für den entsprechenden Monat sind bzw. in denen auf Grund von Wetteränderungen deutlich unterschiedliche Steuerstrategien innerhalb der einzelnen Tage zur Anwendung kamen.

Der für den Monat April in Abb. 8-37 dargestellte Zeitraum umfasst fünf Tage. Das Klima dieser Tage unterscheidet sich am stärksten in der gemessenen Globalstrahlung. Am ersten Tag ist die Globalstrahlung im mittleren Bereich. Die folgenden beiden Tage wurde eine wesentlich geringere Globalstrahlung gemessen. An den letzten beiden Tagen dagegen war die Globalstrahlung sehr hoch, wobei der Wert am fünften Tag noch höher als am vierten Tag war. Entspre-

chend dieser Unterteilung lassen sich unterschiedliche Verläufe für die Steuer- und Klimagrößen beobachten.

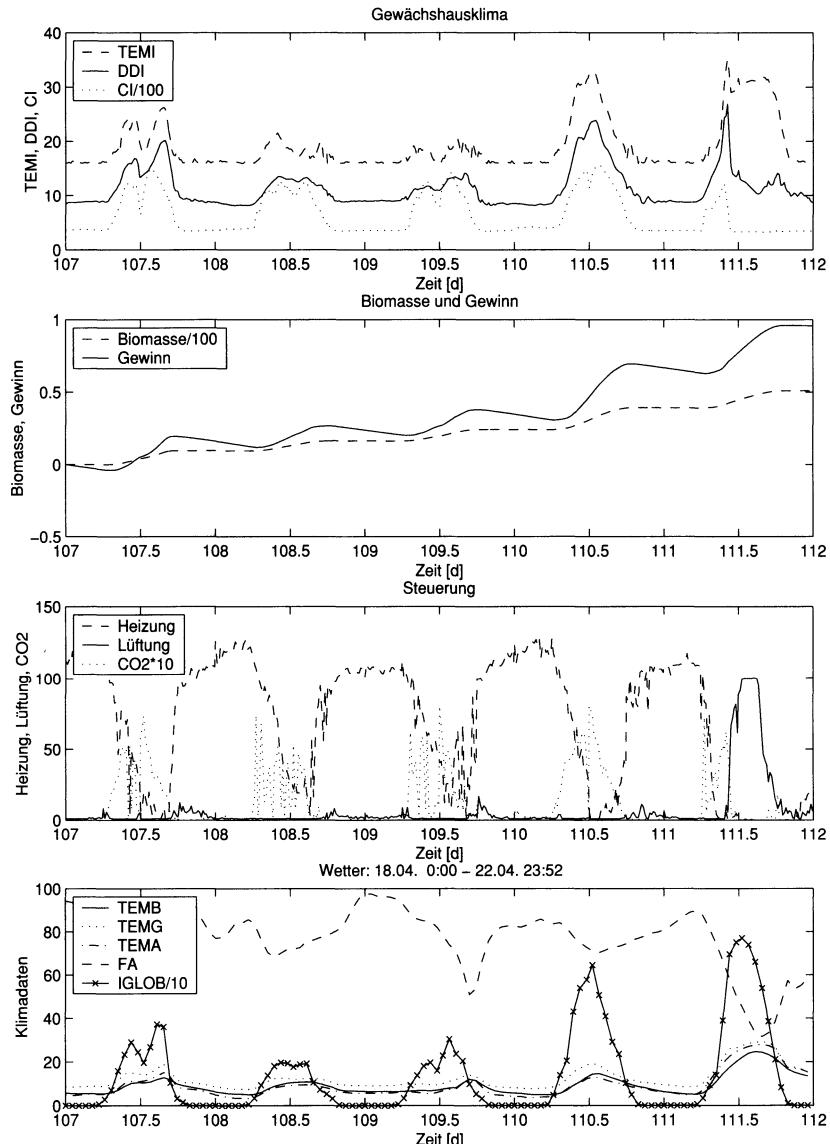


Abb. 8-37. Gewächshaussteuerung unter Verwendung realer Wetterdaten (5 Tage im April)

An den ersten vier Tagen wird tagsüber keine Lüftung benötigt. Dadurch kann durch Kohlendioxidzufuhr die Kohlendioxidkonzentration im Gewächshaus er-

höht werden. In der oberen Grafik ist deutlich die Zunahme der Kohlendioxidkonzentration tagsüber für diese Tage zu erkennen. Am fünften Tag sieht die Steuerung und damit der Verlauf der Klimagrößen deutlich anders aus. Ab dem frühen Morgen wird die Lüftung benutzt, ab Mittag ist die obere Beschränkung der Lüftungsrate erreicht. Eine Anreicherung mit Kohlendioxid wird nicht vorgenommen, da durch die hohe Lüftungsrate eine Erhöhung der Kohlendioxidkonzentration zu teuer ist.

An allen Tagen muß nachts geheizt werden, um den vorgegebenen Minimalwert der Innentemperatur nicht zu unterschreiten. Am zweiten und dritten Tag muß allerdings auch tagsüber, wenn auch deutlich weniger, geheizt werden, um die untere Temperaturschranke einzuhalten. Die erhöhte Biomasseproduktion an den Tagen mit höherer Globalstrahlung ist in der zweiten Grafik von oben zu erkennen. Da zusätzlich weniger geheizt werden muß, wird der Gewinn noch weiter erhöht. Besonders deutlich ist dies am fünften Tag zu erkennen.

Die Ergebnisse der ausgewählten Tage im Monat Mai sind in Abb. 8-38 dargestellt. Besonders interessant für diese Periode ist der dritte Tag. An diesem Tag war die Globalstrahlung gering. Entsprechend groß waren die Auswirkungen auf die für diesen Tag berechnete Steuerstrategie. Über den gesamten Tag mußte geheizt werden, um die untere Temperaturschwelle im Gewächshaus zu halten. Weiterhin ist zu sehen, daß nur in der Zeit Kohlendioxid zugeführt wird, in der auch Globalstrahlung vorhanden war, ein erkennbar kürzerer Zeitraum, als an den anderen Tagen. In der kurzen Zeit, in der ausreichend Globalstrahlung für eine Zunahme der Biomasse vorhanden war, konnte allerdings nicht so viel Biomasse produziert werden, wie über Nacht wieder verbraucht wurde. In der zweiten Grafik von oben ist zu erkennen, daß es am dritten Tag zu einer deutlichen Verringerung der Biomasse kam.

Abbildung 8-39 enthält die Ergebnisse für Tage des Monats Juni. Bei einem ersten Blick auf den Verlauf der Biomassezunahme und des Gewinns, zweite Grafik von oben, sehen alle fünf Tage sehr ähnlich aus. Ein Blick auf die anderen Grafiken zeigt aber, daß dies nicht so ist. Am zweiten und dritten Tag ist die Globalstrahlung geringer, als an den anderen Tagen. Es wird keine Lüftung benötigt, wodurch wiederum Kohlendioxid zugeführt werden kann. An den anderen Tagen wird nur in der Zeit am Morgen Kohlendioxid zugeführt, in der die Lüftung noch nicht geöffnet ist. Die geringeren Außentemperaturen haben für die Nächte zum zweiten, dritten und vierten Tag zur Folge, daß geheizt werden muß.

Die in diesem Unterabschnitt gezeigten Ergebnisse der Optimierung mit dem Gewächshausklimamodell unter Verwendung realer Klimadaten zeigen die direkte Reaktion der Steuergrößen auf sich verändernde Klimagrößen. Auf den ersten Blick nicht sehr groß erscheinende Unterschiede in den klimatischen Bedingungen verschiedener Tage können zu deutlichen Unterschieden in den berechneten Steuerstrategien führen. Ein Beispiel dafür ist der Unterschied der Steuerstrategien für den vierten und fünften Tag der dargestellten Tage im April, Abb. 8-37. Am fünften Tag sind die Globalstrahlung und die Außentemperatur höher als an den anderen Tagen im April. Dadurch muß die Lüftung verwendet werden, eine Kohlendioxidzufuhr macht keinen Sinn. Trotz der höheren Globalstrahlung am fünften Tag ist die Biomasseproduktion am vierten Tag auf Grund der höheren Kohlendioxidkonzentration höher. Ähnliche Beispiele lassen sich an verschiedenen anderen Stellen finden.

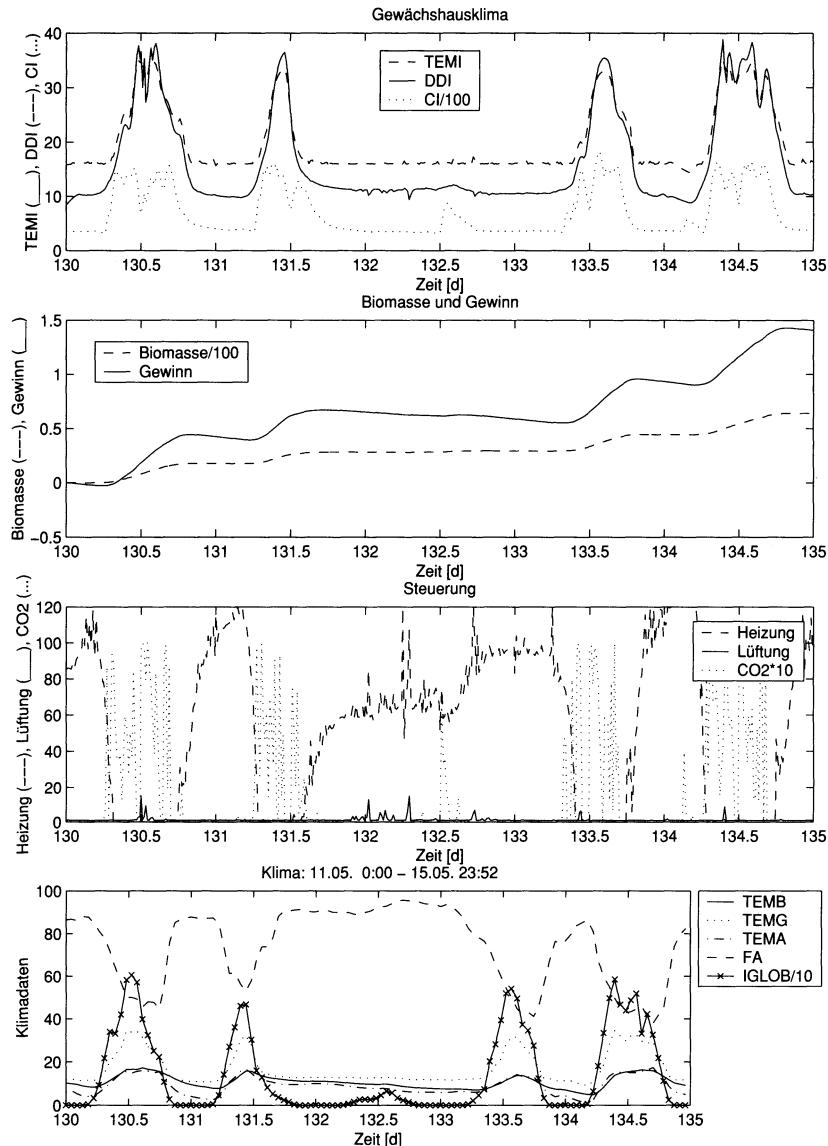


Abb. 8-38. Gewächshaussteuerung unter Verwendung realer Wetterdaten (5 Tage im Mai)

In den Erläuterungen dieses Unterabschnitts wurde nur auf einige der Abhängigkeiten bzw. erklärenden Ursachen für bestimmte Steuerungen oder Effekte eingegangen. Bei der Optimierung und Simulation des Modells werden viele weitere Abhängigkeiten einbezogen, die sich in den Grafiken oftmals nicht so einfach ablesen lassen. Die hier dargestellten Ausschnitte aus einer mehrere Monate umfassenden Optimierung der Steuerung zeigen die Funktionsweise des

verwendeten Gewächshausklimamodells und der Optimierungsmethode bei der Verwendung realer Wetterdaten.

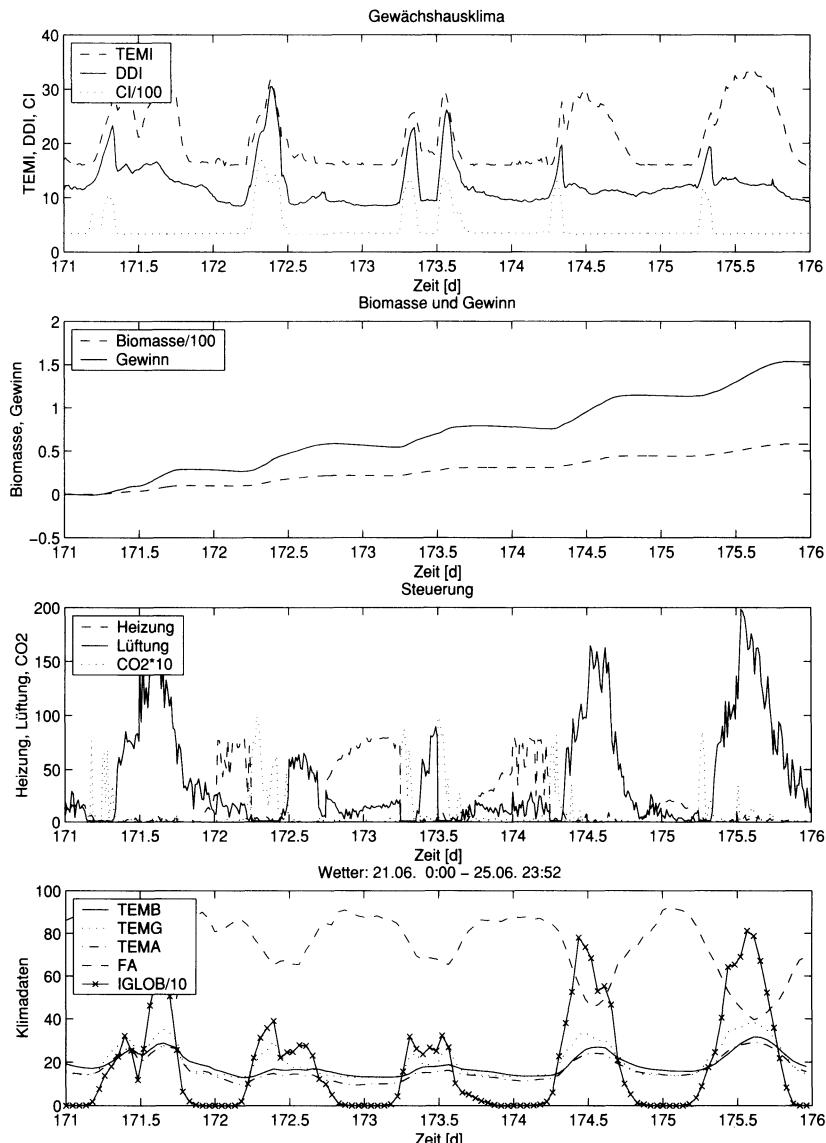


Abb. 8-39. Gewächshaussteuerung unter Verwendung realer Wetterdaten (5 Tage im Juni)

### 8.5.8 Zusammenfassung Gewächshausklima

Die in diesem Abschnitt vorgestellte Optimierung der Steuerung des Gewächshausklimas verwendet einen anderen Ansatz als viele andere Arbeiten auf dem Gebiet der Gewächshausklimatisierung. Es wird nicht, wie sonst üblich, nach der besten Reglereinstellung bzw. -vorgabe für Temperatur und Kohlendioxidkonzentration im Gewächshaus gesucht. Statt dessen werden direkt die Steuergrößen Heizung, Lüftung und Kohlendioxidzufuhr optimiert. Durch diese direkte Beeinflussung des Gewächshausklimas ist eine vollständige Steuerung aller beeinflußbaren Zustandsgrößen möglich. Außerdem ist durch das Kurzfristmodell eine detaillierte Steuerung und damit Beeinflussung der Zustandsgrößen erreichbar.

Die ermittelten Steuerstrategien für Heizung, Lüftung und CO<sub>2</sub>-Anreicherung zur Einstellung optimaler Klimazustände im Gewächshaus in Abhängigkeit von der Jahreszeit und den Preisen stehen im Einklang mit der Erfahrung und theoretischen Erkenntnissen. Die Einsatzfähigkeit der vorgestellten Optimierungsmethode zeigen insbesondere die Szenarienrechnungen mit realen Wetterdaten, die sich über einen großen Teil einer Vegetationsperiode ausdehnen. Die dargestellten Ausschnitte zeigen nur einzelne Tage bzw. Wochen, die charakteristisch für längere Perioden sind bzw. Extremwerte darstellen. Die Optimierung wurde für die Monate April bis September komplett durchgeführt.

Eine Weiterentwicklung der zugrunde gelegten Optimierungsmethode und die gezielte Anwendung auf spezifische Aufgaben der Gewächshausklimasteuerung ist somit gerechtfertigt. Für eine Online-Optimierung muß weiter versucht werden, die gegenwärtigen Rechenzeiten (15-30 Minuten für ein Optimierungsintervall von 4 Stunden) zu reduzieren. Möglichkeiten dazu werden u.a. in der weiteren Einbeziehung problemspezifischen Wissens und in der Weiterentwicklung der Algorithmen, z.B. durch Entwicklung problemspezifischer Rekombinations- und Mutationsoperatoren, gesehen.

Ein nächster Schritt der Anwendung der hier vorgestellten und optimierten Steuerung des Gewächshausklimas ist die Einbeziehung verschiedener Strategien zur Vorgabe unterschiedlicher Optimalitätskriterien und Beschränkungen, z.B. durch Einbeziehung übergeordneter Langfristmodelle des Wachstums. Diese Kopplung von Langfristmodellen mit Kurzfristmodellen stellt eine notwendige Voraussetzung für den späteren praktischen Einsatz einer solchen Steuerung dar. Als Langfristmodelle können z.B. die in den Arbeiten von ARNOLD [Arn87], MARKERT [Mar90] und SCHMIDT [Sch85] verwendeten Modelle und optimierten Steuerstrategien zum Einsatz kommen. Die Kopplung zwischen den Lang- und Kurzfristmodellen ist aber nicht so einfach. Eine offene Frage ist die Definition der Zielkriterien und welches Modell bzw. welche Optimierung welche Kriterien behandelt.

Die Kopplung von Modellen unterschiedlicher Abstraktionsebenen (Lang-, Mittel- und Kurzfristmodell) wurde von ALSCHER in [Als93] vorgestellt. Sie verwendete ein Phasenmodell SAPHASE (Langfristmodell) für die Berechnung der Steuerstrategien über eine Wachstumsphase. Diesem war ein Tagesmodell SATAG nachgeordnet, das zur Bestimmung der Steuerstrategie eines Tages diente. Das Tagesmodell wurde mit einem Stundenmodell SASTUNDE (Kurzfristmodell) unterbaut, das die Steuerstrategien für die stündliche kurzfristige Steuerung berechnet. Es war ein zusätzliches Kopplungsmodell

SAKOPPEL notwendig, das die voneinander abweichenden Vorhersageergebnisse der einzelnen Modelle miteinander koppelt und zu einer einheitlichen Bewertung führt. Eine Optimierung fand unter Anwendung der Simulationsmethode statt. Damit wurden die Ergebnisse aller Kombinationen einer Anzahl von vorgegebenen Reglereinstellungen durch Simulation der Modelle ermittelt und die Kombination der Reglereinstellungen mit dem besten Ergebnis (direkt- und flächenkostenfreie Leistung) als optimale Lösung verwendet. Eine zeitabhängige Veränderung der Reglereinstellung über den Simulationszeitraum war nicht vorgesehen, für jedes der Modelle wurde mit einer konstanten Reglereinstellung gearbeitet.

Das Problem der im allgemeinen im voraus nicht ausreichend genau bekannten Klimagrößen lässt sich durch Klimavoraussagen und die Anwendung der repetierenden Optimierung lösen. Mit einer langfristigen Klimavorhersage (ein bis mehrere Tage) wird eine erste Optimierung durchgeführt. Diese dient als Grundlage für weitere Optimierungen, die auf einer kurzfristigeren Wettervorhersage basieren. Je nach Güte der Wettervorhersage kann damit eine ausreichende Übereinstimmung zwischen den mit der berechneten optimalen Steuerung einzustellenden Zustandsgrößen und den tatsächlich gemessenen Zustandsgrößen erreicht werden. Bei immer noch zu großen Abweichungen muß in einer Wiederholung der Optimierung eine weitere Verkürzung des Vorhersagezeitraumes unternommen werden.

Die in dieser Arbeit verwendete spezielle Initialisierung kann als eine repetierende Optimierung angesehen werden. Die Daten für die Initialisierung stammen aus einer Optimierung der Gewächshaussteuerung unter Verwendung von Klimadaten durchschnittlicher Tage dieser Jahreszeit bzw. des entsprechenden Monats. In der nachfolgenden Optimierung unter Verwendung realer Klimadaten wurden die Ergebnisse der ersten Optimierung für die Initialisierung genutzt. Diese nachfolgende Optimierung mußte „nur“ noch die Anpassung auf die veränderten Klimagrößen vornehmen.

Eine repetierende Optimierung wurde in den Arbeiten von ARNOLD [Arn87] und MARKERT [Mar90] bei der Anwendung einer Langfriststeuerung vorgestellt und verwendet. Auch ALSCHER verwendete in [Als93] eine repetierende Optimierung.

Durch eine weitere Verbesserung und Erweiterung der Pflanzenmodelle ( $\text{CO}_2$ -Gaswechsel, Transpiration, Wachstum) und des Gewächshausmodells kann die Genauigkeit der Vorhersagen weiter verbessert werden. Dies würde einen erweiterten und verbesserten Einsatz der Modelle ermöglichen.

## 8.6 Zusammenfassung

In diesem Kapitel wurde die systemtechnische Anwendung Evolutionärer Algorithmen für die Lösung einiger praktischer Probleme demonstriert. Den Anfang bildete die Darstellung des prinzipiellen Vorgehens bei der Lösung neuer Optimierungsprobleme. Dies sollte helfen, die Schritte bei der Optimierung zu verdeutlichen und das systematische Erreichen von Ergebnissen zu erleichtern. Die

nachfolgenden Anwendungen wurden auf wenige Beispiele beschränkt, die jeweils aus unterschiedlichen Bereichen kommen.

Schrittweise wurde die Komplexität der vorgestellten Anwendungen erhöht. Ausgangspunkt waren einfache mehrdimensionale Funktionen in Abschn. 8.2. Diese stellen Testfunktionen dar, die in vielen technischen Berichten als Beispiele verwendet werden. Zwei häufig zu lösende Aufgaben waren die Parameteridentifikation eines Modells (Abschn. 8.3) und die Parameterdimensionierung eines Reglers (Abschn. 8.4).

Die umfangreichste Anwendung stellte die Optimierung des Gewächshausklimas in Abschn. 8.5 dar. Dieses Problem beinhaltet in einem Teilmodell ein biologisches System, das in seiner Interaktion mit den äußeren Zuständen ein äußerst komplexes System bildet. Erst durch eine Reduktion der Komplexität des biologischen Systems, wie es in dem hier verwendeten Modell stattgefunden hat, ist eine Bearbeitung und Optimierung des Gesamtsystems mit heutigen Mitteln möglich.

Es wurde gezeigt, daß sich komplexe praktische Probleme durch die Einbeziehung von problemspezifischem Wissen besser und schneller lösen lassen, als wenn dieses Wissen nicht verwendet wird. Beim Einsatz Evolutionärer Algorithmen ist es möglich, dieses Wissen in den Optimierungsprozeß einzubeziehen. Die hier verwendete GEATbx macht es einfach, problemspezifisches Wissen an verschiedenen Stellen des Optimierungsprozesses zu verwenden. Dies erleichtert die Bearbeitung praktischer Probleme deutlich.

Ein weiterer interessanter Aspekt der vorgestellten Praxianwendungen ist die Aufgabenteilung bei der Bearbeitung des Problems. Die Modelle und teilweise die Zielfunktionen und Simulationen wurden von den entsprechenden Fachleuten auf diesen Gebieten erstellt. Im Kennenlernen der Systeme und der Aufbereitung der Modelle für die folgenden Optimierungen konnten oftmals Fragen aufgeworfen und diskutiert werden, die manchmal zu einer Verbesserung oder Erweiterung der Modelle und Zielfunktionen führten. In ersten Optimierungen, meist noch ohne Verwendung problemspezifischen Wissens, wurden Probleme der Modelle bzw. Zielfunktionen oder Ungereimtheiten gefunden und im Dialog beseitigt. Durch das bis dahin aufgebaute Wissen und Systemverständnis des Systemtechnikers und Eindrücke von ersten Optimierungen konnte für die Optimierung verwendbares problemspezifisches Wissen aufgearbeitet werden. Dies betrifft vor allem im System vorhandene Beschränkungen, die für die weiteren Optimierungen nicht mehr offen gehalten, sondern in den Suchprozeß eingebunden wurden. Dieser Prozeß setzte sich je nach Länge der Bearbeitungszeit in einigen Stufen fort.

Die Aufstellung einer Zielfunktion für die Optimierung führt zu einer neuen Sichtweise auf das zu lösende Problem, wodurch das bekannte Problem von einer anderen Seite betrachtet wird. Es muß die Frage beantwortet werden, wodurch ein gutes Systemverhalten charakterisiert wird. Die Zielfunktion muß so definiert werden, daß gewünschtes Systemverhalten bevorzugt und unerwünschte Eigenschaften unterdrückt werden. Durch diese Arbeit aus einem neuen Sichtwinkel ist zum Ende der Optimierungen das Problem nicht nur gelöst, sondern es konnten neue Erkenntnisse über das System und Wechselwirkungen gesammelt werden, die auf einem anderen Weg nur schwer gefunden worden wären. Dieser Aspekt

der Optimierung eines Systems wird meist noch nicht gesehen bzw. als nicht so wichtig erkannt.

Neben den hier vorgestellten Beispielen wurden in den vergangenen Jahren eine Reihe weiterer Probleme unter Verwendung Evolutionärer Algorithmen gelöst. Auf diese Arbeiten soll an dieser Stelle nur kurz verwiesen werden:

- Erstellung der Modellstruktur dynamischer Systeme (Blockschatzbild) mit Genetischer Programmierung, Optimierung der Parameter durch stochastische Suchverfahren (TH Darmstadt und Daimler Benz, [MBC95], [MBF96], [PM96]),
- Optimierung der Parameter eines Gleichstrom-Stellers (Chopper) für den Einsatz in Bahnfahrzeugen (Daimler Benz, [Poh98] Abschn. 7.2),
- Ermittlung kürzester und längster Ausführungszeiten von *real time* Softwaremodulen eines Fahrzeugsystems (DaimlerChrysler, [PW99]).

Diese Beispiele und die vorgestellten Anwendungen demonstrieren das breite Spektrum praktischer Probleme, die durch die systemtechnische Anwendung Evolutionärer Algorithmen bearbeitet und gelöst werden können. Zudem zeigen sie die Einsatzbreite und Flexibilität Evolutionärer Algorithmen. Unter Verwendung entsprechend zugeschnittener Werkzeuge stellen Evolutionäre Algorithmen leistungsfähige Optimierungsverfahren für die praktische Anwendung dar. Sie lassen sich direkt und einfach einsetzen und können bei der Lösung vieler praktisch auftretender Probleme helfen.

# 9 Schlußbetrachtungen

Das vorliegende Buch gibt eine umfassende Darstellung Evolutionärer Algorithmen. Am Beginn steht ein Überblick zu Aufbau und Struktur Evolutionärer Algorithmen. Auf diesem aufbauend werden die grundlegenden Bestandteile erläutert. Im Vordergrund steht dabei die Anwendbarkeit der vorgestellten Verfahren und Methoden zur Lösung praktischer Probleme. Dies zeigt sich auch bei den Erweiterungen Evolutionärer Algorithmen sowie der Visualisierung Evolutionärer Algorithmen. Beide Bereiche ermöglichen eine bessere Lösung großer und komplexer Probleme. Alle Darstellungen erfolgen unter dem Gesichtspunkt einer einheitlichen Betrachtung, wodurch der Überblick verbessert und das Verständnis erleichtert wird. Im folgenden Abschnitt werden die einzelnen Bereiche des Buches in ihren Grundaussagen kurz zusammengefaßt. Insbesondere werden die eigenständigen Beiträge dieser Buches hervorgehoben.

Im Ausblick (ab S.261) werden einige Fragestellungen bzw. Bereiche angeprochen, die für die weitere Entwicklung und Erweiterung Evolutionärer Algorithmen von großer Bedeutung sein dürften. Dabei werden insbesondere Parallelen zur natürlichen Umwelt bzw. Evolution gezogen.

## 9.1 Zusammenfassung

In dieser Arbeit werden Evolutionäre Algorithmen als Verfahren zur Optimierung von Systemen vorgestellt. Kapitel 1 spannt einen Bogen von den Vorgängen in der Natur, speziell der natürlichen Evolution, zu deren Anwendung bei der Lösung von Problemen durch Adaption an eine vorgegebene Aufgabenstellung. Durch eine Übertragung der in der Natur zu beobachtenden Vorgänge sowie durch die Abstraktion dieser Vorgänge können Evolutionäre Algorithmen entwickelt werden. Diese Algorithmen erlauben die Lösung technischer Probleme in einer Weise, die ähnlich einfach und gleichzeitig leistungsfähig wie die natürliche Evolution ist.

Die Darstellung der Evolutionären Algorithmen beginnt mit einer Übersicht von Struktur und Aufbau Evolutionärer Algorithmen in Kap. 2. Ausgehend davon werden die Verfahren und Operatoren vorgestellt, aus denen sich ein Evolutionärer Algorithmus zusammensetzt. Dabei werden die grundlegenden Merkmale und Einsatzmöglichkeiten Evolutionärer Algorithmen genannt. Außerdem wird auf Unterschiede zu traditionellen Suchverfahren eingegangen.

Darauf aufbauend beschreibt Kap. 3 die grundlegenden Verfahren und Operatoren Evolutionärer Algorithmus ausführlich und stellt sie in ihrer Wirkungsweise vor. Anders als in vielen Arbeiten zu Evolutionären Algorithmen werden

bei dieser Darstellung nicht einzelne Algorithmen oder Operatoren hervorgehoben oder erläutert. Statt dessen werden ausgehend von einer einfachen Struktur diese durch weitere Verfahren erweitert. Nachfolgend werden die einzelnen Teilbereiche anhand ausgewählter Operatoren und Methoden beschrieben. Durch diese hierarchische Darstellung erfolgt eine einheitliche Beschreibung, die vom allgemeinen Fall ausgeht und diesen mit speziellen Varianten unterstellt. Dies ermöglicht nicht nur eine sinnvollere Beschreibung, sondern erleichtert das Erfassen und Verstehen der einzelnen Komponenten als Teil eines Ganzen und in ihrer Wechselwirkung untereinander.

Evolutionäre Algorithmen arbeiten mit einer Population von Individuen. In erweiterten Evolutionären Algorithmen wird eine Unterteilung der Population vorgenommen, die zur Beschreibung von Populationsmodellen führt. Zu Beginn von Kap. 4 wird ein Überblick über bisherige Klassifikationen von Populationsmodellen gegeben. Für die weitere Arbeit wird die Klassifikation nach der Reichweite der Selektion als am besten geeignet ausgewählt. Diese teilt die Populationsmodelle in globales, regionales und lokales Modell ein. Die nachfolgenden Abschnitte stellen die Modelle ausführlich vor. Insbesondere werden Hinweise für die Wahl und Einstellung der verschiedenen Parameter und ihre Abhängigkeiten untereinander gegeben. Eine allgemeine Beschreibung von Nachbarschaftstopologien, die für das lokale und das regionale Modell angewendet werden kann, wird in Unterabschnitt 4.3.2 vorgenommen. Diese Beschreibung ermöglicht eine Ausdehnung der Nachbarschaftsstrukturen auf beliebige Dimensionen und enthält alle vorher behandelten Topologien als Spezialfälle. Die Vorteile dieser einheitlichen Betrachtung zeigen sich beim Vergleich verschiedener Topologien.

In den Abschn. 4.5 und 4.6 werden neue Erweiterungen des regionalen Modells vorgestellt: die Anwendung verschiedener Strategien und der Einsatz konkurrierender Unterpopulationen. Die Anwendung verschiedener Strategien ermöglicht die gleichzeitige Verwendung verschiedener Parametereinstellungen für jede der Unterpopulationen. Dadurch ergeben sich mehrere Verbesserungen. Einerseits ist die Anwendung verschiedener Strategien in der Durchführung schneller und einfacher als mehrere unabhängige Experimente mit unterschiedlichen Parametersätzen. Die Ergebnisse können in einer kompakten und überschaubaren Weise dargeboten werden, wodurch die Bewertung des Erfolgs der Strategien direkt und einfach möglich ist. Erfolglose und erfolgreiche Strategien können voneinander getrennt werden. Der während eines Laufs unterschiedliche Erfolg der Strategien kann direkt und in Wechselwirkung mit anderen Strategien bewertet werden. Andererseits können sich die Strategien innerhalb eines Laufs untereinander ergänzen, wodurch es zu besseren Ergebnissen kommt, als bei getrennt voneinander verwendeten Strategien. In Abschn. 4.5 werden neben der Berechnung des Erfolgs der einzelnen Strategien durch die Ordnung der Unterpopulationen insbesondere Methoden zum visuellen Vergleich der Strategien und zur Auswertung des Erfolgs der einzelnen Strategien vorgestellt und in der Anwendung gezeigt.

Noch einen Schritt weiter geht der Einsatz konkurrierender Unterpopulationen in Abschn. 4.6. Dabei werden nicht nur verschiedene Strategien gleichzeitig angewendet, sondern es findet ebenfalls eine Verteilung der zur Verfügung stehenden Ressourcen entsprechend des Erfolgs der einzelnen Strategien statt. Erfolg-

reiche Strategien erhalten mehr Ressourcen zur Verfügung gestellt als erfolglose Strategien. Dadurch kommt es zu einer dynamischen Verteilung der Ressourcen mit einer Bevorzugung der erfolgreichen Strategien. Für die Verteilung der Ressourcen werden Analogien zur Fitneßzuweisung gezogen und diese Verfahren entsprechend für die Aufteilung der Ressourcen angewendet. Durch die Verteilung der Ressourcen kommt es zu einer Anpassung der Strategieparameter während eines Laufs. Diese Anpassung muß nicht von außen oder vom Benutzer vorgegeben werden, sondern erfolgt in Abhängigkeit des Verlaufs der Optimierung. Es können damit nicht nur verschiedene Strategien gleichzeitig angewendet werden, sondern die erfolgreichen Strategien erhalten einen großen Teil der Ressourcen zur Verfügung gestellt, und die erfolglosen Strategien verbrauchen nur einen kleinen Teil der Ressourcen. Ein „Aussterben“ der zu einer Zeit erfolglosen Strategien wird verhindert. Damit wird eine Ergänzung mehrerer Strategien untereinander bei effektiver Verteilung der Ressourcen ermöglicht.

Die Anwendung verschiedener Strategien und der Einsatz konkurrierender Unterpopulationen sind ein weiterer Schritt zur Entwicklung leistungsfähiger Evolutionärer Algorithmen für die Lösung großer und schwieriger Probleme. Die Vorteile dieser Erweiterungen des regionalen Populationsmodells zeigen sich besonders bei der Bearbeitung praktischer Probleme, bei denen es auf den schnellen Test verschiedener Strategien und die Auswertung dieser Läufe bei einer gleichzeitig effektiven Verteilung der zur Verfügung stehenden Ressourcen ankommt.

Nach der Beschreibung Evolutionärer Algorithmen werden in Kap. 5 Methoden zur visuellen Auswertung Evolutionärer Algorithmen vorgestellt. Die bei der Arbeit Evolutionärer Algorithmen anfallenden Daten sind relativ einfach strukturiert. Durch die große Menge an Daten müssen diese aber einer Bearbeitung unterzogen werden, um sie für den Anwender verständlich und benutzbar zu machen. Die dargestellten Methoden und Verfahren ermöglichen eine Visualisierung des Verlaufs und der Ergebnisse der Arbeit Evolutionärer Algorithmen. Die Visualisierung kann für unterschiedliche zeitliche Bereiche (eine, mehrere, alle Generationen eines Laufs), für unterschiedliche Mengen an Daten (einzelne, mehrere, alle Individuen einer Population), für direkt zur Verfügung stehende Werte oder für abgeleitete Werte erfolgen. Zusätzlich werden Methoden zur hochdimensionalen Visualisierung und zur Visualisierung von Eigenschaften der Zielfunktion vorgestellt, die bei der Analyse eines Problems hilfreich sind.

Mit den vorgestellten Verfahren zur Visualisierung stehen leistungsfähige Methoden zur Verfügung, die einen schnellen und effektiven Überblick über die verschiedensten Aspekte des Verlaufs und der Ergebnisse Evolutionärer Algorithmen ermöglichen. Durch den Einsatz der Visualisierungsverfahren können Einsichten in die Arbeit Evolutionärer Algorithmen gewonnen werden, die auf anderem Wege nicht möglich sind bzw. in den anfallenden Datensätzen verborgen bleiben würden. Die präsentierten umfangreichen Beispiele zeigen den Einsatz der Verfahren an realen und Testproblemen und machen auf die aus den jeweiligen Darstellungen ablesbaren Informationen aufmerksam.

Alle in den Kapiteln 3, 4 und 5 vorgestellten und beschriebenen Verfahren, Operatoren und Methoden sind in der GENETIC AND EVOLUTIONARY ALGORITHM TOOLBOX FOR USE WITH MATLAB [GEATbx] implementiert. Der grundlegende Aufbau dieser Toolbox wird in Kap. 6 vorgestellt. Die GEATbx stellt die umfangreichste Implementierung Evolutionärer Algorithmen unter dem Programmiersy-

stem MATLAB dar, die zur Zeit verfügbar ist. Alle in dieser Arbeit vorgestellten Beispiele und Grafiken wurden mit dieser Toolbox berechnet und erzeugt. Die in Kap. 8 besprochenen Anwendungen wurden mit dieser Toolbox gelöst.

Mit der dem Buch beiliegenden CD erhält der Leser des Buches Version 1.95 der GEATbx. Damit ist eine sofortige Anwendung der im Buch besprochenen Verfahren und Methoden auf eigene Probleme möglich. Die Toolbox ist bereits bei mehreren industriellen Anwendern und internationalen Forschergruppen im Einsatz und hat sich dort in den verschiedensten Anwendungsbereichen bewährt. Die plattformunabhängige Programmierung und Dokumentation erlaubt einen direkten Einsatz auf allen Rechnersystemen, auf denen MATLAB verfügbar ist.

Kapitel 8 beschreibt der Lösung mehrerer Anwendungen, die als Beispiele von in den letzten Jahren gelösten Problemen exemplarisch ausgewählt wurden. Den Anfang bildet die Darstellung des prinzipiellen Vorgehens bei der Lösung neuer Optimierungsprobleme. Damit wird insbesondere dem Einsteiger ein Leitfaden in die Hand gegeben, um schneller und sicherer zu guten Ergebnissen bei der Optimierung von Problemen zu gelangen. Bei den vorgestellten Anwendungen wird schrittweise die Komplexität erhöht. Ausgangspunkt sind einfache mehrdimensionale Funktionen in Abschn. 8.2. Diese stellen Testfunktionen dar, die in vielen Berichten als Beispiele verwendet werden. Zwei häufig zu lösende Aufgaben sind die Parameteridentifikation eines Modells (Dieselmotormodell in Abschn. 8.3) und die Parameterdimensionierung eines Reglers (Querlenkung eines Fahrzeugs in Abschn. 8.4). Die umfangreichste Anwendung stellt die Optimierung des Gewächshausklimas in Abschn. 8.5 dar. Dieses Problem beinhaltet in einem Teilmodell ein biologisches System, das in seiner Interaktion mit den äußeren Zuständen ein äußerst komplexes System bildet. Erst durch eine Reduktion der Komplexität des biologischen Systems, wie es in dem hier verwendeten Modell stattgefunden hat, ist eine Bearbeitung und Optimierung des Gesamtsystems mit heutigen Mitteln möglich.

Die Anwendungen Dieselmotormodell, Fahrzeuglenkung und Gewächshausklima stellen komplexe Probleme dar. Besonderes Augenmerk wurde auf die Einbringung von problemspezifischem Wissen in den Optimierungsprozeß gelegt. Dies ist bei Evolutionären Algorithmen an verschiedenen Punkten und relativ einfach möglich. Durch dieses problemspezifische Wissen kann die Suche nach Lösungen deutlich beschleunigt bzw. die Qualität der erreichten Lösungen verbessert werden.

Die hier dargestellten Anwendungen geben einen kleinen Einblick in das breite Spektrum praktischer Probleme, die durch die systemtechnische Anwendung Evolutionärer Algorithmen bearbeitet und gelöst werden können. Unter Verwendung entsprechend zugeschnittener Werkzeuge stellen Evolutionäre Algorithmen leistungsfähige Optimierungsverfahren mit unterschiedlichsten Eigenschaften für die praktische Anwendung dar. Sie können direkt und einfach eingesetzt werden und bei der Lösung einer großen Klasse von Problemen helfen. Mit Evolutionären Algorithmen konnten in vielen Anwendungen bessere Lösungen gefunden werden, als dies mit klassischen Optimierungsverfahren möglich war.

In Abschn. A.1 wird eine Übersicht über die historische Entwicklung Evolutionärer Algorithmen sowie die frühen Arbeiten auf diesem Gebiet gegeben. Dabei wird kurz auf die drei Schulen Evolutionärer Algorithmen eingegangen: Evolutionäre Programmierung, Evolutionsstrategien und Genetische Algorith-

men. Aus heutiger Sicht sind die früheren Unterschiede zwischen diesen Richtungen kaum noch vorhanden. Statt auf diese Unterschiede zu verweisen, sollten die Gemeinsamkeiten sowie die Anwendbarkeit der Verfahren und Operatoren auf verschiedene Klassen von Problemen in den Vordergrund gestellt werden. Dieser Gedanke war eine der Grundlagen bei der Anfertigung des vorliegenden Buches.

Ein ausführliches Literaturverzeichnis zu allen thematischen Bereichen des Buches bietet Zugang zu weiteren Arbeiten. Das Glossar erläutert viele im Buch verwendete Begriffe. Mit dem Sachverzeichnis ist ein schneller Zugriff auf alle Bereiche des Buches über die entsprechenden Stichworte gegeben.

## 9.2 Ausblick

Die heute verwendeten Evolutionären Algorithmen, eingeschlossen die in diesem Buch vorgestellten, sind im Vergleich zu den Vorgängen in der Natur immer noch stark vereinfacht. Dies betrifft verschiedene Aspekte:

- die Art der Kodierung der genetischen Information,
- den Zusammenhang zwischen Genotyp (genauer: Idiotyp) und Phänotyp; wie wird die genetische Information phänotypisch ausgeprägt,
- die Interaktion zwischen den Individuen in einer Population,
- die Veränderungen der Umwelt im Verlauf der Zeit.

Als Ausblick sollen einige dieser Aspekte untersucht und auf ihre Relevanz für die weitere Entwicklung Evolutionärer Algorithmen hin bewertet werden. Dabei muß beachtet werden, daß Merkmale, Eigenschaften und Abläufe, die sich an einer Stelle entwickelt haben, meist mit denen auf einer oder mehreren anderen Ebenen in Zusammenhang stehen, auch wenn dies nicht für jedes der Beispiele extra betont wird.

### ***Kodierung der genetischen Information***

Die Art der Kodierung der genetischen Information ist eine Frage, die im Bereich der Evolutionären Algorithmen mit jedem neuen Problem aufs neue zur Diskussion steht. In der Natur hat sich eine Variante etabliert. Die genetische Erbinformation liegt in Chromosomensätzen vor. Jeder Chromosomensatz besteht aus einer (für die jeweilige Art charakteristischen) Anzahl von Chromosomen. Die Chromosomen sind Träger der Gene. Die Gene wiederum lassen sich Teilen bzw. Abschnitten der DNS zuordnen. Die DNS besteht von den bestimmenden Komponenten her aus einer Sequenz von 4 verschiedenen Basen, wobei immer drei aufeinanderfolgende Basen eine Aminosäure kodieren. Von den theoretisch möglichen 64 Aminosäuren werden allerdings nur 20 verschiedene Aminosäuren benutzt. Mit diesen in der Grundlage einfachen Mitteln lassen sich die verschiedensten Informationen kodieren und speichern. Stark vereinfacht könnte man sagen, daß die Information durch Sequenzen der Zahlen 1, 2, 3 und 4 gespeichert wird. Oder, wenn man die Aminosäuren als Grundlage verwendet, aus einer Sequenz der ganzzahligen Ziffern 1 bis 20. Komplexere Strukturen, wie z.B. ein

größerer Bereich der zur Kodierung verwendeten Symbole oder eine komplexere Struktur als die Sequenz haben sich im Prinzip nicht herausgebildet bzw. durchgesetzt. Die Grundlage der Kodierung und Speicherung der Erbinformation bleiben einfachste Varianten; komplexere Strukturen entstehen erst durch die Interpretation bzw. Umsetzung des einfachen genetischen Kodes.

Damit in direktem Zusammenhang stehen die Fragen, wie der genetische Kode interpretiert wird bzw. wie die genetische Information in phänotypische Merkmale umgesetzt wird. Wie oben schon angedeutet, hat sich dafür in der Natur ein teilweise mehrstufiges System herausgebildet. An der Ausbildung eines phänotypischen Merkmals sind meist mehrere Gene beteiligt und die Wechselwirkungen dabei variieren auf den verschiedensten Ebenen. Deshalb ist es nicht so einfach, einem Merkmal eine bestimmte Gensequenz bzw. einen Abschnitt des genetischen Kodes zuzuordnen. Rückschlüsse von einer Veränderung der genetischen Information auf die folgende Veränderung von phänotypischen Merkmalen sind bisher nur in wenigen Fällen möglich (z.B., wenn nur ein Gen die Ausprägung eines Merkmals bestimmt und/bzw. nur wenige Zustände dieses Merkmals möglich sind).

Welche Überlegungen ergeben sich für die Anwendung bei Evolutionären Algorithmen? Mit einer einfachen Kodierung lassen sich die verschiedensten Informationen darstellen. Die Komplexität liegt dann in der Interpretation der einfachen Information und ihrer Ausbildung in entsprechende Merkmale. Der Zusammenhang zwischen genetischer Information und den Merkmalen ist um so schwerer zu interpretieren, je komplexer die Interaktionen der genetischen Information zur Ausbildung der Merkmale werden.

Im Sinne einer einfachen Kodierung ist die binäre Kodierung nicht zu übertreffen. Es gibt nur 2 Zustände jeder einzelnen Einheit. Dies sind weniger, als die 4 Zustände (4 Basen) in der Natur. Der Ansatz, alle Informationen binär zu kodieren, wurde in den frühen Arbeiten der Genetischen Algorithmen propagiert und wird auch heute noch an vielen Stellen verwendet. Der Vorteil ist, daß dieselben Operatoren für jedes Problem verwendet werden können, egal wie komplex die Umwandlung der genetischen Information in Merkmale ist. Der große Nachteil ist gerade dieser sehr komplex werdende Zusammenhang bei großen und komplizierten Problemen.

Die andere Variante stellen Evolutionäre Algorithmen dar, bei denen die genetische Information direkt im Format der Merkmale gespeichert wird. Allerdings müssen für diese merkmalsspezifischen Formate entsprechende Operatoren entwickelt werden. Diese Operatoren lassen sich dann nur für die Manipulation dieses einen Formats einsetzen. Beispiele dafür sind die in dieser Arbeit vorgestellten Rekombinations- und Mutationsoperatoren für reelle Variablen und die Operatoren, die auf Baumstrukturen anwendbar sind, wie sie im Bereich der Genetischen Programmierung verwendet werden.

Zwischen diesen beiden Extremen, Speicherung der genetischen Information in binären Variablen und Speicherung der genetischen Information im Format der Merkmale, gibt es noch eine Reihe von Zwischenstufen, bei denen eine teilweise Umsetzung zwischen den Formaten stattfindet. Diese Zwischenstufen werden im Moment bei Evolutionären Algorithmen nur selten eingesetzt.

Welche Variante der Kodierung im Endeffekt bei der Bearbeitung eines Problems angewendet wird, hängt sehr stark von der Art des Problems und der Ziel-

stellung der Bearbeitung ab. Bei der Optimierung von Systemen nach einer Zielfunktion hat sich die Anwendung merkmalsspezifischer Formate für die Kodierung der Informationen an vielen Stellen durchgesetzt. Die stärkere Spezialisierung und Fokussierung dieser Kodierung und der korrespondierenden Operatoren sowie das indirekte Einbringen von problemspezifischem Wissen über die Struktur und Interaktion der Variablen konnten bessere Ergebnisse (Ergebnisse in kürzerer Zeit oder mit einem geringeren Aufwand) erreichen. Beim Vergleich der Verwendung einer binären Kodierung und der direkten Verwendung der entsprechenden reellen Variablen wurden für die reelle Kodierung und deren Operatoren bessere Ergebnisse berichtet (u.a. [Mic94] und [Dav91]).

Wenn es dagegen um die Erzeugung einer großen Variabilität und möglichst vieler Formen geht, so ergeben sich mit einer einfachen Kodierung, wie der binären Kodierung, die meisten Möglichkeiten, die zudem mit einfachen und universell anwendbaren Operatoren bearbeitet werden können. Und genau diese Variante scheint sich in der Natur durchgesetzt zu haben. Dort geht es ja auch nicht um die „Lösung einer einfachen Optimierungsaufgabe“.

### **Diploidie und Polyploidie**

In direktem Zusammenhang mit der eben besprochenen Kodierung der genetischen Information steht die Frage der Ploidie der Information. Bisher wurde davon ausgegangen, daß die genetische Information in einfacher Form vorliegt, d.h. in einfachen Chromosomensätzen (haploid). Diploidie kennzeichnet den Zustand, daß die genetische Information in doppelter Ausführung vorliegt, bei Polyploidie ist die genetische Information noch öfter vorhanden (triploid, tetraploid, usw.).

In der Natur kommen alle diese Formen vor. Die meisten niederen Lebewesen sind haploid, die genetische Information liegt einfach vor. Aber so gut wie alle höheren Lebewesen haben einen diploiden oder polyploiden Chromosomensatz<sup>1</sup>. Je komplexer die Organismen sind, um so komplexer ist die zugrundeliegende Struktur – und dies bedeutet insbesondere eine diploide oder polyploide Struktur.

Die bei diploiden Organismen in doppelter Ausführung vorliegende genetische Information ist in beiden Strängen nicht vollständig identisch. Die Frage ist, welcher Teil der doppelten Information zur Ausprägung gelangt. Sind die Bausteine für gleiche Stellen identisch (*homozygote* Allele), setzt sich diese Information durch. Bei unterschiedlicher Information auf den entsprechenden (homologen) Stellen eines Chromosoms (*heterozygote* Allele) wird durch einen Dominanzmechanismus entschieden, welche Information für die Ausprägung des Merkmals verwendet wird. Das sich durchsetzende Allel wird als dominant bezeichnet, das andere als rezessiv. Mit anderen Worten, das dominante Allel unterdrückt die Manifestierung des rezessiven Allels. Dieser Mechanismus muß nicht eindeutig

<sup>1</sup> Bei höheren Pflanzen wurde z.B. beobachtet, daß bei zunehmend extremeren Umweltbedingungen der Anteil der polyploiden Arten zunimmt (s. [Hag91] und [Ode97]). Je weiter man nach Norden in die Nähe der Arktis kommt oder in der Tundra ist, um so mehr Arten sind polyploid. Der Anteil liegt bei über 50%. Ähnliches wurde bei einer Annäherung an Wüstengebiete beobachtet.

sein, es kann auch eine „Mischung“ zwischen den Informationen der beiden Allele stattfinden.

Daraus ergibt sich die Frage, welche Vorteile diese Vervielfachung der genetischen Information bringt, die auf den ersten Blick die Kodierung der genetischen Information nur noch komplizierter und komplexer macht.

Durch die Vervielfachung wird einmal mehr Information gespeichert. Die Erweiterung der Speicherkapazität ist aber nicht der Hauptaspekt (dies könnte auch durch eine Verlängerung der Information erfolgen). Statt dessen muß gesehen werden, daß die Information, die nicht zur Ausprägung kommt, damit auch nicht der Selektion unterliegt. Es wird ein Speicher von alternativen Informationen gebildet. Erst durch eine Veränderung des Dominanzmechanismus kann diese Information zur Ausprägung kommen und unterliegt dann auch der Selektion. Die zusätzliche Information stellt ein Reservoir dar.

Unter anderen Umständen bzw. in der Vergangenheit kann bzw. konnte diese Information einmal von Vorteil sein (verteilter Langzeitspeicher). Dies bedeutet die Speicherung mehrerer Lösungen, wobei zu einer Zeit immer nur eine dieser Lösungen zur Ausprägung gelangt. Und da die nicht ausgeprägten Lösungen nicht direkt der Selektion unterliegen, werden sie im Speicher vor einer Zerstörung bewahrt. Von Zeit zu Zeit kann an die alten Lösungen „erinnert“, d.h. diese können unter den neuen Umständen getestet werden.

Fast alle heute verwendeten Evolutionären Algorithmen verwenden eine haploide Darstellung der Information. Die Anzahl der Arbeiten, die eine diploide oder polyploide Implementierung vorstellen, läßt sich noch gut überschauen. Auf einige dieser Arbeiten soll im folgenden kurz eingegangen werden.

Einige der frühen Arbeiten zu Evolutionären Algorithmen benutzten schon eine diploide Darstellung der Informationen. Daran ist unter anderem zu erkennen, wie stark sich frühe Arbeiten am biologischen Vorbild orientieren. Dies betrifft die Arbeiten von BAGLEY [Bag67] ROSENBERG [Ros67], HOLLSTIEN [Hol71] und BRINDLE [Bri81]. Auf diese Arbeiten, auch den Aspekt der diploiden Darstellung der Information, wird in Unterabschn. A.1.1, ab S.267, eingegangen. Zusammenfassend kann gesagt werden, daß die Ergebnisse dieser Arbeiten nicht sehr ermutigend sind. Es konnte in den Experimenten kein direkter Vorteil der Verwendung eines diploiden Algorithmus gegenüber einem haploiden Algorithmus gefunden werden. Dies lag vor allem daran, daß mit stationären Zielfunktionen gearbeitet wurde. Der durch die diploide Darstellung vorhandene Speicher vergangener Lösungen kann mit diesen Zielfunktionen keine Vorteile erbringen. Im Gegenteil, durch die diploide Darstellung der Information wird eine komplexere Interpretation der genetischen Information in den Algorithmus eingeführt, die zu einer langsameren Entwicklung des Systems führt.

Das von HOLLSTIEN [Hol71] eingeführte dreiallelige (*trallelic*) Schema, bei dem die binäre genetische Information und die Dominanzinformation an jeder Position direkt kodiert wurden, war über viele Jahre hinweg Grundlage weiterer Arbeiten, u.a. auch von HOLLAND in [Hol75].

GOLDBERG und SMITH veröffentlichten in [GS87], [Smi87], [Smi88] und [Gol89] Vergleiche zwischen haploiden und diploiden Genetischen Algorithmen. Die diploiden Algorithmen unterschieden sich in der Art des Dominanzschemas (fixiertes Dominanzschema bzw. das Schema von HOLLSTIEN und HOLLAND). Als Testfunktion verwendeten sie eine nicht-stationäre Funktion (*blind non-*

(*stationary knapsack*). Diese Funktion oszillierte mit einer bestimmten Frequenz. Mit den von ihnen verwendeten Parametern des Genetischen Algorithmus war der haploide Algorithmus nicht in der Lage, den Oszillationen zu folgen, der diploide Algorithmus mit fixiertem Dominanzschema konnte den Oszillationen etwas folgen. Aber nur bei Verwendung des dreialleligen Schemas konnte eine schnelle und volle Adaption erreicht werden.

In [NW95] wurde gezeigt, daß die Ergebnisse von GOLDBERG und SMITH in [GS87] nur durch die Wahl ungünstiger Parameter für das haploide Schema zu stande kamen. Das dreiallelige Schema hatte keine Vorteile gegenüber dem haploiden Schema, wenn andere Oszillationsfrequenzen der Zielfunktion bzw. höhere Mutationsraten verwendet wurden. In [NW95] wurde ein neues diploides Schema zur Kodierung vorgeschlagen. Außerdem wurde ein drastischer Dominanzwechsel durchgeführt, wenn der Zielfunktionswert eines Individuums sich plötzlich stark verschlechterte. Das neue diploide Schema lässt sich auch auf Variablen anwenden, die mehr als zwei Zustände eines Merkmals haben.

RYAN berichtet in [Rya96] über ein neues diploides Schema, das sich auf Variablen mit 2 und mehr Zuständen ausdehnen lässt und keinen Bias (Bevorzugung) in Richtung eines der Zustände hat. Für binäre Variablen nannte er es '*Degree of Oneness*', für Variablen mit mehr als 2 Zuständen '*Degree of Nness*'. Zusätzlich untersuchte er die Verwendung mehrerer diploider Variablen (mehrerer Gene) zur Kodierung eines Merkmals (*polygenic inheritance*). In seinen Untersuchungen mit einer einfachen nicht-stationären Funktion erzielte diese Variante die besten Ergebnisse.

Ein in verschiedenen Arbeiten immer wieder betonter Aspekt der Anwendung eines diploiden Schemas ist die Erhaltung oder Erhöhung der Verschiedenartigkeit der Individuen der Population (*population diversity*) (u.a. [CCR96]). In [YA94] wurde ein *pseudo-Meiosis GA* vorgestellt, der zwei unterschiedliche Chromosomen miteinander zur Erstellung eines Phänotyps kombiniert. Dieser Algorithmus wurde auf ein nicht-stationäres *travelling salesman* Problem angewendet. In Vergleichen mit einem einfachen Genetischen Algorithmus war der *pseudo-Meiosis GA* besser. Es wurden nicht nur kürzere Touren gefunden, sondern nach einer Änderung der Zielfunktion konnte die Adaption schneller durchgeführt werden. Die Verschiedenartigkeit in der Population war während des Laufs, insbesondere zum Ende hin, deutlich höher als mit dem einfachen Genetischen Algorithmus.

Lassen sich aus diesen Arbeiten Schlußfolgerungen ziehen, ob, wann und in welchen Bereichen die Anwendung der Diploidie Vorteile für Evolutionäre Algorithmen bringt? Zwei Bereiche wurden öfters erwähnt:

- Erhaltung der Verschiedenartigkeit der Population und
- schnelle Adaption an Veränderungen der Zielfunktion.

Die Erhaltung der Verschiedenartigkeit der Population ist eine immer wieder aufgestellte Forderung und wird auch durch verschiedene andere Methoden gefördert. Hingewiesen sei nur auf die Anwendung verschiedener Populationsmodelle, die in Kap. 4, ab S.73, vorgestellt wurden.

Interessant wird die Erhaltung der Verschiedenartigkeit der Population durch Diploidie allerdings im Zusammenhang mit einer sich verändernden Zielfunktion. Die Verschiedenartigkeit beinhaltet dann nicht nur verschiedene Varianten,

sondern insbesondere Varianten, die unter anderen Bedingungen der Zielfunktion schon einmal erfolgreich waren. Durch Diploidie wird in der Population ein Speicher vergangener Lösungen bewahrt, der von Zeit zu Zeit wieder getestet wird. Diese Information muß nicht erst durch Mutation neu erzeugt, sondern nur neu kombiniert bzw. getestet werden. Allerdings sind die meisten heute verwendeten Probleme bzw. Zielfunktionen stationäre Funktionen, ein Speicher vergangener Lösungen bringt damit nur in wenigen Fällen einen Vorteil.

Ein weiterer Aspekt der vervielfachten Information ist die erhöhte Stabilität der Information. Eine Mutation in einem haploiden Chromosom kommt zur Ausprägung. Eine Mutation in einem diploiden Chromosom muß nicht zur Ausprägung gelangen, insbesondere, wenn rezessive Allele verändert werden. Diese Veränderungen können im Verborgenen bleiben und beeinflussen nicht den Phänotyp. In der Natur ist diese erhöhte Stabilität ein wichtiger Aspekt, da gute Informationen nicht durch die ständig stattfindenden Mutationen wieder zerstört werden sollen.

In den künstlichen Welten heutiger Evolutionärer Algorithmen, die beschränkte Populationen, stationäre Zielfunktionen und eine geringe Anzahl von Generationen benutzen, ist kein Bereich erkennbar, in dem ein diploides oder polyploides System einen Vorteil gegenüber einem vernünftig entworfenen haploiden System bringt. Mit der weiteren Entwicklung Evolutionärer Algorithmen werden aber schrittweise immer größere Probleme bearbeitet. Bei diesen können mehrere Zielfunktionen gleichzeitig zum Einsatz kommen; die Entwicklung des Systems verläuft über eine wesentlich längere Zeit und die Populationen sind deutlich größer. In diesen neuen Umgebungen sollte der Einsatz diploider Systeme in Betracht gezogen werden, da dann die Vorteile dieser Systeme gegenüber haploiden Systemen besser zur Wirkung kommen können.

# Anhang

## A.1 Historische Entwicklung Evolutionärer Algorithmen

Der Evolutionsprozeß ist ein in der Natur ablaufender immerwährender Optimierungsprozeß, der zu einer schrittweisen Anpassung der Individuen an die sich verändernde Umwelt führt. In den 50er Jahren und mit Beginn der 60er Jahre wurden Teile dieses bisher nur im biologischen Kontext untersuchten Evolutionsprozesses abstrahiert und für die Lösung von praktischen Aufgabenstellungen verwendet. Diese ersten Arbeiten mit und an Evolutionären Algorithmen begannen weltweit etwa gleichzeitig an verschiedenen Stellen. Aus diesen frühen Arbeiten entwickelten sich unabhängig voneinander drei Schulen Evolutionärer Algorithmen:

- Evolutionäre Programmierung (*Evolutionary Programming*),
- Evolutionsstrategien (*Evolution Strategies*) und
- Genetische Algorithmen (*Genetic Algorithms*).

Alle diese Methoden verwenden die Grundelemente des Evolutionsprozesses: Selektion und Variation. Sie unterscheiden sich aber in der Ausprägung dieser Elemente.

In diesem Kapitel wird ein Überblick über die ersten Arbeiten sowie die Entwicklung der drei Schulen Evolutionärer Algorithmen gegeben. Im ersten Abschnitt werden die frühen Arbeiten vorgestellt, die heute als die Vorfürher Evolutionärer Algorithmen angesehen werden. Die nachfolgenden drei Abschnitte stellen jeweils einige der frühen Arbeiten der drei verschiedenen Schulen Evolutionärer Algorithmen vor: Evolutionäre Programmierung, Evolutionsstrategien und Genetische Algorithmen. Zum Abschluß wird im fünften Abschnitt auf das Zusammenfließen und Verschmelzen der Methoden der drei Schulen seit Beginn der 90er Jahre eingegangen. Seitdem hat sich der Begriff Evolutionäre Algorithmen (*evolutionary algorithms*) als Bezeichnung für die Algorithmen der verschiedenen Schulen durchgesetzt.

### A.1.1 Erste Arbeiten zu Evolutionären Algorithmen

Viele der frühen Arbeiten zur Verwendung von Prinzipien der Evolution in Suchalgorithmen waren biologisch motiviert. Das Ziel dieser Untersuchungen war das Verständnis natürlicher Phänomene. Beispiele dafür sind in [Fra62],

[Ros67] und [Wei70] zu finden. Eine weitere Gruppe von Arbeiten untersuchte verschiedene Evolutionäre Algorithmen und wendete diese auf Testbeispiele an, wie z.B. [Bre62], [Bag67] und [Hol71]. Ein Teil der frühen Arbeiten beschäftigte sich bereits mit der Lösung praktischer Probleme ([Box57] und [Cav70]; [FOW66]: s. Unterabschn. A.1.2; [Sch68] und [Rec73]: s. Unterabschn. A.1.3). Einen frühen Versuch der Generierung von Computerprogrammen stellten die Arbeiten in [Fri58] und [FDN59] dar. In diesem Abschnitt werden die frühen Arbeiten kurz vorgestellt und auf ihre wesentlichen Bestandteile eingegangen.

FRASER ([Fra62]) untersuchte die Abhängigkeiten zwischen einzelnen Genen und die Auswirkungen auf den Phänotyp. Dabei wurde eine Population von Individuen verwendet, die als Bitstring kodiert waren. Die Individuen zur Produktion der folgenden Generation (Eltern) wurden in Abhängigkeit vom Phänotyp (Zielfunktionswert) aus dieser Population ausgewählt. Dieser Phänotyp mußte in einem bestimmten Intervall liegen. Mittels dieser Simulationen ermittelte FRASER den Anteil akzeptabler Individuen in folgenden Generationen. Bei FRASER findet sich allerdings kein Hinweis, daß sein natürliches Suchverfahren auch auf technische Probleme anwendbar sei.

ROSENBERG ([Ros67]) veröffentlichte seine Dissertation 1967. Er betonte mehr den biologischen Aspekt und die Simulationen, so daß die Dinge, die heute als Evolutionäre Algorithmen gesehen werden, weniger Beachtung fanden. ROSENBERG verwendete eine Population einzelliger Organismen, die eine einfache Biochemie und die klassische genetische Struktur (1 Gen, 1 Enzym) besaßen. Außerdem war jedes Individuum als Chromosomenpaar (diploide Variablenstruktur) kodiert. Paarung und Selektion wurden entsprechend einer Fitneßfunktion durchgeführt. Obwohl ROSENBERG mehrere Fitneßwerte berechnete, wurde immer nur einer dieser Werte für die Selektion verwendet. Weiterhin benutzte ROSENBERG einen adaptiven Rekombinationsoperator. Eine Betrachtung der Effekte der diploiden Kodierung (Dominanz, Rezessivität) als separate Effekte fand aber nicht statt. Dominanz war nur ein Merkmal in Abhängigkeit von internen Variablen, das nachfolgend die Ausbildung eines anderen Merkmals kontrollierte.

WEINBERG ([Wei70]) benutzte genetische Algorithmen, um die Parameter eines biologischen Simulationsmodells (Simulation von *E. coli* – Zellen) zu optimieren. Sein Modell hatte 15 Parameter, die in einem großen Bereich veränderbar waren. Der genetische Algorithmus arbeitete direkt mit den reellen Variablen. Rekombination und Inversion kamen zur Anwendung. Zur Mutation wurde ein Operator verwendet, den er als *directed mutation* bezeichnete. Außerdem beschreibt WEINBERG die Verwendung eines zweistufigen genetischen Algorithmus. Die untere Stufe (*adaptive genetic program*) dient der Optimierung der Parameter des Problems, die obere Stufe (*non-adaptive genetic program*) optimiert die Parameter des genetischen Algorithmus auf der unteren Stufe. Die Verbesserung in der Güte des besten Individuums auf der unteren Stufe diente jeweils als Güte eines Parametersatzes (Rekombinations-, Inversions- und Mutationswahrscheinlichkeit) auf der oberen Stufe. Dadurch sollte eine schrittweise Adaption der Parameter des genetischen Algorithmus der unteren Stufe erreicht werden. WEINBERG präsentierte in seiner Arbeit allerdings keine Ergebnisse mit diesem mehrstufigen genetischen Algorithmus.

BREMERMANN ([Bre62]) berichtete in seiner Arbeit von der Generierung von Populationen. Die Populationen bestanden aus binär bzw. reell kodierten Individuen. Durch Selektion wurden Eltern ausgewählt. Aus diesen wurden durch Mutation neue Individuen gebildet, wobei immer eine diskrete Mutation verwendet wurde (starke Orientierung an der diskreten Darstellung der genetischen Information in der Natur). Von BREMERMANN wurde auch die Verwendung eines Rekombinationsoperators vorgeschlagen. Unter anderem schlug er vor, mehrere Nachkommen pro Elter zu generieren und die besten dieser Nachkommen durch eine Rekombination auf ein einzelnes Individuum zu reduzieren. BREMERMANN verwendete für seine Experimente einfache lineare und konvexe Funktionen. Die experimentellen Ergebnisse waren nicht sehr erfolgreich und konnten nur durch die Einbeziehung speziellen Wissens verbessert werden.

In seiner Dissertation erwähnte BAGLEY ([Bag67]) als Erster den Begriff „Genetischer Algorithmus“ und veröffentlichte auch dessen erste Anwendung. BAGLEY konstruierte einen genetischen Algorithmus, der alle Bestandteile heute verwendeter Algorithmen beinhaltete: Selektion, Reproduktion, Rekombination und Mutation. Dabei verwendete er eine diploide und nicht-binäre Repräsentation für seine Individuen. Die diploiden Chromosomen wurden durch variable Dominanzparameter, die Teil der Repräsentation waren, in einen Phänotyp umgewandelt. Dies erlaubte die Anpassung des Dominanzschemas über mehrere Generationen. Da keine Mutation in den Dominanzparametern zugelassen wurde, kam es aber zu einer schnellen Fixierung dieser Werte. Es fand kein Vergleich mit einer haploiden Repräsentation statt. In der Selektion benutzte BAGLEY eine variable Fitneßzuweisung. Zu Beginn der Berechnungen wurde der Selektionsdruck verringert, wodurch verhindert werden sollte, daß wenige gute Individuen sehr schnell die Population dominieren. Im Verlauf der Berechnungen wurde der Selektionsdruck erhöht, um den Wettbewerb zwischen den immer ähnlicher werdenden Individuen zu erhalten. BAGLEY wendete den GA in der Spieltheorie auf ein 3x3 Damespiel an und verglich die Ergebnisse mit anderen, damals gebräuchlichen Ansätzen. Der GA funktionierte dabei gut über eine große Breite von Problemen.

HOLLSTIEN ([Hol71]) untersuchte in seiner Dissertation die Anwendung genetischer Algorithmen zur Optimierung einer Anzahl von Funktionen, die je von zwei Variablen abhängig waren. Das spezielle Augenmerk galt der Untersuchung verschiedener Selektionsverfahren sowie dem Test verschiedener Schemata, welche Individuen miteinander Nachkommen erzeugen können (*mating schemes*). Am Ende waren fünf Selektionsverfahren und acht *mating schemes* an 14 Testfunktionen zu untersuchen. HOLLSTIEN verwendete Populationen mit 16 Individuen, deren Variablen binär- oder gray-kodiert waren. Einige Kombinationen der Verfahren wurden als am robustesten ermittelt. Außerdem wurde festgestellt, daß die gray-kodierten Individuen bessere Ergebnisse lieferten als die binär-kodierten. Die verwendeten Populationen stellten sich als zu klein heraus, es wurde die Verwendung größerer Populationen empfohlen. HOLLSTIEN untersuchte außerdem Diploidie mit einem sich entwickelnden Dominanzmechanismus. Dafür führte er neben den binären Variablen 0 und 1 eine dritte Variable 2 ein. Dadurch ändert sich die Bedeutung der Werte der Variablen. Die 1 wird zu einer rezessiven 1 und die 2 zu einer dominanten 1. Nur die 0 blieb 0. Damit werden zwei der drei Variablen, 1 und 2, zu einer 1, aber die 2 ist dominant über

eine 0, während die 0 dominant über eine 1 ist. Dies wird auch als dreialleliges Schema {0, 1, 2} bezeichnet. Nach der Anwendung eines Dominanzschemas bleiben nur noch die binären Variablen 0 und 1. Eines der diploiden Schemata von HOLLSTIEN hatte bei Simulationen eine höhere Diversität in der Population als das haploide Schema. Ansonsten wurden jedoch keine signifikanten Verbesserungen berichtet.

Die Arbeit von Box ([Box57]) stellt einen der frühen Fälle dar, bei dem der Begriff *evolutionary* zur Charakterisierung des benutzten Verfahrens verwendet wurde. Box beschrieb ein Verfahren zur Durchführung von Experimenten, die von einem aktuellen Arbeitspunkt ausgehen. Um den Arbeitspunkt wurde ein Hyperwürfel gelegt, entsprechend der Anzahl der Parameter. Die Ecken dieses Hyperwürfels wurden systematisch bewertet. Wenn eine deutliche Verbesserung gefunden wurde, so konnte ein neuer Arbeitspunkt festgelegt und ein neuer Hyperwürfel erstellt werden. Mit einem Evolutionären Algorithmus hat dieses Verfahren aber nur wenig gemeinsam, auch wenn Box schreibt, daß die Veränderungen als Mutationen angesehen werden können und die Auswahl eines besseren Punktes als Selektion. Das Verfahren von Box erinnert eher an einen Simplex-Algorithmus und sollte auch in dieser Richtung eingeordnet werden.

CAVICCHIO ([Cav70]) wendete einen genetischen Algorithmus auf die Entwicklung eines Sets von Detektoren zur Mustererkennung (hier zur Erkennung von Buchstaben) an. Dadurch wurde, im Gegensatz zur direkten Mustererkennung, das Problem der Buchstabenerkennung auf den Entwurf eines Sets von Detektoren reduziert. CAVICCHIO verwendete in seinem genetischen Algorithmus Selektion, Crossover und Mutation. Verschiedene Selektionsverfahren wurden getestet. Am Ende kam ein Verfahren zum Einsatz, daß gute Individuen bevorzugt. Gleichzeitig verhinderte das Verfahren, daß die guten Individuen zu viele Nachkommen erzeugten. Der Crossoveroperator ist ein *single-point crossover*, wobei sichergestellt wurde, daß nur zwischen Blöcken gekreuzt werden konnte (und nicht an jeder Stelle). Drei verschiedene Mutationsoperatoren wurden verwendet, die auf Grund der komplexen Repräsentation der Individuen notwendig waren. Außerdem wurden *inversion* (Umkehrung), *two-point crossover* und *duplication* (Vervielfachung) verwendet. CAVICCHIO unternahm den Versuch der automatischen Anpassung der Parameter seines Algorithmus. Die Parameter waren aber nicht Teil des Algorithmus selbst, sondern wurden von außen in Abhängigkeit der globalen Verbesserung verändert. Es konnten deutliche Vorteile des adaptiven genetischen Algorithmus gegenüber einem einfachen genetischen Algorithmus sowie einem anderen adaptiven Algorithmus und einer Zufallssuche, die zum Vergleich herangezogen wurden, gezeigt werden.

Ein sehr früher Versuch zur Generierung von Computerprogrammen wurde von FRIEDBERG ([Fri58], [FDN59]) unternommen. FRIEDBERG verwendete eine binäre Kodierung. Neue Programme wurden durch Austausch von Instruktionen sowie die zufällige Veränderung von Instruktionen erzeugt. Die erzeugten Computerprogramme sollten einfache Aufgaben lösen. Von FRIEDBERG wurde eine Erfolgszahl (*success number*) eingeführt, die angab, wie gut eine Instruktion in vorherigen Tests war. Die Mutationswahrscheinlichkeit einer Instruktion hing von dieser Erfolgszahl ab; je kleiner der Erfolg, um so größer war die Mutationswahrscheinlichkeit. Erfolgreiche Instruktionen wurden dadurch seltener mutiert. Der Ansatz von FRIEDBERG hatte zwei Hauptprobleme: Es fehlte ein ausrei-

chender Selektionsdruck<sup>1</sup>, und die verwendete Methode der Bewertung der Computerprogramme führte zu einer stark nichtkontinuierlichen Zielfunktion. Um das erste Problem des fehlenden Selektionsdruckes zu lösen, wurde vorgeschlagen, mehrere Programme auf der Grundlage eines Programmes durch Austausch von Instruktionen und zufällige Veränderung von Instruktionen zu generieren und das beste dieser Programme als Startpunkt für den nächsten Schritt zu verwenden. Diese Variante konnte wegen Beschränkungen der vorhandenen Rechentechnik aber nicht getestet werden.

Eine der ersten Veröffentlichungen der erfolgreichen Anwendung Evolutionärer Algorithmen zur Generierung und Optimierung strukturierter Lösungen und damit ein Vorläufer der Genetischen Programmierung stammt von FORSYTH [For81]. Er beschrieb das Computerprogramm *BEAGLE* zur Erstellung von Entscheidungsregeln. Die Regeln wurden als binäre Bäume kodiert, auf die Mutation und Rekombination angewendet wurden. Zwischen den Individuen (den Regeln) fand eine natürliche Auslese statt. Die Regeln wurden entsprechend ihrer Kategorisierungsgüte sortiert (*ranking*) und nachfolgend die Nachkommen produziert. Die besten 25% der Individuen überlebten unverändert, die nächsten 25% wurden durch Mutation vergrößert, die nächsten 25% durch Mutation verkleinert und die letzten 25% wurden aus den besten 25% durch Rekombination neu gebildet. Mit diesem System konnte FORSYTH sehr gute Ergebnisse für einige Klassifikationsprobleme erzielen, die sich nachfolgend durch ihre Verständlichkeit einfach anwenden ließen.

Neben den hier aufgeführten Arbeiten gab es einzelne weitere Veröffentlichungen. Tiefergehende Besprechungen weiterer früher Arbeiten finden sich u.a. in [Gol89], S.89-104, [Sch81], S.100-103, [Bäc96], S.57-60 und [Fdb95], S.67-75.

## A.1.2 Evolutionäre Programmierung

Von FOGEL, OWENS und WALSH ([FOW66]) wurde 1966 die Entwicklung und Anwendung der Evolutionären Programmierung (*Evolutionary Programming*) berichtet. Sie wendeten diese Methode auf ein relativ kompliziertes Problem an, die Vorhersage einer Kette von Symbolen (*sequential symbol prediction*).

Die Individuen stellten einfache finite Automaten (*finite-state machines*) dar. Neue Individuen wurden durch zufällige Änderungen im *transition table* erstellt. Ein Nachkomme wurde bewertet, indem ein Vergleich der durch das Individuum erzeugten Sequenz von Symbolen mit einer gegebenen Sequenz durchgeführt wurde. Der Prozentsatz der Übereinstimmungen ergab die Güte. Der Nachkomme und sein Elter wurden miteinander verglichen und nur der bessere überlebte. Das schlechtere Individuum wurde verworfen.

FOGEL, OWENS und WALSH war bekannt, daß so ein Algorithmus in lokalen Minima hängen bleiben kann. Deshalb arbeiteten sie mit einer Population von

<sup>1</sup> Der Selektionsdruck, der durch die Erfolgszahl hervorgerufen wurde, war nur bedingt geeignet, da viele Tests benötigt wurden, bevor eine annähernd nützliche Aussage gemacht werden konnte.

Individuen. Es wurden genauso viele Nachkommen erstellt, wie Individuen in der Population waren. Durch Selektion wurde die jeweils bessere Hälfte der Eltern und Nachkommen ausgewählt und bildete die neue Population.

Obwohl Mutation als der hauptsächliche Operator angesehen wurde, erwähnen FOGEL, OWENS und WALSH die Möglichkeit der Verwendung einer Rekombination, die sogar zwischen mehr als zwei Eltern stattfinden könne.

Die Arbeiten von FOGEL, OWENS und WALSH können als erste erfolgreiche Anwendung Evolutionärer Algorithmen angesehen werden. Sie wurden aber für viele Jahre nicht beachtet bzw. abgelehnt.

Erst in den 80er Jahren kam es wieder zu einer Belebung des *Evolutionary Programming*. Dies ist insbesondere ein Verdienst von DAVID FOGEL, der die über lange Zeit ignorierten Arbeiten seines Vaters wieder aufnahm bzw. fortführte.

In [Fdb92] und [Fdb95] werden ein Überblick sowie eine ausführliche Erläuterung des Ablaufs und der Operatoren des *Evolutionary Programming* gegeben. Dort werden auch Erweiterungen des Standard-EP, wie z.B. *Meta-EP*, *RMeta-EP* und *Continous-EP*, vorgestellt.

Das Gebiet des *Evolutionary Programming* ist auch heute noch gut überschaubar, da im Vergleich zu den anderen Schulen Evolutionärer Algorithmen nur wenige Leute auf diesem Gebiet arbeiten. Aber nicht zuletzt durch die seit 1992 jährlich stattfindende internationale Konferenz zu *Evolutionary Programming* (z.B. [EP96]) und die *Evolutionary Programming Society* ist das Interesse in den letzten Jahren stetig angestiegen.

### **A.1.3 Evolutionsstrategien**

Die Entwicklung der Evolutionsstrategien (*Evolution Strategies*) begann in den 60er Jahren an der Technischen Universität Berlin durch BIENERT, RECHENBERG und SCHWEFEL. Sie bearbeiteten verschiedene Probleme experimentell, die sich nicht analytisch beschreiben ließen und auf die keine traditionellen Methoden angewendet werden konnten. Deshalb entwickelten sie einen einfachen Algorithmus, mit dem die experimentellen Bedingungen verändert wurden. Der Algorithmus basierte auf zufälligen Veränderungen, wobei kleinere Veränderungen wahrscheinlicher waren als große Veränderungen. Wenn die neuen experimentellen Bedingungen eine Verbesserung erbrachten, wurden sie als Ausgangspunkt für die Einstellung neuer Bedingungen genutzt. Ansonsten dienten die vorhergehenden Bedingungen als Ausgangspunkt für weitere Experimente. Mit diesem einfachen Verfahren wurden verschiedene Probleme gelöst, u.a. Widerstandsmimierung einer Gelenkplatte im Windkanal 1964 [Rec73], Optimierung eines Rohrkrümmers [Rec73] und Strukturoptimierung einer Zweiphasen-Überschalldüse [Sch68]. Mit der Verfügbarkeit eines ersten Rechners simulierte SCHWEFEL verschiedene Versionen der Evolutionsstrategie genannten Experimentiermethode auf einem Computer. Andere wendeten nachfolgend die Evolutionsstrategie zur Lösung numerischer Optimierungsprobleme an.

Die ersten Arbeiten mit Evolutionsstrategien verwendeten ein Individuum, das einen Nachkommen produzierte. Ein Nachkomme entstand durch Mutation aus dem Elter. Das in den ersten Experimenten verwendete diskrete Mutationssche-

ma wurde wenig später durch ein kontinuierliches Mutationsschema unter Verwendung normal verteilter Mutationen ersetzt. Das bessere Individuum von Elter und Nachkomme wurde als Elter des nächsten Nachkommen verwendet. Diese einfache Evolutionsstrategie mit zwei Individuen bildete die Grundlage für die weiteren Entwicklungen der Evolutionsstrategie.

Eine Theorie der Evolutionsstrategien sowie ein Überblick über einige der frühen Anwendungen der Evolutionsstrategie wurde von RECHENBERG 1973 in [Rec73] veröffentlicht.

Die Verwendung einer vollständigen Population für Evolutionsstrategien wurde von SCHWEFEL entwickelt und u.a. in [Sch75] und [Sch81] vorgestellt. Neben der Verwendung einer Population von Individuen und Rekombination zur Produktion von Nachkommen bezogen diese Evolutionsstrategien außerdem die wichtigsten Parameter des Suchverfahrens (Standardabweichung und Korrelationskoeffizienten der normalverteilten Mutation) in den Optimierungsprozeß mit ein. Dadurch wurden nicht nur die Parameter des Problems optimiert, sondern auch die Parameter der Suchstrategie den aktuellen Gegebenheiten des Optimierungsproblems angepaßt (Selbstadaptation der Strategieparameter).

Nachdem es seit dem Ende der 70er Jahre etwas ruhiger um die Evolutionsstrategien geworden war, kam es ab Ende der 80er Jahre zu einem deutlichen Aufschwung des Interesses. Die Weiterentwicklung der Evolutionsstrategien wird auch heute hauptsächlich in Deutschland vorangetrieben. Die Zentren der Aktivitäten liegen an der TU Berlin (Prof. RECHENBERG) und an der TU Dortmund (Prof. SCHWEFEL).

In den letzten Jahren wurden viele Arbeiten zu Evolutionsstrategien veröffentlicht. Sehr gut zu lesen ist das Buch von RECHENBERG „Evolutionsstrategie 94“ [Rec94], in dem er die Weiterentwicklung der Evolutionsstrategien seit [Rec73] darstellt und sich mit anderen Evolutionären Algorithmen auseinandersetzt. BÄCK gibt in [Bäc96] einen Überblick über Evolutionäre Algorithmen, in welchem Evolutionsstrategien ausführlich behandelt werden. SCHWEFEL stellt in [Sch95] viele ableitungsfreie Optimierungsverfahren vor, einschließlich einer ausführlichen Darstellung verschiedener Evolutionsstrategien und vergleicht diese untereinander. Von OSTERMEIER, HANSEN und GAWELZYK werden in mehreren Arbeiten ([OGH93], [OGH94], [HOG95], [HO96], [Ost97] und [Han98]) neue Methoden für die Adaption der Strategieparameter einer Evolutionsstrategie (Schrittweitenanpassung der Mutation) vorgestellt.

Die Evolutionsstrategien haben heute einen festen Platz unter den Evolutionären Algorithmen. Dazu haben nicht nur die langjährige Entwicklung und der erfolgreiche Einsatz beigetragen, sondern auch die Ergebnisse zur Theorie der Evolutionsstrategien, u.a. Aussagen zur Fortschrittsgeschwindigkeit ([Rec94], [Bey95]) und der Konvergenz zum globalen Optimum.

#### A.1.4 Genetische Algorithmen

Genetische Algorithmen (*Genetic Algorithms*) sind der am weitesten bekannte Typ Evolutionärer Algorithmen. Der Begriff GENETISCHER ALGORITHMUS geht auf eine der frühen Arbeiten von BAGLEY [Bag67] zurück. In ihrer heute bekannten Form wurden sie aber von JOHN H. HOLLAND an der University of Mi-

chigan, USA, entwickelt. Breite Aufmerksamkeit erhielten Genetische Algorithmen ab 1975. Zu diesem Zeitpunkt erschien HOLLAND's Buch „*Adaptation in natural and artificial systems*“ [Hol75], in dem er seine Arbeiten zur Adaption in natürlichen und künstlichen Systemen zusammenfaßte. Dieses auch heute noch lesenswerte Buch zeigt dabei nicht nur, wie die natürlichen Suchalgorithmen zur Bearbeitung praktischer Probleme verwendet werden können, sondern wirft viele weitere Fragen auf, die bis heute der Beantwortung harren.

HOLLAND's Ziel war weit gesteckt. Er versuchte, eine allgemeine Theorie adaptiver Systeme zu entwickeln. Das Ziel der Arbeit war dabei zweigeteilt:

- Erklärung der adaptiven Prozesse in der Natur (Theorie) und
- Entwicklung von Software, welche die Mechanismen der natürlichen Systeme beibehält und die unbegrenzte Fähigkeit hat, sich den verschiedenen Gegebenheiten anzupassen.

Eines der herausragenden Ergebnisse von HOLLAND's Arbeit war die Formulierung des Schema-Theorems. Damit wurde der Versuch einer Erklärung unternommen, wie und warum Genetische Algorithmen funktionieren und arbeiten<sup>2</sup>.

Ein weiterer Meilenstein bei der Erforschung Genetischer Algorithmen ist die Dissertation von DE JONG [DeJ75]. Er analysierte eine Klasse von genetisch adaptiven Systemen. Dabei kombinierte er HOLLAND's Schema-Theorem mit seinen eigenen Experimenten. DE JONG stellte eine Testsuite von 5 Funktionen zusammen (DE JONG's Funktionen 1-5), die unterschiedliche Charakteristika aufwiesen und untersuchte das Verhalten verschiedenster Parametereinstellungen eines Genetischen Algorithmus. Das Verhalten des Genetischen Algorithmus wurde mittels zweier Kriterien quantifiziert (*online* und *offline* Performanz) und machte es damit vergleichbar. Weiterhin untersuchte DE JONG die Verfahren *elitism* und *crowding*. DE JONG's Experimente können als Grundlage vieler weiterer Experimente und Arbeiten mit Genetischen Algorithmen angesehen werden.

Die Arbeit DE JONG's gab Genetischen Algorithmen sowie deren Anwendung eine sichere Grundlage. Ausgehend von seinen Ergebnissen wurden viele Erweiterungen und Verbesserungen Genetischer Algorithmen erarbeitet.

GOLDBERG beschrieb in seiner Dissertation 1983 [Gol83] die erste erfolgreiche Anwendung Genetischer Algorithmen. Er beschäftigte sich mit der Optimierung der Steuerung von Gasleitungssystemen. Der verwendete Optimierungsalgorithmus war ein einfacher Genetischer Algorithmus mit proportionaler Selektion, binärer Mutation und Rekombination. Die Population bestand aus 50 Individuen. Beschränkungen wurden durch eine quadratische Gewichtungsfunktion einbezogen. Für die untersuchten Probleme erreichte GOLDBERG Ergebnisse, die sehr nah am Optimum lagen.

In den folgenden Jahren nahm das Interesse am Einsatz Genetischer Algorithmen stetig zu. Es wurden Anwendungen aus vielen Bereichen berichtet. Eine chronologische Übersicht der frühen Arbeiten und der Anwendung Genetischer Algorithmen in den verschiedensten Bereichen wurde von GOLDBERG in [Gol89], S.126-129 zusammengestellt.

---

<sup>2</sup> Die Gültigkeit des Schema-Theorems wird heute bezweifelt bzw. ist stark eingeschränkt (s. [Alt95], [Müh91] und [Gre93]).

### A.1.5 Evolutionäre Algorithmen heute

Bis Mitte der 80er Jahre entwickelten sich die drei Schulen Evolutionärer Algorithmen – Evolutionäre Programmierung, Evolutionsstrategien und Genetische Algorithmen – unabhängig voneinander. Erst ab 1985 begann ein Austausch von Ideen zwischen den Schulen Evolutionärer Algorithmen. Einen maßgeblichen Anteil daran haben die seitdem regelmäßig stattfindenden Konferenzen.

Die „*International Conference on Genetic Algorithms (ICGA)*“ ([ICGA1] – [ICGA7]) findet seit 1985 alle 2 Jahre in den USA statt und bringt diejenigen zusammen, die an Theorie und Anwendung Genetischer Algorithmen interessiert sind. Seit 1990 gibt es eine ähnliche Konferenz auf dem europäischen Kontinent „*Parallel Problems solving from Nature (PPSN)*“ ([PPSN1] – [PPSN5]), die jeweils in den anderen Jahren als ICGA stattfindet.

Seit 1992 findet jährlich die Konferenz „*Evolutionary Programming (EP)*“ (z.B. [EP96]) statt, seit 1994 die Konferenz „*IEEE International Conference on Evolutionary Computation (ICEC)*“ ([ICEC94] – [ICEC98]) als Teil des „*World Congress on Computational Intelligence (WCCI)*“ und seit 1996 wird jährlich eine eigenständige Konferenz für „*Genetic Programming (GP)*“ ([GP96] – [GP98]) veranstaltet. Seit Beginn der 90er Jahre gibt es eine Vielzahl weiterer Konferenzen, die sich mit verschiedenen Aspekten Evolutionärer Algorithmen beschäftigen (z.B. [AISB96]).

Durch den Zusammenschluß mehrerer Konferenzen 1999 ergab sich eine gewisse Konzentration auf wenige wichtige Veranstaltungen. Die Konferenzen ICEC, GALESIA und EP fanden als „*Congress on Evolutionary Computation CEC'99*“ zusammen statt. Die „*Genetic and Evolutionary Computation Conference GECCO'99*“ ist ein Zusammenschluß der ICGA und GP Konferenzen.

1990 ([PPSN1]) bzw. 1991 ([ICGA4]) verständigten sich die führenden Vertreter Genetischer Algorithmen und Evolutionsstrategien sowie der Evolutionären Programmierung darauf, den Begriff **Evolutionäre Algorithmen** als gemeinsame Bezeichnung für die Algorithmen der verschiedenen Schulen zu verwenden. Für die Berechnungen selbst wird außerdem der Begriff *evolutionary computation* verwendet, was sich nicht zuletzt im Titel des internationalen Journals „*Evolutionary Computation*“ [ECJ] widerspiegelt, das seit 1993 vierteljährlich erscheint und Beiträge aus allen Bereichen der Evolutionären Algorithmen publiziert. Ein weiteres Journal in diesem Bereich ist *IEEE Transactions on Evolutionary Computation* [ECT], das seit 1997 vierteljährlich erscheint.

In den letzten Jahren wurden viele neue Varianten und Verfahren im Bereich der Evolutionären Algorithmen entwickelt. Besondere Bedeutung hat seit Beginn der 90er Jahre die Methode der Genetischen Programmierung (*Genetic Programming*) gewonnen, die maßgeblich durch die Arbeiten von KOZA ([Koz92], [Koz94], [Koz99]) entwickelt wurde. Diese Methode arbeitet mit einer strukturierten Repräsentation der Individuen. Grundgedanke war die Evolution von Computerprogrammen. Die Genetische Programmierung kann aber auf viele andere Probleme angewendet werden, bei denen die Lösungen strukturiert vorliegen (z.B. Baumstruktur, Programm, Konstruktionen usw.).

Durch die Entwicklungen der letzten Jahre und den Austausch zwischen den Forschern sollte über die Kategorisierung der einzelnen Algorithmen neu nach-

gedacht werden. Eine Einordnung im Sinne der historisch vorhandenen Unterschiede kann als veraltet gelten und ist nicht länger sinnvoll. Das Augenmerk sollte weniger auf die zwischen den Algorithmen vorhandenen Unterschiede gelegt werden, sondern vielmehr auf die Richtungen und Bereiche, in denen die Komponenten der Evolutionären Algorithmen neu zusammengestellt, erweitert und angewendet werden können.

## A.2 Gewächshaus- und Pflanzenmodell

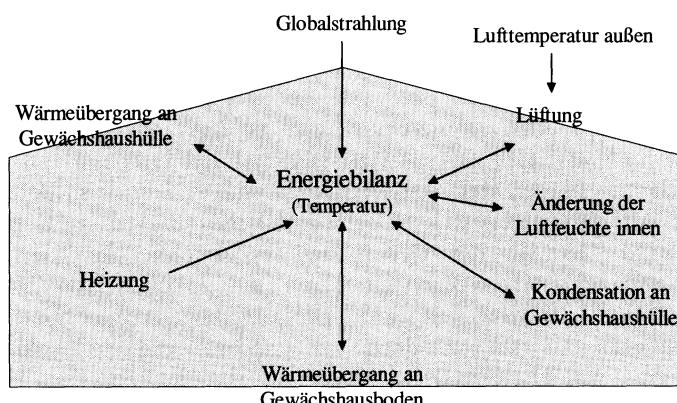
In diesem Abschnitt werden die Bilanzgleichungen des Gewächshauses und die Zustandsgleichungen des Pflanzenmodells ausführlich dargestellt. Diese Erläuterungen erweitern die Beschreibung aus Abschn. 8.5.

### A.2.1 Zustandsgleichungen des Gewächshauses

#### Energiebilanz

Die sich aus der Energiebilanz, Abb. A2-1, ergebende Temperaturänderung,  $DTEMI$  [K/s], wird durch folgende Komponenten dargestellt:

1. Wärmezufuhr durch Heizen,  $Q$ ,
2. Erwärmung durch Globalstrahlung,  $q_{glob}$ ,
3. Änderung der Luftfeuchte,  $DDDI$ ,
4. Energieaustausch mit der Umgebung durch Lüftung,  $q_{luwe}$ ,
5. Wärmedurchgang durch die Gewächshaushülle,  $qduga$ ,
6. Wärmetübertragung an der Gewächshausbodenoberfläche,  $qboden$  und
7. Kondensation/Verdunstung an der Gewächshaushülle,  $qkonden$ .



**Abb. A2-1.** Schema der Energiebilanz

$$DTEMI = \frac{1}{CG + GH \cdot CPD \cdot DDI} \cdot (Q + q_{glob} - q_{luwe} - q_{duga} - q_{boden} - q_{konden} - \dots \\ GH \cdot (VDWO + CPD \cdot TEMI) \cdot DDDI) \quad (A2-1)$$

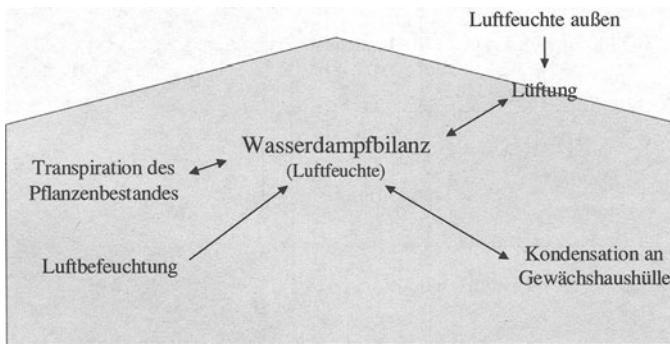
### **Wasserdampfbilanz**

Die Dampfdichteänderung,  $DDDI$  [ $\text{g}/(\text{m}^3 \cdot \text{s})$ ], die sich aus der Wasserdampfbilanz ergibt, wird durch folgende Komponenten dargestellt, Abb. A2-2:

1. Transpiration des Pflanzenbestandes,  $trans$ ,
2. Luftbefeuchtung,  $lube$ ,
3. Wasserdampfaustausch mit der Umgebung durch Lüftung,  $wadawe$  und
4. Kondensation/Verdunstung an der Gewächshaushülle,  $kondverd$ .

$$DDDI = \frac{1}{GH \cdot 3600} \cdot (trans + lube - wadawe - kondverd) \quad (A2-2)$$

Kondensation (Verdunstung) tritt an der Gewächshaushülle auf, wenn der Funktionswert von  $kondverd$  positiv (negativ) ist. Bei Überschreiten der maximalen Kondensatflächendichte fließt Kondensat an der Gewächshaushülle ab. Ist kein Kondensat vorhanden, wird  $kondverd$  zu 0 gesetzt (keine Verdunstung).



**Abb. A2-2.** Schema der Wasserdampfbilanz

### **Kohlendioxidbilanz**

Die  $\text{CO}_2$ -Bilanz, Abb. A2-3, aus der sich die Änderung der  $\text{CO}_2$ -Konzentration,  $DCI$  [ppm/s], ergibt, wird durch folgende Komponenten dargestellt:

1.  $\text{CO}_2$ -Zufuhr durch Begasung,  $W$ ,
2.  $\text{CO}_2$ -Gaswechsel (Photosynthese und Respiration) des Pflanzenbestandes,  $gawebfl$  und
3.  $\text{CO}_2$ -Austausch mit der Umgebung durch Lüftung,  $kodiwe$ .

$$DCI = (W - gawebfl - kodiwe) \cdot \frac{1}{DC \cdot GH \cdot 3600 \cdot 10^{-6}} \quad (\text{A2-3})$$

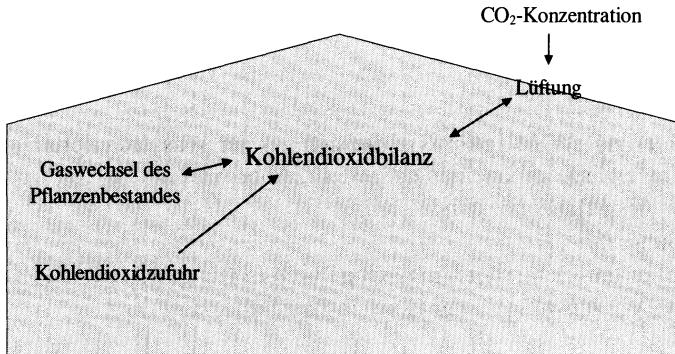


Abb. A2-3. Schema der Kohlendioxidbilanz

## A.2.2 Zustandsgleichungen des Pflanzenmodells (Paprika)

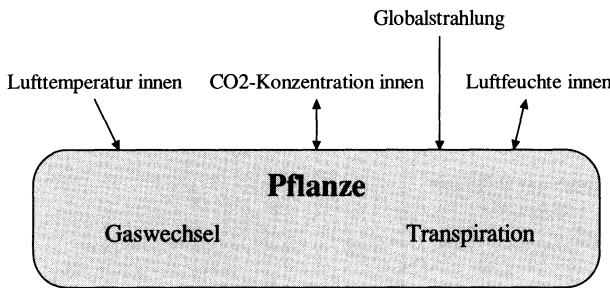


Abb. A2-4. Schema des Pflanzenwachstumsmodells

### **CO<sub>2</sub>-Gaswechsel**

Der auf die Blattfläche bezogene CO<sub>2</sub>-Gaswechsel der Pflanzen, *gawebfl* [g/(m<sup>2</sup>·h)], Abb. A2-4, wird wie folgt berechnet:

$$gawebfl = P_0 \cdot FICT \cdot FSD \quad (\text{A2-4})$$

*P<sub>0</sub>* ist die CO<sub>2</sub>-Gaswechselrate unter Standardbedingungen. Die Größe *FICT* wird mittels einer parametrischen Gleichung ermittelt:

$$\begin{aligned} FICT &= C_1 \cdot (1 - \exp(-C_2 \cdot 0.5 \cdot I)) \cdot (1 - \exp(-C_3 \cdot CI)) \cdots \\ &\quad (TEMI + C_4 \cdot TEMI^2) - C_5 \cdot (TEMI + C_6 \cdot TEMI^2) \end{aligned}$$

Die Größe  $FSD$  wird durch eine von 3 Gleichungen in Abhängigkeit des Wertes des Dampfdrucksättigungsdefizits innen,  $SDI$ , bestimmt:

$$SDI < SDIG_1 \quad FSD = \exp\left(EE_1 \cdot (SDIG_1 - SDI)^2\right)$$

$$SDIG_1 \leq SDI \leq SDIG_2 \quad FSD = 1$$

$$SDIG_2 < SDI \quad FSD = \exp\left(EE_2 \cdot (SDIG_2 - SDI)^2\right)$$

### Transpiration

Die auf die Blattfläche bezogene Transpiration der Pflanzen,  $trblfl$  [g/(m<sup>2</sup>·h)], Abb. A2-4, ergibt sich aus:

$$trblfl = V_0 \cdot VREL \quad (\text{A2-5})$$

$V_0$  ist die Transpirationsrate unter Standardbedingungen. Die Größe  $VREL$  wird durch 3 parametrische Gleichungen berechnet:

$$VREL = VRELC \cdot SDI \cdot \left( B_1 + B_2 \cdot I + B_3 \cdot I^2 + B_4 \cdot \frac{\left(\frac{PDL}{PSI}\right) \cdot 100}{VSTAN} \right)$$

$$VRELC = 1 - B_0 \cdot (CI - 600)$$

$$VSTAN = 10 \cdot (B_1 + B_2 \cdot 300 + B_3 \cdot 300^2 + B_4 \cdot 60)$$

### A.2.3 Biomasse und Gewinn

Der Gewinn,  $GEWI$  [0,01 DM/(m<sup>2</sup>·h)], Abb. A2-5, wird aus folgenden Komponenten bestimmt:

1. Biomasse,  $BIOM$ ,
2. Gemüsepreis,  $PR_1$ ,
3. Kosten der CO<sub>2</sub>-Zufuhr,  $PR_2$ , und
4. Kosten für Heizung,  $PR_3$ .

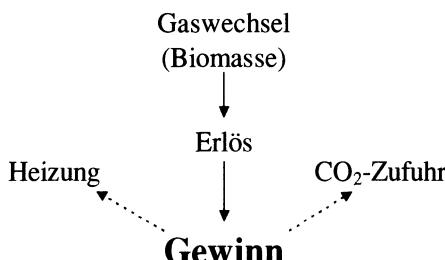


Abb. A2-5. Schema der Gewinnberechnung

Die Biomasse,  $BIOM$ , ist die aus dem CO<sub>2</sub>-Gaswechsel der Pflanzen,  $g_{\text{awebfl}}$ , berechnete Trockensubstanzproduktion, die durch Multiplikation mit dem Anteil am Ertrag (50%) und durch Division mit dem Trockensubstanzgehalt (5%) in eine Ertragsgröße,  $Ertrag$ , und nachfolgend durch Multiplikation mit dem Gemüsepreis in eine Erlösgröße,  $GEWINN$ , umgerechnet wird. Die Kosten für die CO<sub>2</sub>-Zufuhr werden aus dem CO<sub>2</sub>-Preis errechnet. Analog erfolgt die Berechnung der Kosten der Wärmezufuhr durch Heizen über den Energiepreis für Heizöl.

$$GEWINN = Ertrag \cdot PR_1 - W \cdot PR_2 - Q \cdot PR_3 \quad (\text{A2-6})$$

**Tabelle A2-1.** Komponenten der Berechnung des Gewinns

Komponenten	Symbol	Einheit	Funktion / Wert
Biomasse	$BIOM$	$\text{g}/(\text{m}^2 \cdot \text{h})$	$g_{\text{awebfl}} \cdot (30/44)$
Ertrag	$Ertrag$	$\text{g}/(\text{m}^2 \cdot \text{h})$	$BIOM \cdot 0,5/0,05$
Gemüsepreis	$PR_1$	0,01 DM/g	0,3
CO <sub>2</sub> -Preis	$PR_2$	0,01 DM/g	0,1
Energiepreis	$PR_3$	0,01 DM/(W·h)	0,005

#### A.2.4 Beschränkungen

Die Beschränkungen der Stellgrößen sind durch die Grenzen der Wertebereiche der Steuerungsvariablen für Heizung/Wärmezufuhr, Lüftung und CO<sub>2</sub>-Zufuhr definiert. In den Untersuchungen wurde keine Luftbefeuhtung vorgenommen.

**Tabelle A2-2.** Beschränkungen der Steuer- und Zustandsgrößen

Komponente	Minimum	Maximum
Heizung/Wärmezufuhr	0 W/m <sup>2</sup>	150 W/m <sup>2</sup>
Lüftung	1 m <sup>3</sup> /(m <sup>2</sup> ·h)	100 m <sup>3</sup> /(m <sup>2</sup> ·h)
CO <sub>2</sub> -Zufuhr	0 g/(m <sup>2</sup> ·h)	10 g/(m <sup>2</sup> ·h)
Temperatur im Gewächshaus	16°C	35°C

Die Zustände des Gewächshausklimas sind zur Vermeidung von Streß für die Pflanzen und zur Einhaltung von Randbedingungen, die sich aus langfristigen Strategien ergeben, beschränkt. Vorgenommen wird eine Beschränkung der Temperatur im Gewächshaus.

# Literaturverzeichnis

Das Literaturverzeichnis enthält die Literatur, die bei der Entstehung dieses Buches verwendet wurde. Um eine bessere Übersichtlichkeit zu gewährleisten, wurden die Einträge entsprechend den in dieser Arbeit vertiefend behandelten Gebieten geordnet. Dadurch wird es möglich, sich einen schnellen Überblick der auf dem jeweiligen Gebiet wichtigen und interessanten Arbeiten zu verschaffen.

## Evolutionäre Algorithmen und Optimierung

- [Ack87] *Ackley, D. H.*: A connectionist machine for genetic hillclimbing. Boston: Kluwer Academic Publishers, 1987.
- [Ack93] *Ackermann, J.*: Robuste Regelung. Berlin, Heidelberg, New York: Springer-Verlag, 1993.
- [AISB96] *Fogarty, T. C.*: Evolutionary Computing. Proceedings of AISB Workshop on Evolutionary Computing 1996, Vol. 1143 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1996.
- [Alt95] *Altenberg, L.*: The Schema Theorem and Price's Theorem. In [FGA3], pp. 23-49, 1995.  
<http://pueo.mhpcc.edu/~altenber/PAPERS/STPT/>
- [AGP94] *Kinnear, K. E.*: Advances in Genetic Programming. Cambridge: MIT Press, 1994.
- [AGP96] *Angeline, P. J. and Kinnear, K. E.*: Advances in Genetic Programming II. Cambridge: MIT Press, 1996.
- [AGP99] *Spector, L., Langdon, W., O'Reilly, U.-M. and Angeline, P. J. (eds.)*: Advances in Genetic Programming III. Cambridge: MIT Press, 1999.
- [Bäc93] *Bäck, T.*: Optimal Mutation Rates in Genetic Search. In [ICGA5], pp. 2-8, 1993.  
<http://lumpi.informatik.uni-dortmund.de/people/baeck/papers/icga93.ps.Z>
- [Bäc96] *Bäck, T.*: Evolutionary Algorithms in Theory and Practice – Evolution Strategies, Evolutionary Programming, Genetic Algorithms. New York, Oxford: Oxford University Press, 1996.  
<http://www.oup-usa.org/docs/0195099710.html>
- [BH91] *Bäck, T. and Hoffmeister, F.*: Extended Selection Mechanisms in Genetic Algorithms. In [ICGA4], pp. 92-99, 1991.
- [BS93] *Bäck, T. and Schwefel, H.-P.*: An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1), pp. 1-23, 1993.  
<http://lumpi.informatik.uni-dortmund.de/people/baeck/papers/ec93.ps.Z>
- [BS96] *Bäck, T. and Schütz, M.*: Intelligent Mutation Rate Control in Canonical Genetic Algorithms. In *Ras, Z. W. and Michalewicz, M.: Foundation of Intelligent Systems. 9th International Symposium, ISMIS '96*, pp. 158-167, Berlin: Springer-Verlag, 1996.  
<http://lumpi.informatik.uni-dortmund.de/people/baeck/papers/ismis.ps.Z>

- [Bak85] *Baker, J. E.*: Adaptive Selection Methods for Genetic Algorithms. In [ICGA1], pp. 101-111, 1985.
- [Bak87] *Baker, J. E.*: Reducing Bias and Inefficiency in the Selection Algorithm. In [ICGA2], pp. 14-21, 1987.
- [Bal94] *Baluja, S.*: Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Technical Report CMU-CS-94-163, Pittsburgh, Pennsylvania: School of Computer Science, Carnegie Mellon University, 1994.  
<ftp://reports.adm.cs.cmu.edu/1994/CMU-CS-94-163.ps>
- [Bey95] *Beyer, H.-G.*: Toward a Theory of Evolution Strategies: On the Benefits of Sex – the  $(\mu/\mu,\lambda)$  Theory. *Evolutionary Computation*, 3(1), pp. 81-111, 1995.
- [BT95] *Blickle, T. and Thiele, L.*: A Comparison of Selection Schemes used in Genetic Algorithms (2. Edition). TIK Report No. 11, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zürich, Switzerland, 1995.  
<http://www.tik.ee.ethz.ch/Publications/TIK-Reports/TIK-Report11abstract.html>
- [Bli97] *Blickle, T.*: Theory of Evolutionary Algorithms and Application to System Synthesis. Ph.D. thesis. TIK-Schriftenreihe Nr. 17., Zürich: vdf Verlag, 1997.  
<http://www.handshake.de/user/blickle/publications/index.html>
- [BEL95] *Böcker, J., Endrikat, C. and Liu, S.*: A Systematic Approach to State Feedback Controller Design for DC/DC Line-Side Traction Converters. Proc. EPE'95, Sevilla, Vol. 1, pp. 314-318, 1995.
- [BW96] *Böcker, J. and Wu, Z.*: Symmetry Properties of Multi-Variable Control Systems. Technical Note, 25/96, Daimler Benz AG, 1996.
- [Boo87] *Booker, L.*: Improving search in genetic algorithms. In [Dav87], pp. 61-73, 1987.
- [Box57] *Box, G. E. P.*: Evolutionary operation: A method for increasing industrial productivity. In *Journal of the Royal Statistical Society, C*, 6(2), pp. 81-101, 1957.
- [Bra72] *Branin, F. K.*: A widely convergent method for finding multiple solutions of simultaneous nonlinear equations. *IBM J. Res. Develop.*, pp. 504-522, Sept., 1972.
- [Bre62] *Bremermann, H. J.*: Optimization through evolution and recombination. In *Yovits, M. C. et al.*: Self-organizing systems. Washington, DC: Spartan Books, pp. 93-106, 1962.
- [CEC99] *Angeline, P. J. (ed.)*: Proceedings of the Congress on Evolutionary Computation. Piscataway, New Jersey, USA: IEEE Press, 1999.
- [Cha95] *Chambers, L. (ed.)*: Genetic Algorithm Applications Vol. I. New York: CRC Press, 1995.
- [CS88] *Caruana, R. A. and Schaffer, J. D.*: Representation and Hidden Bias: Gray v. Binary Coding for Genetic Algorithms. In Fifth International Conference on Machine Learning, San Mateo, California, USA: Morgan Kaufmann Publishers, pp. 153-161, 1988.
- [CES89] *Caruana, R. A., Eshelmann, L. A. and Schaffer, J. D.*: Representation and hidden bias II: Eliminating defining length bias in genetic search via shuffle crossover. In *Sridharan, N. S. (ed.)*: Eleventh International Joint Conference on Artificial Intelligence, San Mateo, California, USA: Morgan Kaufmann Publishers, Vol. 1, pp. 750-755, 1989.
- [Cav70] *Cavicchio, D. J.*: Adaptive search using simulated evolution. Unpublished doctoral dissertation, University of Michigan, Ann Arbor, 1970.
- [CF94] *Chipperfield, A. J. and Fleming, P. J.*: Parallel Genetic Algorithms: A Survey. Technical Report No. 518, Department of Automatic Control and Systems Engineering, University of Sheffield, 1994.

- [CFP94b] *Chipperfield, A. J., Fleming, P. J. and Pohlheim, H.*: A Genetic Algorithm Toolbox for MATLAB. Proc. Int. Conf. Sys. Engineering, Coventry, UK, 6-8 Sept., pp. 200-207, 1994.
- [CFPF94a] *Chipperfield, A., Fleming, P. J., Pohlheim, H. and Fonseca, C. M.*: Genetic Algorithm Toolbox for use with Matlab. Technical Report No. 512, Department of Automatic Control and Systems Engineering, University of Sheffield, 1994.
- [Coe99] *Coello, C. A.*: List of References on Evolutionary Multiobjective Optimization. Laboratorio Nacional de Informática Avanzada, México, 1999.  
<http://www.lania.mx/~ccoello/EMOO/EMOObib.html>
- [CMR91] *Cohoon, J. P., Martin, W. N. and Richards, D.S.*: Genetic Algorithms and Punctuated Equilibria in VLSI. In [PPSN1], pp. 134-144, 1991.
- [Dav91a] *Davidor, Y.*: A Naturally Occurring Niche & Species Phenomenon: The Model and First Results. In [ICGA4], pp. 257-263, 1991.
- [Dav91b] *Davidor, Y.*: Epistasis Variance: A Viewpoint on GA-Hardness. In [FGA1], pp. 23-35, 1991.
- [Dav87] *Davis, L. D.*: Genetic Algorithms and Simulated Annealing. San Mateo, California, USA: Morgan Kaufmann Publishers, 1987.
- [Dav91] *Davis, L. D.*: Handbook of Genetic Algorithms. Van Nostrand Reinhold, 1991.
- [DJS93] *De Jong, K. and Spears, W.*: On the state of evolutionary computation. In [ICGA5], pp. 618-623, 1993.
- [DeJ75] *De Jong, K.*: An analysis of the behavior of a class of genetic adaptive systems. Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 36(10), 5140B, University Microfilms No. 76-9381, 1975.
- [DS78] *Dixon, L. C. W. and Szego, G. P.*: The optimization problem: An introduction. In *Dixon, L. C. W. and Szego, G. P. (ed.)*: Towards Global Optimization II, New York: North Holland, 1978.
- [Eas90] *Easom, E. E.*: A survey of global optimization techniques. M. Eng. thesis, Univ. Louisville, Louisville, KY, 1990.
- [ECJ] *Whitley, D. (ed.)*: Evolutionary Computation. Journal, Cambridge, Massachusetts: MIT Press, 1993-1999.
- [ECT] *Fogel, D. B. (ed.)*: IEEE Transactions on Evolutionary Computation. Journal, Piscataway, New Jersey, USA: IEEE Press.
- [EP96] *Fogel, D. B. (ed.)*: Evolutionary Programming V, Proceedings of the Fifth Annual Conference on Evolutionary Programming. Cambridge, MA: MIT Press, 1996.
- [EP97] *Angeline, P. J., Reynolds, R. G., McDonnell, J. R. and Eberhart, R. (eds.)*: Evolutionary Programming VI, Proceedings of the Sixth Annual Conference on Evolutionary Programming. Vol. 1213 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1997.
- [EP98] *Porto, V. W., Saravanan, N., Waagen, D. and Eiben, A. E. (eds)*: Evolutionary Programming VII, Proceedings of the Seventh Annual Conference on Evolutionary Programming. Vol. 1447 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1998.
- [Esh91] *Eshelman, L. J.*: The CHC Adaptive Algorithm: How to have safe search when engaging in Nontraditional Genetic Recombination. In [FGA1], pp. 265-283, 1991.
- [Fdb92] *Fogel, D. B.*: Evolving Artificial Intelligence. Dissertation, University of California, San Diego, 1992.
- [Fdb94a] *Fogel, D. B.*: An Introduction to Simulated Evolutionary Optimization. IEEE Trans. on Neural Networks: Special Issue on Evolutionary Computation, Vol. 5, No. 1, pp. 3-14, 1994.
- [Fdb94b] *Fogel, D. B.*: Applying Evolutionary Programming to Selected Control Problems. Comp. Math. App., 11(27), pp. 89-104, 1994.

- [Fdb95] *Fogel, D. B.:* Evolutionary computation: toward a new philosophy of machine intelligence. New York: IEEE Press, 1995.  
<http://www.natural-selection.com/misc/evolCompBook.html>
- [FOW66] *Fogel, L. J., Owens, A. J. and Walsh, M. J.:* Artificial Intelligence through Simulated Evolution. New York: John Wiley, 1966.
- [FF93] *Fonseca, C. M. and Fleming, P. J.:* Genetic Algorithms for Multiple Objective Optimization: Formulation, Discussion and Generalization. In [ICGA5], pp. 416-423, 1993.
- [FF95a] *Fonseca, C. M. and Fleming, P. J.:* Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms I: A Unified Formulation. Research report 564, Dept. Automatic Control and Systems Eng., University of Sheffield, Sheffield, U.K., 1995.
- [FF95b] *Fonseca, C. M. and Fleming, P. J.:* Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms II: Application Example. Research report 565, Dept. Automatic Control and Systems Eng., University of Sheffield, Sheffield, U.K., 1995.
- [FF95c] *Fonseca, C. M. and Fleming, P. J.:* An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1), pp. 1-16, 1995.
- [Fon95] *Fonseca, C. M.:* Multiobjective Genetic Algorithms with Application to Control Engineering Problems. Ph.D. Thesis, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U.K., 1995.
- [FF96] *Fonseca, C. M. and Fleming, P. J.:* On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In [PPSN4], pp.584-593, 1996.
- [For81] *Forsyth, R.:* BEAGLE – A Darwinian Approach to Pattern Recognition. *Kybernetes*, 10, pp. 159-166, 1981.
- [Fra62] *Fraser, A. S.:* Simulation of genetic systems. *Journal of Theoretical Biology*, 2, pp. 329-346, 1962.
- [Fri58] *Friedberg, R. M.:* A learning machine: Part I. *IBM Journal*, 2(1), pp. 2-13, 1958.
- [FDN59] *Friedberg, R. M., Dunham, B. and North, J. H.:* A learning machine: Part II. *IBM Journal*, 3(7), pp. 282-287, 1959.
- [FGA1] *Rawlins, G. J. E.:* Foundations of Genetic Algorithms. San Mateo, California, USA: Morgan Kaufmann Publishers, 1991.
- [FGA2] *Whitley, L. D.:* Foundations of Genetic Algorithms 2. San Mateo, California, USA: Morgan Kaufmann Publishers, 1993.
- [FGA3] *Whitley, L. D. and Vose, M. D.:* Foundations of Genetic Algorithms 3. San Francisco, California, USA: Morgan Kaufmann Publishers, 1995.
- [FGA4] *Belew, R. K. and Vose, M. D.:* Foundations of Genetic Algorithms 4. San Francisco, California, USA: Morgan Kaufmann Publishers, 1997.
- [FGA5] *Banzhaf, W. and Reeves, C.:* Foundations of Genetic Algorithms 5. San Francisco, California, USA: Morgan Kaufmann Publishers, 1999.
- [GAOT] *Houck, C., Joines, J. and Kay, M.:* The Genetic Algorithm Optimization Toolbox (GAOT). North Carolina State University, Department of Industrial Engineering, NCSU-IE TR 95-09, 1995.  
<http://www.ie.ncsu.edu/gaot/>
- [GEC99] *Banzhaf, W. (ed.):* GECCO'99 – Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA: Morgan Kaufmann, 1999.
- [GP96] *Koza, J. R., Goldberg, D. E., Fogel, D. B. and Riolo, R. L.:* Genetic Programming 1996: Proceedings of the First Annual Conference. Cambridge: MIT Press, 1996.
- [GP97] *Koza, J. R. et al. (eds.):* Genetic Programming 1997: Proceedings of the Second Annual Conference. San Francisco, CA: Morgan Kaufmann, 1997.

- [GP98] *Koza, J. R. et al. (eds.): Genetic Programming 1998: Proceedings of the Third Annual Conference.* San Francisco, CA: Morgan Kaufmann, 1998.
- [Gol83] *Goldberg, D. E.: Computer-aided gas pipeline operation using genetic algorithms and rule learning.* Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 44(10), 3174B, University Microfilms No. 8402282, 1983.
- [Gol87] *Goldberg, D. E.: Simple Genetic Algorithms and the Minimal Deceptive Problem.* In [Dav87], pp. 74-88, 1987.
- [Gol89] *Goldberg, D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, Mass.: Addison-Wesley, 1989.
- [Gol99] *Goldberg, D. E.: Genetic and Evolutionary Algorithms in the Real World.* Technical Report IlliGAL No. 99013, University of Illinois at Urbana-Champaign, 1999.  
<ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/99013.ps.Z>
- [GD91] *Goldberg, D. E. and Deb, K.: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms.* In [FGA1], pp. 69-93, 1991.
- [GP71] *Goldstein, A. A. and Price, I. F.: On descent from local minima.* Math. Comput., Vol. 25, No. 115, 1971.
- [Gre86] *Grefenstette, J. J.: Optimization of Control Parameters for Genetic Algorithms.* In IEEE Transactions on Systems, Man and Cybernetics, 16 (1986) 1, pp.122-128, 1986.
- [GB89] *Grefenstette, J. J. and Baker, J. E.: How Genetic Algorithms Work: A Critical Look at Implicit Parallelism.* In [ICGA3], pp. 20-27, 1989.
- [Gre93] *Grefenstette, J. J.: Deception Considered Harmful.* In [FGA2], pp. 75-91, 1983.
- [HH94] *Haas, R. and Hunt, K. J.: Genetic based optimisation of a fuzzy-neural vehicle controller.* Technical Report, Daimler Benz AG, Research and Technology Berlin, 1994.
- [Ham97] *Hammel, U.: Evolutionary Computation Applications: Simulation models.* In [HEC97], F1.8, pp. F1.8:1-F1.8:9, 1997.
- [Han98] *Hansen, N.: Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie – eine Untersuchung zur entstochastisierten, koordinatenunabhängigen Adaption der Mutationsverteilung.* Berlin: Mensch & Buch Verlag, 1998
- [HOG95] *Hansen, N., Ostermeier, A. and Gawelczyk, A.: On the Adaptation of Arbitrary Mutation Distributions in Evolution Strategies: The Generating Set Adaptation.* In [ICGA6], pp. 57-64, 1995.  
<ftp://ftp-bionik.fb10.tu-berlin.de/pub/papers/Bionik/GSAES.ps.Z>
- [HGO95] *Hansen, N., Gawelczyk, A. and Ostermeier, A.: Sizing the Population with Respect to the Local Progress in  $(1,\lambda)$ -Evolution Strategies – A Theoretical Analysis.* In [ICEC95], pp. 80-85, 1995.
- [HO96] *Hansen, N. and Ostermeier, A.: Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation.* In [ICEC96], pp. 312-317, 1996.  
<ftp://ftp-bionik.fb10.tu-berlin.de/pub/papers/Bionik/CMAES.ps.Z>
- [HH98] *Haupt, R. L. and Haupt, S. E.: Practical Genetic Algorithms.* New York: John Wiley & Sons, 1998.
- [HB91] *Hoffmeister, F. and Bäck, T.: Genetic Algorithms and Evolutionary Strategies: Similarities and Differences.* In [PPSN1], pp. 455-469, 1991.
- [HEC97] *Bäck, T., Fogel, D. B. and Michalewicz, Z. (eds.): Handbook on Evolutionary Computation.* Bristol: Institute of Physics Publishing and Oxford, New York: Oxford University Press, 1997.
- [Hol75] *Holland, J. H.: Adaptation in natural and artificial systems.* Ann Arbor: The University of Michigan Press, 1975.

- [Hol95] *Holland, J. H.*: Hidden order: how adaptation builds complexity. Reading, Massachusetts: Addison-Wesley, 1995.
- [Hoo95] *Hooker, J. N.*: Testing Heuristics: We Have It All Wrong. *Journal of Heuristics*, 1 (1995), pp. 33-42, 1995.
- [HN93] *Horn, J. and Nafpliotis, N.*: Multiobjective optimization using the niched pareto genetic algorithm. IlligAL Report 93005, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana, Champaign, 1993.
- [HG95] *Horn, J. and Goldberg, D. E.*: Genetic Algorithm Difficulty and the Modality of Fitness Landscapes. In [FGA3], pp. 243-269, 1995.
- [Hor97] *Horn, J.*: Evolutionary Computation Applications: Multicriterion decision making. In [HEC97], F1.9, pp. F1.9:1-F1.9:15, 1997.
- [ICEC94] *Fogel, D. B.*: Proceedings of The First IEEE Conference on Evolutionary Computation, Piscataway, New Jersey, USA: IEEE Service Center, 1994.
- [ICEC95] Proceedings of the Second IEEE Conference on Evolutionary Computation 1995, Piscataway, New Jersey, USA: IEEE Press, 1995.
- [ICEC96] Proceedings of the 1996 IEEE Conference on Evolutionary Computation, Piscataway, New Jersey, USA: IEEE Press, 1996.
- [ICEC97] Proceedings of the 1997 IEEE Int. Conf. on Evolutionary Computation, Piscataway, New Jersey, USA: IEEE Press, 1997.
- [ICEC98] Proceedings of the 1998 IEEE Int. Conf. on Evolutionary Computation, Piscataway, New Jersey, USA: IEEE Press, 1998.
- [ICGA1] *Grefenstette, J. J. (ed.)*: Proceedings of an International Conference on Genetic Algorithms and their Application, Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates, 1985.
- [ICGA2] *Grefenstette, J. J. (ed.)*: Proceedings of the Second International Conference on Genetic Algorithms and their Application, Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates, 1987.
- [ICGA3] *Schaffer, J. D. (ed.)*: Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, California, USA: Morgan Kaufmann Publishers, 1989.
- [ICGA4] *Belew, R. K. and Booker, L. B. (eds.)*: Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo, California, USA: Morgan Kaufmann Publishers, 1991.
- [ICGA5] *Forrest, S. (ed.)*: Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California, USA: Morgan Kaufmann Publishers, 1993.
- [ICGA6] *Eshelman, L. J. (ed.)*: Proceedings of the Sixth International Conference on Genetic Algorithms, San Francisco, California, USA: Morgan Kaufmann Publishers, 1995.
- [ICGA7] *Bäck, T. (ed.)*: Proceedings of the Seventh International Conference on Genetic Algorithms, San Francisco, California, USA: Morgan Kaufmann Publishers, 1997.
- [Jac95] *Jacob, C.*: MathEvolvica – Simulierte Evolution von Entwicklungsprogrammen der Natur. Dissertation, Arbeitsberichte des Instituts für mathematische Maschinen und Datenverarbeitung (Informatik), Universität Erlangen-Nürnberg, Band 28, Nummer 10, 1995.
- [JF95] *Jones, T. and Forrest, S.*: Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In [ICGA6], pp. 184-192, 1995.  
<http://www.santafe.edu/sfi/publications/Working-Papers/95-02-022.ps>
- [Koz92] *Koza, J. R.*: Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge: MIT Press, 1992.
- [Koz94] *Koza, J. R.*: Genetic Programming II: Automatic Discovery of Reusable Programs. Cambridge: MIT Press, 1994.

- [Koz99] *Koza, J. R., Bennett, F. H., Andre, D. and Keane, M. A.: Genetic Programming III: Darwinian Invention and Problem Solving.* San Francisco, CA: Morgan Kaufmann, 1999.
- [KS79] *Kreisselmeier, G. and Steinhäuser, R.: Systematische Auslegung von Reglern durch Optimierung eines vektoriellen Gütekriteriums.* In Regelungstechnik, 3, pp. 76-79, 1979.
- [Lan95] *Langermann:* Definition of a test function for contest on Evolutionary Computation. In [ICEC95], 1995.
- [Lit92] *Littger, K.: Optimierung – Eine Einführung in rechnergestützte Methoden und Anwendungen.* Berlin, Heidelberg: Springer-Verlag, 1992.
- [MBC95] *Marenbach, P., Bettenhausen, K.D. and Cuno, B.: Selbstorganisierende Generierung strukturierter Prozeßmodelle.* at-Automatisierungstechnik 6 (1995), pp. 277-288, Berlin, 1995.
- [MBF96] *Marenbach, P., Bettenhausen, K. D. and Freyer, S.: Signal path oriented approach to generation of dynamic process models,* in [GP96], pp. 327-332, 1996.
- [MW94] *MathWorks, The: Matlab – User Guide.* Natick, Mass.: The MathWorks, Inc., 1994. <http://www.mathworks.com/>
- [MHF92] *Mecklenburg, K., Hrycej, T., Franke, U. and Fritz, H.: Neural control of autonomous vehicle.* In IEEE Vehicular Technology Conference, Denver, 1992.
- [Men2] *Osmera, P.: MENDEL'96 – 2nd International Conference on Genetic Algorithms.* 26.-28. June 1996, Technical University of Brno, Czech Republic, 1996.
- [Mic92] *Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs.* Berlin, Heidelberg, New York: Springer-Verlag, 1992.
- [Mic94] *Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, Second, Extended Edition.* Berlin, Heidelberg, New York: Springer-Verlag, 1994.
- [Mie99] *Miettinen, K. M.: Nonlinear multiobjective optimization.* Boston, London, Dordrecht: Kluwer Academic Publishers, 1999.
- [Mit96] *Mitchell, M.: An Introduction to Genetic Algorithms.* Cambridge, Massachusetts: MIT Press, 1996.
- [Mit90] *Mitschke, M.: Dynamik der Kraftfahrzeuge: Band C, Fahrverhalten.* Berlin, Heidelberg, New York: Springer-Verlag, 1990.
- [Müh94] *Mühlenbein, H.: The Breeder Genetic Algorithm – a provable optimal search algorithm and its application.* Colloquium on Applications of Genetic Algorithms, IEE 94/067, London, 1994.
- [Müh95a] *Mühlenbein, H.: Adaptive Systeme in offenen Welten.* GMD-Spiegel 2/95, 1995. <http://borneo.gmd.de/AS/gmdsp/editorial.html>
- [Müh95b] *Mühlenbein, H.: Genetische Algorithmen und Evolutionsstrategien – Auf der Suche nach verschollenen Schätzten.* GMD-Spiegel 2/95, 1995. <http://borneo.gmd.de/AS/gmdsp/muehlen.html>
- [MSV93a] *Mühlenbein, H. and Schlierkamp-Voosen, D.: Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization.* Evolutionary Computation, 1 (1), pp. 25-49, 1993. [ftp://borneo.gmd.de/pub/as/ga/gmd\\_as\\_ga-93\\_01.ps](ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-93_01.ps)
- [MSV95] *Mühlenbein, H. and Schlierkamp-Voosen, D.: Analysis of Selection, Mutation and Recombination in Genetic Algorithms.* In Banzhaf, W. and Eeckman, F. H.: Evolution as a Computational Process. Lecture Notes in Computer Science 899, pp. 142-168, Berlin: Springer-Verlag, 1995. [ftp://borneo.gmd.de/pub/as/ga/gmd\\_as\\_ga-95\\_03.ps](ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-95_03.ps)
- [Nis97] *Nissen, V.: Einführung in evolutionäre Algorithmen: Optimierung nach dem Vorbild der Evolution.* Braunschweig, Wiesbaden: Vieweg, 1997.

- [Ost97] *Ostermeier, A.*: Schrittweitenadaption in der Evolutionsstrategie mit einem entstochastisierten Ansatz. Dissertation, Technische Universität Berlin, Fachbereich 6, 1997.
- [OGH93] *Ostermeier, A., Gawelczyk, A. and Hansen, N.*: A Derandomized Approach to Self Adaptation of Evolution Strategies. Technical Report TR-93-003, TU Berlin, 1993. <ftp://ftp-bionik.fb10.tu-berlin.de/pub/papers/Bionik/tr-03-93.ps.Z>
- [OGH94] *Ostermeier, A., Gawelczyk, A. and Hansen, N.*: Step-size adaptation based on non-local use of selection information. In [PPSN3], pp. 189-198, 1994.
- [Pad97] *Institut für angewandte Daten- und Wissenstechnik*: Prognose der minimalen Ausführungszeiten von Echtzeit-Systemen auf Basis von genetisch erzeugten Testfallmengen – Version 1.0. Institut für angewandte Daten- und Wissenstechnik AD/WT, Universität Paderborn, 1997.
- [Poh93] *Pohlheim, H.*: Simulation und Optimierung eines Blaualgen-Wachstums-Modells. Diplomarbeit, Technische Universität Ilmenau, 1993.
- [Poh95] *Pohlheim, H.*: Ein genetischer Algorithmus mit Mehrfachpopulationen zur Numerischen Optimierung. at-Automatisierungstechnik 3 (1995), pp. 127-135, 1995.
- [GEATbx] *Pohlheim, H.*: GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with Matlab. [www.geatbx.com](http://www.geatbx.com), 1995-1999. <http://www.geatbx.com/index.html>
- [Poh97] *Pohlheim, H.*: Advanced Techniques for the Visualization of Evolutionary Algorithms. Proceedings of 42. International Scientific Colloquium Ilmenau, Vol. 3, pp. 60-68, 1997. [http://www.pohlheim.com/papers\\_pohlheim.html](http://www.pohlheim.com/papers_pohlheim.html)
- [Poh98] *Pohlheim, H.*: Entwicklung und systemtechnische Anwendung Evolutionärer Algorithmen. Aachen, Germany: Shaker Verlag, 1998. (Development and Engineering Application of Evolutionary Algorithms) <http://www.pohlheim.com/diss/index.html>
- [Poh99] *Pohlheim, H.*: Visualization of Evolutionary Algorithms – set of standard techniques and multidimensional visualization. In [GEC99], p. 533-540, 1999. [http://www.pohlheim.com/papers\\_pohlheim.html](http://www.pohlheim.com/papers_pohlheim.html)
- [PH96a] *Pohlheim, H. und Heißner, A.*: Optimale Steuerung der Zustandsgrößen im Gewächshaus mit Genetischen Algorithmen: Grundlagen, Verfahren und Ergebnisse. Technischer Bericht, Technische Universität Ilmenau, 1996.
- [PH96b] *Pohlheim, H. and Heißner, A.*: Anwendung genetischer Algorithmen zur optimalen Steuerung des Gewächshausklimas. In GMA-Kongreß'96, VDI-Berichte 1282, pp. 799-809, Düsseldorf: VDI-Verlag, 1996.
- [PH97a] *Pohlheim, H. and Heißner, A.*: Optimal Control of Greenhouse Climate using a Short Time Greenhouse Climate Model and Evolutionary Algorithms. In Proceedings of 3<sup>rd</sup> IFAC/ISHS Workshop on „Mathematical and Control Applications in Agriculture & Horticulture“, pp. 113-118, 1997.
- [PM96] *Pohlheim, H. and Marenbach, P.*: Generation of structured process models using genetic programming. In [AISB96], pp. 102-109, 1996. [http://www.pohlheim.com/papers\\_pohlheim.html](http://www.pohlheim.com/papers_pohlheim.html)
- [PPW99] *Pohlheim, H., Pawletta, S. and Westphal, A.*: Parallel Evolutionary Optimization under Matlab on standard computing networks. In [GEC99], Evolutionary Computation and Parallel Processing Workshop, pp. 174-176, 1999. [http://www.pohlheim.com/papers\\_pohlheim.html](http://www.pohlheim.com/papers_pohlheim.html)
- [PS98] *Pohlheim, H. und Schütte, A.*: Optimierung der Parameter in einem Verbrennungsmodell für einen Dieselmotor mit Evolutionären Algorithmen. interner Technischer Bericht FT3/A-1998-001, Daimler Benz AG, 1998.

- [PW99] *Pohlheim, H. and Wegener, J.*: Testing the Temporal Behavior of Real-Time Software Modules using Extended Evolutionary Algorithms. In [GEC99], p. 1795, 1999.  
[http://www.pohlheim.com/papers\\_pohlheim.html](http://www.pohlheim.com/papers_pohlheim.html)
- [PL96] *Poli, R. and Logan, B.*: Evolutionary Computation Cookbook: Recipes for Designing New Algorithms. In Second Online Workshop on Evolutionary Computation, Japan, 1996.  
<http://www.bioele.nuee.nagoya-u.ac.jp/wec2/papers/index.html>
- [PPA93] *Puta, H., Pohlheim, H. and Affa, I.*: Simulation und Entscheidungshilfe für das Ökosystem Barther Bodden. In VDI-Berichte 1067, pp. 429-440, Düsseldorf: VDI-Verlag, 1993.
- [PPSN1] *Schwefel, H.-P. and Männer, R.*: Parallel Problem Solving from Nature – PPSN I. Vol. 496 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1991.
- [PPSN2] *Männer, R. and Manderick, B.*: Parallel Problem Solving from Nature, 2. Amsterdam: Elsevier Science Publishers, 1992.
- [PPSN3] *Davidor, Y., Schwefel, H.-P. and Männer, R.*: Parallel Problem Solving from Nature – PPSN III: International Conference on Evolutionary Computation. Vol. 866 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1994.
- [PPSN4] *Voigt, H.-M., Ebeling, W., Rechenberg, I. and Schwefel, H.-P.*: Parallel Problem Solving from Nature – PPSN IV: International Conference on Evolutionary Computation. Vol. 1141 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1996.
- [PPSN5] *Voigt, H.-M., Ebeling, W., Rechenberg, I. and Schwefel, H.-P.*: Parallel Problem Solving from Nature – PPSN V: International Conference on Evolutionary Computation. Vol. 1498 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1998.
- [Ray94] *Ray, T. S.*: An evolutionary approach to synthetic biology: Zen and the art of creating life. Artificial Life, 1 (1/2), pp. 195-226, 1994.
- [Rec73] *Rechenberg, I.*: Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart: Frommann-Holzboog, 1973.
- [Rec94] *Rechenberg, I.*: Evolutionsstrategie 94. Stuttgart: Frommann-Holzboog, 1994.
- [RB94] *Renders, J.-M. and Bersini, H.*: Hybridizing Genetic Algorithms with hill-climbing Methods for Global Optimization: Two Possible Ways. In [ICEC94] Vol. I, pp. 312-317, 1994.
- [Ric95] *Richter, G.*: Adaptive Systeme: Computer passen sich an. In GMD-Spiegel 2/95, 1995.  
<http://borneo.gmd.de/AS/gmdsp/richter.html>
- [RB93] *Riedmiller, M. and Braun, H.*: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *H. Ruspini (ed.)*: Proceedings of the IEEE International Conference on Neural Networks (ICNN), pp.586-591, 1993.
- [Sch68] *Schwefel, H.-P.*: Projekt MHD-Staurohr: Experimentelle Optimierung einer Zweiphasendüse, Teil I. Technischer Bericht 11.034/68, 35, AEG Forschungsinstitut, Berlin, 1968.
- [Sch75] *Schwefel, H.-P.*: Evolutionsstrategie und numerische Optimierung. Dissertation, Technische Universität Berlin, 1975.
- [Sch81] *Schwefel, H.-P.*: Numerical optimization of computer models. Chichester: Wiley & Sons, 1981.
- [Sch95] *Schwefel, H.-P.*: Evolution and optimum seeking. New York: John Wiley & Sons, 1995.

- [SK92] *Schwefel, H. P. and Kursawe, F.*: Künstliche Evolution als Modell für natürliche Intelligenz. In Nachtigall, W. (Ed.): Technische Biologie und Bionik 1, Proceedings 1. Bionik-Kongreß, BIONA report 8, Stuttgart: G. Fischer, pp. 73-91, 1992.
- [SHF94] *Schöneburg, E., Heinzmann, F. and Feddersen, S.*: Genetische Algorithmen und Evolutionsstrategien. Bonn, Paris, Reading, Mass.: Addison-Wesley, 1994.
- [SDJ91a] *Spears, W.M. and De Jong, K. A.*: On the Virtues of Parameterised Uniform Crossover. In [ICGA4], pp. 230-236, 1991.
- [SDJ91b] *Spears, W.M. and De Jong, K. A.*: An Analysis of Multi-Point Crossover. In [FGA1], pp. 301-315, 1991.
- [SD94] *Srinivas, N. and Deb, K.*: Multiobjective Optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3), pp. 221-248, 1994.
- [SE92] *Stuckmann, B. E. and Easom, E. E.*: A Comparison of Bayesian Sampling and Global Optimization Techniques. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 22, No. 5, pp.1024-1032, 1992.
- [Sum95] *Sumser, S.*: Verbrennungsmodell für direkteinspritzende Dieselmotoren. interner Technischer Bericht F1M/ST 95-0036, Daimler Benz AG, 1995.
- [SR96] *Surry, P. D. and Radcliffe, N. J.*: Innocation to Initialise Evolutionary Search. In [AISB96], pp. 260-275, 1996.
- [Sys89] *Syswerda, G.*: Uniform crossover in genetic algorithms. In [ICGA3], pp. 2-9, 1989.
- [TZ89] *Törn, A. and Zilinskas, A.*: Global Optimization. Vol. 350 of Lecture Notes in Computer Science, Berlin, Heidelberg, New York: Springer-Verlag, 1989.
- [Vel99] *Veldhuizen, D. A. van.*: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1999.  
<http://www.lania.mx/~ccoello/EMOO/veldhuizen99a.ps.gz>
- [Vös97] *Vössner, S.*: Convergence Measures for Genetic Algorithms. Technical Report, Stanford University, Department of EES, Operations Research, 1997.
- [VA94] *Voigt, H.-M. and Anheyer, T.*: Modal Mutations in Evolutionary Algorithms. In [ICEC94] Vol. I, pp. 88-92, 1994.
- [VMC95] *Voigt, H.-M., Mühlenbein, H. and Cvetkovi, D.*: Fuzzy recombination for the continuous Breeder Genetic Algorithm. In [ICGA6], pp. 104-111, 1995.  
[ftp://borneo.gmd.de/pub/as/ga/gmd\\_as\\_ga-95\\_01.ps](ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-95_01.ps)
- [Wei70] *Weinberg, R.*: Computer simulation of a living cell. Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 31(9), 5312B, University Microfilms No. 71-4766, 1970.
- [Why89] *Whitley, D.*: The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In [ICGA3], pp. 116-121, 1989.
- [WM95] *Wolpert, D. H. and Macready, W. G.*: No free lunch theorems for search. Technical report SFI-TR-95-02-010, The Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM, 87501, USA, 1995.  
<http://www.santafe.edu/sfi/publications/Working-Papers/95-02-010.ps>
- [Wri91] *Wright, A. H.*: Genetic Algorithms for Real Parameter Optimization. In [FGA1], pp. 205-218, 1991.
- [ZT98] *Zitzler, E. and Thiele, L.*: An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 1998.  
<ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/ZT1998a.ps>
- [ZDT99] *Zitzler, E., Deb, K. and Thiele, L.*: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Technical Report 70, Computer Engineering and

Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH)  
 Zurich, 1999.  
<ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/ZDT1999.ps>

## Kombinatorische Optimierung (TSP, Scheduling)

- [AV97] *Aarts, E. and Verhoeven, M.*: Evolutionary Computation in Practice: Genetic local search for the traveling salesman problem. In [HEC97], G9.5, pp. G9.5:1-G9.5:7, 1997.
- [AM96] *Asveren, T. and Molitor, P.*: New Crossover Methods For Sequencing Problems. In [PPSN4], pp. 290-299, 1996.
- [BUM91] *Bagchi, S., Uckun, S., Miyabe, Y. and Kawamura, K.*: Exploring problem-specific recombination operators for job shop scheduling. In [ICGA4], pp. 10-17, 1991.
- [BMK96] *Bierwirth, C., Mattfeld, D. C. and Kopfer, H.*: On Permutation Representations for Scheduling Problems. In [PPSN4], pp. 310-318, 1996.
- [Bru97] *Bruns, R.*: Evolutionary Computation Applications: Scheduling. In [HEC97], F1.5, pp. F1.5:1-F1.5:9, 1997.
- [Dav85] *Davis, L.*: Applying adaptive algorithms to epistatic domains. In Proc. Int. Joint Conf. on Artificial Intelligence, 1985.
- [DW94] *Dzubera, J. and Whitley, D.*: Advanced Correlation Analysis of Operators for the Traveling Salesman Problem. In [PPSN3], pp. 68-77, 1994.
- [GL85] *Goldberg, D. and Lingle, R.*: Alleles, loci, and the traveling salesman problem. In [ICGA1], 1985.
- [GS89] s. S.292
- [GS97a] *Gorges-Schleuter, M.*: Asparagos96 and the Traveling Salesman Problem. In [ICEC97], pp. 171-174, 1997.
- [GS97b] *Gorges-Schleuter, M.*: On the power of evolutionary optimization at the example of ATSP and large TSP Problems. In European Conference on Artificial Life '97, Brighton, U.K., 1997.
- [LK73] *Lin, S. and Kernighan, B.*: An efficient heuristic procedure for the traveling salesman problem. Operations Res. 21, pp. 498-516, 1973.
- [MW92] *Mathias, K. and Whitley, D.*: Genetic operators, the fitness landscape and the traveling salesman problem. In [PPSN2], pp. 219-228, 1992.
- [Mat96] *Mattfeld, D. C.*: Evolutionary Search and the Job Shop: Investigations on Genetic Algorithms for Production Scheduling. Heidelberg: Physica-Verlag, 1996.
- [Müh89] s. S.293
- [MGK88] s. S.293
- [NK97] *Nagata, Y. and Kobayashi, S.*: Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In [ICGA7], pp. 450-457, 1997.
- [Nis97] *Nissen, V.*: Evolutionary Computation in Practice: Quadratic assignment. In [HEC97], G9.10, pp. G9.10:1-G9.10:8, 1997.
- [OSH87] *Oliver, I. M., Smith, D. J. and Holland, J. R. C.*: A study of permutation crossover operators on the traveling salesman problem. In [ICGA2], pp. 224-230, 1987.
- [Ron95] *Ronald, S.*: Routing and scheduling problems. In [Cha95], pp. 367-430, 1995.
- [SMW91] *Starkweather, T., McDaniel, S., Mathias, K., Whitley, D. and Whitley, C.*: A Comparison of Genetic Sequencing Operators. In [ICGA4], pp. 69-76, 1991.
- [Sys91] *Syswerda, G.*: Schedule Optimization Using Genetic Algorithms. In [Dav91], pp. 332-349, 1991.

- [TM98] *Tao, G. and Michalewicz, Z.*: Inver-over Operator for the TSP. In [PPSN5], pp. 803-812, 1998.
- [TSPBIB] *Moscato, P.*: TSPBIB Home page. 1996.  
[http://www.densis.fee.unicamp.br/~moscato/TSPBIB\\_home.html](http://www.densis.fee.unicamp.br/~moscato/TSPBIB_home.html)
- [TSPLIB] TSPLIB – A Traveling Salesman Problem Library. 1995.  
<http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html>
- [WRE98] *Watson, J. P., Ross, C., Eisele, V., Denton, J., Bins, J., Guerra, C., Whitley, D. and Howe, A.*: The Traveling Salesrep Problem, Edge Assembly Crossover, and 2-opt. In [PPSN5], pp. 823-832, 1998.  
<http://www.cs.colostate.edu/~genitor/1998/ppsn98b.ps.gz>
- [WSF89] *Whitley, D., Starkweather, T. and Fuquay, D.*: Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator. In [ICGA3], pp. 133-140, 1989.
- [WSS91] *Whitley, D., Starkweather, T. and Shaner, D.*: Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination. In [Dav91], pp. 350-372, 1991.
- [WY95] *Whitley, D. and Yoo, N.-W.*: Modeling simple genetic algorithms for permutation problems. In [FGA3], pp. 163-184, 1995.
- [Whi97a] *Whitley, D.*: Search Operators: Mutation: Permutations. In [HEC97], C3.2.3, pp. C3.2:5-C3.2:7, 1997.
- [Whi97b] *Whitley, D.*: Search Operators: Recombination: Permutations. In [HEC97], C3.3.3, pp. C3.3:14-C3.3:20, 1997.
- [Whi99] *Whitley, D.*: A free lunch proof for gray versus binary encodings. In [GEC99], pp. 726-733, 1999.

## Behandlung von Populationen – Parallele Modelle

- [Bel95] *Belding, T. C.*: The Distributed Genetic Algorithm Revisited. In [ICGA6], pp.114-121, 1995.
- [Can95] *Cantú-Paz, E.*: A Summary of Research on Parallel Genetic Algorithms. Technical Report IlliGAL No. 95007, July 1995, University of Illinois at Urbana-Champaign, 1995.  
<ftp://ftp.illigal.ge.uiuc.edu/pub/papers/IlliGALs/95007.ps.Z>
- [CJ91] *Collins, R. J. and Jefferson, D. R.*: Selection in Massively Parallel Genetic Algorithms. In [ICGA4], pp. 249-256, 1991.
- [CPR96] *Corno, F., Prinetto, P., Rebaudengo, M. and Reorda, M. S.*: Exploiting Competing Subpopulations for Automatic Generation of Test Sequences for Digital Circuits. In [PPSN4], pp. 792-800, 1996.
- [DJS95] *DeJong, K. and Sarma, J.*: On Decentralizing Selection Algorithms. In [ICGA6], pp. 17-23, 1995.
- [FH91] *Fogarty, T. C. and Huang, R.*: Implementing the Genetic Algorithm on Transputer Based Parallel Processing Systems. In [PPSN1], pp. 145-149, 1991.
- [GW91] *Gordon, V. S. and Whitley, D.*: Serial and Parallel Genetic Algorithms as Function Optimizers. In [ICGA5], pp. 177-183, 1993.
- [GS89] *Gorges-Schleuter, M.*: ASPARAGOUS An Asynchronous Parallel Genetic Optimization Strategy. In [ICGA3], pp. 422-427, 1989.
- [GS91] *Gorges-Schleuter, M.*: Explicit Parallelism of Genetic Algorithms through Population Structures. In [PPSN1], pp. 150-159, 1991.

- [GS98] *Gorges-Schleuter, M.*: A Comparative Study of Global and Local Selection in Evolution Strategies. In [PPSN5], pp. 367-377, 1998.
- [HM94] *Hauser, R. and Männer, R.*: Implementation of Standard Genetic Algorithms on MIMD Machines. In [PPSN3], pp. 504-513, 1994.
- [Her92] *Herdy, M.*: Reproductive Isolation as Strategy Parameter in Hierarchical Organized Evolution Strategies. In [PPSN2], pp. 207-217, 1992.
- [KSR94] *Kapsalis, A., Smith, G.D. and Rayward-Smith, V.J.*: A unified parallel genetic algorithm. AISB Workshop Evolutionary Computation, April 11-13, Leeds, 1994.
- [Loh91] *Lohmann, R.*: Application of Evolution Strategy in Parallel Populations. In [PPSN1], pp. 198-208, 1991.
- [MS89] *Manderick, B. and Spiessens, P.*: Fine-grained Parallel Genetic Algorithms. In [ICGA3], pp. 428-433, 1989.
- [Müh89] *Mühlenbein, H.*: Parallel genetic algorithms, population genetics and combinatorial optimization. In [ICGA3], pp. 416-421, 1989.
- [Müh91] *Mühlenbein, H.*: Evolution in Time and Space – The Parallel Genetic Algorithm. In [FGA1], pp. 316-337, 1991.  
[ftp://borneo.gmd.de/pub/as/ga/gmd\\_as\\_ga-91\\_01.ps](ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-91_01.ps)
- [MGK88] *Mühlenbein, H., Gorges-Schleuter, M. and Krämer, O.*: Evolution algorithms in combinatorial optimization. Parallel Computing, 7, pp.65-85, 1988.
- [MSB91] *Mühlenbein, H., Schomisch, M. and Born, J.*: The parallel genetic algorithm as function optimizer. Parallel Computing, 17, pp.619-632, 1991.
- [MSV93a] s. S.287
- [PLG87] *Pettey, C. B., Leuze, M. R. and Grefenstette, J. J.*: A Parallel Genetic Algorithm. In [ICGA2], pp. 155-161, 1987.
- [Rob87] *Robertson, G. G.*: Parallel Implementation of Genetic Algorithms in a Classifier System. In [ICGA2], pp. 140-147, 1987.
- [Rud91] *Rudolph, G.*: Global Optimization by Means of Distributed Evolution Strategies. In [PPSN1], pp. 209-213, 1991.
- [SDJ96] *Sarma, J. and DeJong, K.*: An Analysis of the Effects of Neighbourhood Size and Shape on Local Selection Algorithms. In [PPSN4], pp. 236-244, 1996.
- [Swm96] *Schwehm, M.*: Globale Optimierung mit massiv parallelen genetischen Algorithmen. Dissertation, Universität Erlangen-Nürnberg, 1996.  
<http://www7.informatik.uni-erlangen.de/~schwehm/MYpapers/Dissertation.html>
- [SM91] *Spiessens, P. and Manderick, B.*: A Massively Parallel Genetic Algorithms – Implementation and First Analysis. In [ICGA4], pp. 279-286, 1991.
- [SVM94] *Schlirkamp-Voosen, D. and Mühlenbein, H.*: Strategy adaptation by competing subpopulations. In [PPSN3], pp. 199-208, 1994.  
[ftp://borneo.gmd.de/pub/as/ga/gmd\\_as\\_ga-94\\_14.ps](ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-94_14.ps)
- [SVM96] *Schlirkamp-Voosen, D. and Mühlenbein, H.*: Adaptation of Population Sizes by Competing Subpopulations. In [ICEC96], pp. 330-335, 1996.  
[ftp://borneo.gmd.de/pub/as/ga/gmd\\_as\\_ga-96\\_01.ps](ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-96_01.ps)
- [SWM91] *Starkweather, T., Whitley, D. and Mathias, K.*: Optimization using Distributed Genetic Algorithms. In [PPSN1], pp. 176-185, 1991.
- [Tan87] *Tanese, R.*: Parallel Genetic Algorithm for a Hypercube. In [ICGA2], pp. 177-183, 1987.
- [Tan89] *Tanese, R.*: Distributed Genetic Algorithms. In [ICGA3], pp. 434-439, 1989.
- [VBS91] *Voigt, H.-M., Born, J. and Santibanez-Koref, I.*: Modeling and Simulation of Distributed Evolutionary Search Processes for Function Optimization. In [PPSN1], pp. 373-380, 1991.
- [VSB92] *Voigt, H.-M., Santibanez-Koref, I. and Born, J.*: Hierarchically Structured Distributed Genetic Algorithm. In [PPSN2], pp. 145-154, 1992.

## Visualisierung

- [CB96] *Beardah, C. C. and Baxter, M.*: The archaeological use of Kernel Density Estimates. Internet Archaeology 1, 5.1, 1996.  
[http://intarch.ac.uk/journal/issue1/beardah\\_toc.html](http://intarch.ac.uk/journal/issue1/beardah_toc.html)
- [Col93] *Collins, T. D.*: The Visualisation of Genetic Algorithms. Msc. Thesis, De Montfort University, Leicester, GB, 1993.
- [Col95] *Collins, T. D.*: The Visualization of Genetic Algorithms – Related Work. Technical Report, KMI-TR-19, Knowledge Media Institute, The Open University, Milton Keynes, UK, 1995.  
<http://kmi.open.ac.uk/kmi-abstracts/kmi-tr-19-abstract.html>
- [Col97a] *Collins, T. D.*: Genotypic-Space Mapping: Population Visualization for Genetic Algorithms. Technical Report, KMI-TR-39, Knowledge Media Institure, The Open University, Milton Keynes, UK, 1997.  
<http://kmi.open.ac.uk/~trevor/research/publications/kmi-tr-39.ps.gz>
- [Col97b] *Collins, T. D.*: Using Software Visualization technology to help Genetic Algorithms Designers. In Proceedings of The Ninth Annual Conference of the Psychology of Programming Interest Group (PPIG 9), pp. 43-51, 1997.  
<http://kmi.open.ac.uk/~trevor/research/publications/PPIG-97.ps.gz>
- [CC94] *Cox, T. F. and Cox, M. A. A.*: Multidimensional Scaling. London: Chapman & Hall, 1994.
- [DH73] *Duda, R. O. and Hart, P. E.*: Pattern Classification and Scene Analysis. New York: John Wiley & Sons, 1973.
- [DCW96] *Dybowski, R., Collins, T. D. and Weller, P. D.*: Visualization of binary string convergence by Sammon mapping. In [EP96], pp. 377-383, 1996.  
<http://kmi.open.ac.uk/~trevor/research/publications/EP96.ps.gz>
- [JF95] s. S.286
- [NEA94] *Nassersharif, B., Ence, D. and Au, M.*: Visualization of Evolution of Genetic Algorithms, In Proceedings of World Congress on Neural Networks WCNN'94, San Diego, CA, USA, Hillside, NJ, USA: Lawrence Erlbaum Associates, pp. 1/560-1/565, 1994.
- [Poh97] s. S.288
- [Poh99] s. S.288
- [Rip96] *Ripley, B. D.*: Pattern Recognition and Neural Networks. Cambridge, GB: Cambridge University Press, 1996.
- [RC93] *Routen, T. W. and Collins, T. D.*: The Visualisation of AI Techniques. In Proceedings of Third International Conference on Computational Graphics and Visualisation Techniques COMPUGRAPH'93, Alvor, Portugal, New York, USA: ACM Press, pp. 274-282, 1993
- [Rou94] *Routen, T. W.*: Techniques for the Visualisation of Genetic Algorithms. In [ICEC94] Vol. II, pp. 846-851, 1994.
- [Sam69] *Sammon, J. W. jr.*: A Nonlinear Mapping for Data Structure Analysis. IEEE Transactions on Computers, Vol. C-18, no. 5, pp. 401-409, 1969.
- [Swm96] s. S.293

## Polyploidie bei Evolutionären Algorithmen

- [Bag67] *Bagley, J. D.*: The behavior of adaptive systems which employ genetic and correlation algorithms. Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 28(12), 5106B, University Microfilms No. 68-7556, 1967.
- [Bri81] *Brindle, A.*: Genetic algorithms for function optimization. unpublished doctoral dissertation, University of Alberta: Edmonton, 1981.
- [CCR96] *Collingwood, E., Corne, D. and Ross, P.*: Useful Diversity via Multiploidy. In [AISB96], Workshop Proceedings, pp. 49-53, 1996.
- [GS87] *Goldberg, D. E. and Smith, R. E.*: Nonstationary function optimization using genetic algorithms with dominance and diploidy. In [ICGA2], pp. 59-68, 1987.
- [Gol89] s. S.285
- [Hol71] *Hollstien, R. B.*: Artificial genetic adaptation in computer control systems. Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 32(3), 1510B, University Microfilms No. 71-23,773, 1971.
- [NW95] *Ng, K. P. and Wong, K. C.*: A New Diploid Scheme and Dominance Change Mechanism for Non-Stationary Function Optimization. In [ICGA6], pp. 159-166, 1995.
- [Ros67] *Rosenberg, R. S.*: Simulation of genetic populations with biochemical properties. Doctoral dissertation, University of Michigan, Dissertation Abstracts International, 28(7), 2732B, University Microfilms No. 67-17,836, 1967.
- [Rya96] *Ryan, C.*: Reducing Premature Convergence in Evolutionary Algorithms. Ph.D. thesis, University College Cork, Ireland, 1996.  
<ftp://odyssey.ucc.ie/pub/genetic/thesis.ps.Z>
- [Smi87] *Smith, R. E.*: Diploid genetic algorithms for search in time varying environments. Proceedings of the 25th Annual Southeast Regional Conference of the ACM, pp. 175-178, 1987.
- [Smi88] *Smith, R. E.*: An investigation of diploid genetic algorithms for adaptive search of nonstationary functions. Unpublished master's thesis, University of Alabama: Tuscaloosa, 1988.
- [YA94] *Yoshida, Y. and Adachi, N.*: A Diploid Genetic Algorithm for Preserving Population Diversity – pseudo-Meiosis GA. In [PPSN3], pp. 36-45, 1994.

## Biologie, Genetik und Populationsgenetik

- [CK70] *Crow, J. F. and Kimura, M.*: An Introduction to Population Genetics Theory. New York: Harper and Row, 1970.
- [Dar1859] *Darwin, C.*: On the origin of species by means of natural selection. London: Murray, 1859.  
(Deutsche Übersetzung: Die Entstehung der Arten durch natürliche Zuchtwahl. 1860. u.a. Stuttgart: Reclam, 1974.)
- [Fwb63] Fremdwörterbuch. Leipzig: Bibliographisches Institut, 1963.
- [Hag91] *Hagemann, R.*: Allgemeine Genetik. Jena: Gustav Fischer Verlag, 1991.
- [Ode97] *Odenbach, W.*: Biologische Grundlagen der Pflanzenzüchtung. Berlin: Parey Buchverlag, 1997.
- [Hen95] *Hennig, W.*: Genetik. Berlin, Heidelberg: Springer-Verlag, 1995.
- [RM54] *Rieger, R. and Michaelis, A.*: Genetisches und Cytogenetisches Wörterbuch. Der Züchter, 2. Sonderheft, Berlin, Göttingen, Heidelberg: Springer-Verlag, 1954.

- [Wer68] *Werner, F.*: Wortelemente lateinisch-griechischer Fachausdrücke in den biologischen Wissenschaften. Halle (Saale): Max Niemeyer Verlag, 1968.

## Pflanzenwachstum und Gewächshaus

- [Als93] *Alischer, G.*: Optimierung der CO<sub>2</sub>- und Temperaturregelung bei Gewächshauskulturnen mit Hilfe von Modellansätzen unterschiedlicher Abstraktionsebenen. Dissertation, Universität Hannover, 1993.
- [Arn87] *Arnold, E.*: Zur optimalen Steuerung zeitdiskreter dynamischer Prozesse mittels nichtlinearer Optimierung mit Anwendungen auf die Klimasteuerung von Gewächshäusern. Dissertation, Technische Hochschule Ilmenau, 1987.
- [BC94] *Bailey, B. J. and Chalabi, Z. S.*: Improving the cost effectiveness of greenhouse climate control. In Computers and Electronics in Agriculture 10 (1994), pp. 203-214, 1994.
- [CBW96] *Chalabi, Z. S., Bailey, B. J. and Wilkinson, D. J.*: A real-time optimal control algorithm for greenhouse heating. In Computers and Electronics in Agriculture 15 (1996), pp. 1-13, 1996.
- [DY96] *Day, W. and Young, P. C.*: Proceedings of the Second IFAC/ISHS Workshop on Mathematical and Control Applications in Agriculture and Horticulture. Acta Horticulturae, 406, 1996.
- [Hei96] *Heißner, A.*: Ein einfaches Gewächshausklimamodell für die Kurzfriststeuerung von Temperatur, Luftfeuchte und CO<sub>2</sub>-Konzentration. Gartenbauwissenschaft, 61 (6), pp. 289-300, 1996.
- [Hei97] *Heißner, A.*: Der CO<sub>2</sub>-Gaswechsel von Paprikapflanzen in Abhängigkeit von der Bestrahlungsstärke, der CO<sub>2</sub>-Konzentration, der Lufttemperatur und dem Dampfdrucksättigungsdefizit der Luft: Messungen und Modell. Gartenbauwissenschaft, 62 (2), pp. 78-90, 1997.
- [JHS95] *Jones, J. W., Hwang, Y. K. and Seginer, I.*: Simulation of Greenhouse Crops, Environments and Control Systems. Acta Horticulturae, 399, pp. 73-84, 1995.
- [Mar90] *Markert, A.*: Aggregation pflanzenphysiologischer Wachstumsmodelle und Berechnung von Steuerstrategien für das Gewächshausinnenklima mittels Verfahren der nichtlinearen Optimierung. Dissertation, Technische Hochschule Ilmenau, 1990.
- [Mat95] *Matitschka, G.*: Vereinfachende Beschreibung der Bruttophotosynthese des physiologisch-dynamischen Simulationsmodells SUCROS und Weiterentwicklung für spezielle Anwendungen im Gemüsebau (Produktionssysteme Kohlrabi und Kopfsalat). Dissertation, Universität Hohenheim, Stuttgart: Verlag Ulrich E. Grauer, 1995.
- [PH96a] s. S.288
- [PH96b] s. S.288
- [PH97a] s. S.288
- [Sch85] *Schmidt, M.*: Bestimmung optimaler klimatischer Wachstumsfaktoren von Gewächshauskulturen auf der Basis pflanzenphysiologischer Beschreibungsmodelle. Dissertation, Technische Hochschule Ilmenau, 1985.
- [Seg96] *Seginer, I.*: Optimal control of the greenhouse environment. In [DY96], pp. 191-202, 1996.
- [vSC95] *van Straten, G. and Challa, H.*: Greenhouse climate control systems. In *Bakker, J. C., Bot, G. P. A., Challa, H. and Van de Braak, N. J. (ed.)*: Greenhouse climate control – an integrated approach. Wageningen: Wageningen Press, 1995.

# Glossar

## Allel

- *allellos* (gr.) – einander, gegenseitig, wechselseitig
- Zustandsformen eines Gens (bei diploiden oder polyploiden Organismen)  
  
(bei haploiden Organismen sind Allel und Gen identisch)  
siehe auch: Dominanz, rezessiv

## bias

- (engl.) schräg, schiefl, Ablenkung nach einer Seite, Bevorzugung (einer Variante)
- (EA) absolute Differenz zwischen der normalisierten Fitneß eines Individuums und seiner erwarteten Reproduktionswahrscheinlichkeit,  
Parameter bei der Bewertung von Selektionsverfahren  
siehe auch: Selektion, *spread*

## binär-kodiert

siehe: Kodierung

## Chromosom

- *chromos* (gr.) – Farbe, *soma* (gr.) – Körper; färbbare Körper
- (biol.) Träger der im Zellkern enthaltenen Erbinformation eines Lebewesens  
siehe auch: Diploidie

## Chromosomensatz

- (biol.) Anzahl der im haploiden Zustand vorhandenen Chromosomen  
(Grundzahl, Basiszahl)  
siehe auch: Chromosom

## crossover

- *crossing-over* (engl.) – überkreuzen
- (biol.) Mechanismus, der zum Austausch von Genen zwischen homologen Chromosomen (in der Prophase der Meiose) führt
- (EA) Synonym für Rekombination (vor allem im amerikanischen Sprachraum benutzt)  
siehe auch: Rekombination

## crowding

- (engl.) wimmeln, zusammengedrängt sein, gefüllt sein

- (EA) der Selektionsdruck auf viele ähnlichen Individuen ist größer als auf Individuen, die sich voneinander stärker unterscheiden; Kampf um begrenzte Ressourcen – Individuen in weniger bevölkerten Nischen haben eine höhere Lebens- und Fortpflanzungswahrscheinlichkeit, als Individuen in überfüllten Nischen  
siehe auch: *niching, sharing*

**deme**

- *demos* (gr.) – Volk
- (biol.) letzte natürliche Einheit in sich sexuell fortlaufenden Arten, die aus taxonomisch eng verwandten Individuen besteht; lokale Populationen, in denen Panmixie herrscht
- (EA) Unterpopulation; Fortpflanzungsgemeinschaft (Selektionspool)  
siehe auch: Unterpopulation, Selektionspool

**diploid / Diploidie**

- *diploos* (gr.) – zweifach, doppelt
- (biol.) doppelter Chromosomensatz in der Zelle; normaler genetischer Zustand höherer Organismen  
siehe auch: haploid, polyploid

**diversity**

- (engl.) Vielfalt, Verschiedenheit, Mannigfaltigkeit; *versus* (lat.) – verschieden
- Maß für die Verschiedenheit bzw. Mannigfaltigkeit von Individuen  
siehe auch: *loss of diversity*

**Domain (einer Variable)**

- (engl.) Gebiet
- (EA) Definitionsbereich einer Variablen;  
Bereich aller möglichen Werte, die eine Variable annehmen kann, wird oft durch eine untere und eine obere Schranke begrenzt

**Dominanz**

- *dominare* (lat.) herrschen, vorherrschen
- (biol.) ein dominantes Allel hat Vorrang über alle anderen Allele (die rezessiven Allele), das dominante Allel bestimmt die Merkmalsausprägung  
siehe auch: rezessiv

**Duplikation**

- (engl.) Verdopplung
- (EA) evolutionärer Operator  
siehe auch: Inversion, Mutation, Rekombination

**gekoppelte Variablen**

siehe: korrelierte Variablen

**gray-kodiert**

siehe: Kodierung

**elitism**

- (engl.) Auswahl der Elite (der Besten)
- (EA) das beste Individuum/die besten Individuen der alten Generation überleben in die neue Generation → elitist Selektion  
siehe auch: Selektion, *generation gap*

**epistasis**

- (biol.) Interaktion zwischen Genen  
siehe auch: korrelierte Variablen

**Fitneß**

- Leistungsfähigkeit, Eignung
- (biol.) Fortpflanzungswahrscheinlichkeit eines Individuums im Vergleich zu anderen Individuen in demselben Lebensraum
- (EA) Fortpflanzungswahrscheinlichkeit eines Individuums im Vergleich zu allen anderen Individuen in demselben Selektionspool  
siehe auch: Zielfunktion, Zielfunktionswert

**Generation**

- (biol.) aufeinanderfolgende Nachkommenschaften
- (EA) zwei Bedeutungen werden verwendet: aufeinanderfolgende Nachkommenschaften und Zeit zwischen aufeinanderfolgenden Nachkommen

**generation gap**

- (engl.) Generationslücke
- (EA) Verhältnis zwischen der Anzahl der Nachkommen und der Anzahl der Individuen in der Elternpopulation; (0,9: (0,9 · Anzahl der Eltern) Nachkommen werden gebildet; 3,0: es werden drei mal so viele Nachkommen gebildet, wie Eltern in der Population sind)

**Gen**

- (gr.)
- (biol.) an spezifischer Stelle eines Chromosoms lokalisierbare Erbanlage  
siehe auch: Allel

**Genotyp**

- (biol.) Gesamtheit der Erbinformation im Zellkern, Teil des Idiotyps  
Idiotyp: Gesamtheit der Erbinformation einer Zelle,  
Plasmotyp: Gesamtheit der Erbinformation im Zytoplasma (alles außerhalb des Zellkerns)  
siehe auch: Phänotyp

**haploid / Haploidie**

- *haploous* (gr.) – einzeln, einfach
- (biol.) einfacher Chromosomensatz in der Zelle, normaler genetischer Zustand von Prokaryoten und von eukaryotischen Keimzellen nach der Meiose.  
siehe auch: diploid, polyploid

**heterozygous**

- *hetero* (gr.) – verschieden, ungleich, *zygous* (gr.) – befruchtete Eizelle; ungleicherbig, gemischterbig
- (biol.) diploide bzw. polyploide Zellen, in denen mindestens ein Allel unterschiedlich ist  
siehe auch: homozygous, diploid, polyploid

**homozygous**

- *homo* (gr.) – gleich, *zygous* (gr.) – befruchtete Eizelle; reinerbig
- (biol.) beide bzw. alle Chromosomensätze sind gleich  
siehe auch: heterozygous, diploid, polyploid

**Individuum**

- *individuus* (lat.) – losgelöst, selbständige, unteilbare ein einzelnes Wesen aus einer Gruppe oder Art, das in sich abgeschlossen ist
- (biol.) Tier: geschlossenes System, Pflanze: offenes System
- (EA) ein Individuum stellt jeweils eine Lösung für das bearbeitete Problem dar (ein Punkt im Suchraum); ein Individuum besteht aus Variablen, mehrere Individuen bilden eine (Unter-)Population  
siehe auch: Variable

**Inversion**

- (engl.) Umkehrung
- (EA) evolutionärer Operator  
siehe auch: Duplikation, Mutation, Rekombination

**Kodierung**

- Art der Darstellung der Information
- (EA) unterschieden wird zwischen: 1. Darstellung im Format der Variablen des Problems und 2. Umsetzung der Variablen in ein anderes Format, z.B. reelle oder ganzzahlige Variablen werden als eine Kette von binären Zahlen dargestellt
- binäre Kodierung: binär – zweiwertig  
(EA) Darstellung der Variablen eines Individuums durch binäre Zahlen; ganzzahlige Zahlen lassen sich eindeutig durch binäre Zahlen darstellen, reelle Zahlen müssen diskretisiert werden, bevor sie durch binäre Zahlen dargestellt werden können
- GRAY-Kodierung (nach FRANK GRAY):  
(EA) Darstellung der Variablen eines Individuums durch binäre Zahlen, wobei benachbarte Werte der Variablen durch solche binäre Zahlen kodiert werden, die sich in ihrer Repräsentation nur in einer Bitposition unterscheiden. Wenn die Werte der Variablen schrittweise verändert werden, so verändert sich dadurch in jedem Schritt die entsprechende binäre Zahl nur an einer Stelle. Ganzzahlige Zahlen lassen sich direkt mittels eines

GRAY-Kodes in binäre Zahlen umsetzen, reelle Zahlen müssen vorher diskretisiert werden.

### **korrelierte Variablen**

- *correlatio* (lat.) – Wechselbeziehung
- auch: *epistasis* oder gekoppelte Variablen (*coupled variables*)
- (EA) Beim Vorliegen korrelierter Variablen kann durch die Veränderung nur einer Variablen keine Verbesserung der Zielfunktion erreicht werden, es müssen alle korrelierten Variablen zusammen und in die richtige Richtung verändert werden. Die Stärke dieser Kopplung/Korrelation zwischen den Variablen entscheidet über die Schwierigkeit der Zielfunktion.
- Die Variablen eines Systems sind unkorreliert, wenn das System separabel ist (keine Wechselwirkung zwischen den Variablen), ansonsten liegt eine Korrelation zwischen den Variablen vor. Die Stärke der Korrelation hängt von der Wechselwirkung der Variablen untereinander ab und lässt sich nicht allgemein beschreiben. Im Rahmen der Optimierung kann man sich den Weg zum Optimum bei stark korrelierten Variablen als ein schmales und gebogenes Tal vorstellen. Ein klassisches Beispiel eines solchen Systems ist ROSEN BROCK's Funktion, auch Bananenfunktion genannt.

### **Lösungsraum**

- Menge der zulässigen Lösungen eines Optimierungsproblems

### **loss of diversity**

- (engl.) Verlust an Vielfalt/Mannigfaltigkeit/Diversität
- (EA) Anteil der Individuen des Selektionspools, die während einer Selektionsphase nicht ausgewählt werden (die Individuen/Information, die bei einer vollständigen Ersetzung der Elternpopulation durch die Nachkommenpopulation verloren gehen würden); Parameter bei der Bewertung von Selektionsverfahren

siehe auch: *diversity*

### **Migration**

- *migrare* (lat.) – wandern, fortziehen, auswandern
- (biol.) Austausch von Individuen zwischen Populationen
- (EA) Wanderung bzw. Bewegung von Individuen von einer Unterpopulation zu anderen Unterpopulationen

### **Mutation**

- *mutare* (lat.) – springen
- (biol.) sprunghaft auftretende Veränderung der Information eines Individuums, meist zufällig oder mit zufälligen Elementen
- (EA) evolutionärer Operator (z.B. Umschalten einer binären Zahl, Addition einer Zufallszahl zu einer diskreten oder reellen Zahl, Austausch von Teilbäumen bei Strukturen, Austausch von Variablen in Reihenfolgeproblemen)

siehe auch: Rekombination

**niching**

- *niche* (engl.) – Nische
- (EA) Ausbildung von Nischen (Gruppen voneinander räumlich getrennter Individuen) durch die Arbeit des Evolutionären Algorithmus, z.B. *sharing* und mehrkriterielles Ranking  
siehe auch: *sharing*

**offline Performanz**

- (EA) Maß für den Fortschritt einer Population durch Bewertung der Güte des besten Individuums einer Population  
siehe auch: online Performanz

**online Performanz**

- (EA) Maß für den Fortschritt einer Population durch Bewertung der durchschnittlichen Güte einer Population  
siehe auch: offline Performanz

**panmictic**

- (engl.) uneingeschränkt  
*Panmixie* (gr.) – Mischung
- (biol.) Bezeichnung für die zufällige Paarung von zwei verschiedengeschlechtlichen Individuen in einer Population. In sehr großen Populationen haben zwei beliebige verschiedengeschlechtliche Individuen die gleiche Paarungswahrscheinlichkeit.
- (EA) panmictische Population: keine Beschränkungen innerhalb der Population für die Selektion von Individuen (im Gegensatz zu einer in Unterpopulationen unterteilten Population bzw. einer Population, in der eine lokale Selektion stattfindet)  
siehe auch: Selektion

**PARETO-Dominanz**

- (EA) Ein Vektor  $a = (a_1, \dots, a_n)$  dominiert einen Vektor  $b = (b_1, \dots, b_n)$  dann und nur dann wenn gilt:  

$$\forall i \in \{1, \dots, n\}, a_i \leq b_i \quad \wedge \quad \exists i \in \{1, \dots, n\}, a_i < b_i.$$

**PARETO-Optimalität**

- (EA) Ein Individuum  $A$  ist dann PARETO-optimal, wenn es kein anderes Individuum  $B$  gibt, durch dessen Kriterienvektor,  $b = f(B) = (b_1, \dots, b_n)$ , der Kriterienvektor des Individuums  $A$ ,  $a = f(A) = (a_1, \dots, a_n)$ , dominiert wird.  
siehe auch: PARETO-Dominanz

**Phänotyp**

- *phen* (gr.) – Merkmal, Erscheinung, das Erscheinende
- Gesamtheit der Merkmale eines Individuums  
siehe auch: Genotyp

**Permutation**

- Vertauschung

- eine Permutation von  $(1, 2, \dots, n)$  ist ein  $n$ -dimensionaler Vektor, dessen Elemente höchstens durch Vertauschungen von  $(1, 2, \dots, n)$  entstehen

### **polyploid /Polyplodie**

- *polyo* (gr.) – viel, *polyplous* (gr.) – vielfältig
  - (biol.) mehrfache (mehr als zwei) Ausführung des haploiden Genoms in einem Zellkern
- siehe auch: diploid, haploid

### **Ranking (*rank-based fitness assignment*)**

- *rank* (engl.) – Reihe, Reihenfolge
  - (EA) Verfahren der Fitneßzuweisung, wobei nur der Rang eines Individuums in der sortierten Reihe der Zielfunktionswerte verwendet wird und nicht der Wert des Zielfunktionswertes
- siehe auch: Selektion

### **Rekombination**

- *recombinare* (lat.) – neu verteilen, wieder vereinigen
  - (biol.) intrachromosomal R.: Austausch von Allelen zwischen homologen Chromosomen, chromosomal R.: Neuverteilung von ursprünglich väterlichen und mütterlichen Chromosomen bei der Meiose (Reduktionsteilung) und Zusammenführung von einfachen Chromosomensätzen bei der Befruchtung
  - (EA) evolutionärer Operator; Erstellung neuer Individuen durch Kombination bzw. Neuverteilung der Information zweier oder mehr Individuen (Eltern)
- siehe auch: crossover

### **rezessiv**

- (lat.) überdeckt
  - (biol.) Allel, das sich im heterozygoten Zustand nicht ausprägt
- siehe auch: Dominanz

### **Selektion**

- *selectio* (lat.) – Auswahl
  - (biol.) Begriff der Populationsgenetik, Auslese, Auswahl, Zuchtwahl
  - (EA) Auswahl von Individuen aus einem Selektionspool entsprechend der Fitneß der Individuen für die Produktion von Nachkommen
- siehe auch: S.-druck, S.-intensität, S.-pool, S.-varianz, *bias*, *spread*, *loss of diversity*

### **Selektionsdruck (*selective pressure*)**

- (biol.) Maßstab für die Stärke der Selektionswirkung in einer Population
- (EA) Parameter der Selektion: Wahrscheinlichkeit der Auswahl des besten Individuums, verglichen mit der durchschnittlichen Selektionswahrscheinlichkeit aller Individuen des Selektionspools

**Selektionsintensität (*selection intensity*)**

- (EA) Parameter der Selektion: erwarteter durchschnittlicher Fitneßwert der selektierten Individuen nach einer durchgeführten Selektion (Voraussetzung: die Fitneßwerte des Selektionspools sind normal verteilt)

**Selektionspool**

- (EA) Bereich der Population, der alle Individuen enthält, die zusammen zur Auswahl der Reproduktionspartner (Eltern) in Betracht gezogen werden. Bei einer Gesamtpopulation ist dies z.B. die Population, bei der Verwendung des regionalen Modells zur Unterteilung der Population in Unterpopulationen jeweils eine Unterpopulation und beim lokalen Modell die lokale Nachbarschaft.

siehe auch: Selektion, *deme, panmictic*

**Selektionsvarianz (*selection variance*)**

- (EA) erwartete Varianz der Fitneßwerte der selektierten Individuen nach einer durchgeführten Selektion (Voraussetzung: die Fitneßwerte des Selektionspools sind normal verteilt); Parameter bei der Bewertung von Selektionsverfahren

**sharing**

- (engl.) verteilen
- (EA) verteilen der räumlich begrenzten Ressourcen auf die Individuen; liegen viele Individuen im Suchraum dicht beieinander, dann sind die Ressourcen in diesem Bereich bald aufgebraucht, weitere Individuen können nicht in diesen Bereich gelangen; dadurch kommt es zu einer Verteilung der Individuen im Suchraum, z.B. auf mehrere vielversprechende Bereiche

siehe auch: *niching*

**spread**

- (engl.) Ausdehnung, Spannweite
- (EA) Bereich der möglichen Werte für die Anzahl der Nachkommen eines Individuums; Parameter bei der Bewertung von Selektionsverfahren

siehe auch: Selektion, *bias*

**subpopulation**

- (engl.) Unterpopulation

siehe auch: Unterpopulation

**Traveling Salesman Problem (TSP)**

- (engl.) Problem des Handlungsreisenden
- Kombinatorisches Problem, bei dem der kürzeste Rundreiseweg gesucht ist, den ein Handlungsreisender nimmt, wenn er von einem Ausgangsort aus vorgegebene Orte genau einmal besucht und wieder zum Ausgangsort zurückkehrt.
- Einige praktische Optimierungsprobleme lassen sich direkt oder teilweise auf dieses Problem zurückführen.

## Unterpopulation

- *subpopulation* (engl.)
- (EA) Teil der Individuen einer Population, die untereinander Nachkommen bilden können (zu einem Selektionspool gehören), aber nicht mit den anderen Individuen der Population in Verbindung stehen  
siehe auch: Migration, *deme*

## Variable

- (EA) Teil eines Individuums, die Gesamtheit der Variablen bildet ein Individuum
- **reelle Variable** (auch kontinuierliche Variable): kann beliebige reelle Werte innerhalb vorgegebener Grenzen annehmen ( $lb \leq x \leq ub$ ),  $lb$  stellt die untere Schranke dar (*lower bound*) und  $ub$  die obere Schranke (*upper bound*)
- **diskrete Variable**: kann nur bestimmte Werte annehmen
- **ganzzahlige Variable**: diskrete Variable, die nur ganzzahlige Werte innerhalb vorgegebener Grenzen annehmen kann ( $lb \leq x \leq ub$ )
- **binäre Variable**: spezielle ganzzahlige Variable mit  $lb = 0$  und  $ub = 1$   
siehe auch: Allel

## Zielfunktion

- auch: Gütfunktion
- (EA) definiert das zu optimierende Kriterium bei einer Optimierungsaufgabe (und damit das zu lösende Problem);  
die Zielfunktion erhält als Eingabeparameter ein Individuum (sowie zusätzliche problemspezifische Parameter), berechnet damit die Zielfunktion und liefert als Ausgabe den Zielfunktionswert für dieses Individuum  
siehe auch: Fitneß

## Zielfunktionswert

- *objective value* (engl.), auch: Gütewert
- (EA) Wert, den die Zielfunktion für ein Individuum liefert; gibt die Güte eines Individuums an; der Zielfunktionswert ist problemspezifisch (z.B. Wertebereich, Skalierung)
- der Zielfunktionswert entspricht **nicht** der Fitneß; allerdings kann der Zielfunktionswert als Fitneß verwendet werden  
siehe auch: Fitneß

Bei der Arbeit am Glossar fanden u.a. die folgenden Quellen Verwendung:  
[Fwb63], [Hen95], [RM54], [Wer68], [Lit92] und [Fon95].

# Sachverzeichnis

*2-opt* 44, 56  
*2-opt move* 56

*3-opt* 56  
3x3 Damespiel 269

4. Dimension 138

5. Dimension 138

Abbild des Gütegebirges 190  
Abbruchkriterium 63, 166  
– abgeleitet 65  
– direkt 64  
– festgelegter Zielfunktionswert 64  
– garantiertes Ende 64, 70  
– globales Optimum 64  
– GuterSchlechtester **68**  
– *Kappa* **69**  
– laufender Mittelwert **67**  
– maximale Generationen 64  
– maximale Rechenzeit 64  
– maximale Zielfunktionsberechnungen 64  
– *Phi* **69**  
– praktischer Einsatz 70  
– problemabhängige Skalierung 66  
– *running mean* **67**  
– Standardabweichung der Zielfunktionswerte **65**  
– zufriedenstellender Zielfunktionswert 64  
abgeleitete Größe 134  
Abhängigkeiten zwischen Operatoren 171  
Abkürzungen XIII  
Ablauf der Migration 94  
Ablauf Evolutionärer Algorithmen 8

Abschneideschwelle 27  
Abschneideselektion 52  
absolute Anordnung 180  
absolute Position 180  
absolute Position der Knoten 81  
Abstand zwischen Knoten 80  
ACKLEY's path 149  
Adaption  
– Kovarianzmatrix 51  
– Mutationsparameter 50  
– Mutationsschrittweiten 46, 51, 177, 195  
– Parameter der Optimierung 101  
– Suchrichtung 51, 177  
adaptiver Algorithmus 270  
Algorithmen-Visualisierung 115  
allgemein einstellbare Verfahren und Operatoren 172  
allgemeine Definition von Nachbarschaften 80  
Alter von Individuen 59, 60  
Aminosäure 261  
Analogie zur Fitneßzuweisung 103  
Analyse der mehrdimensionalen Visualisierung 143  
Anfangspopulation 60  
Anfangsschrittweite 177  
Anfangszustand 191  
Angleichung der Variablenwerte 130  
Anleitung zur Bearbeitung eines Problems 185  
Anpassung der Strategieparameter 114  
Anwendung  
– Dieselmotormodell 204  
– Gewächshaus 239  
– mathematische Funktion 194  
– Querlenkung eines Fahrzeugs 226  
Anwendung auf Praxisprobleme 185  
Anwendung der mehrdimensionalen Visualisierung 140  
Anwendung verschiedener Strategien 50, 74, **96**, 132, 173, 193, 213, 233

- Analyse 101
- Anzahl der nächsten Nachbarn 82
- Anzahl der Unterpulationen 88, 89
- Anzahl Individuen 89
- ASPARAGOUS** 85
- assignment problem* 42
- Aufstellung der Zielfunktion 185, 186
- Aufteilung der Ressourcen 102, 103
- Ausbreitung von Information 79, 88
- Ausbreitungsgeschwindigkeit 79
- Ausführungszeit von Softwaremodulen 147
- Ausgabe
  - Bildschirm 152
  - binäre Datei 152
  - Textdatei 152
- Ausgangsdaten 187
- Ausgangsgrößen der Simulation 151
- Ausprägung der Information 264
- Austausch von Individuen 88
- Austausch von Informationen 87
- Austauschmutation **56**
- Auswahl der Individuen 74
- Auswahl der Migranten 94
- Auswahl der Zentren der Nachbarschaften 83
- Auswahl innerhalb einer Nachbarschaft 83
- Auswertung des Verlaufs der Optimierung 214
- Auswertung von Optimierungen 185, 192
  
- BAGLEY** 264, 269, 273
- BEAGLE** 271
- Bearbeitung eines neuen Optimierungsproblems 186
- Beschleunigung der Berechnung 95
- Beschränkung der Systemgröße 191
- Beschreibung von Nachbarschaften 92
- Bewertung
  - Differenz von Verläufen 212
  - Differenz zu einem kritischen Wert 230
  - exponentiell 212
  - maximale Differenz 212
- Bewertung der Qualität einzelner Lösungen 214
- Bewertung mehrerer Lösungen 151
- bias** **24**, 25, 26
- BIENERT** 272
- Bilanzgleichungen des Gewächshausmodells 276
- Bildung von Clustern 134, 136
- binär
  - Bäume 271
  - Initialisierung der Variablen 61
- Kodierung 53, 262
- Parameteroptimierung 172
- Repräsentation 53, 183
- blind non-stationary knapsack* 265
- BOX** 270
- Breeder Genetic Algorithm* 34
- BREMERMANN** 269
- BRINDLE** 264
- Bubblesort 147
  
- CAVICCHIO** 270
- CEC** 275
- cellular Genetic Algorithm* 86
- CHC** 63
- Chromosom 261
- Chromosomenpaar 268
- Chromosomensatz 261, 263
- cluster* 131
- coarse grained model* 74
- common edges* 42
- competing subpopulations* 102
- Congress on Evolutionary Computation* 275
- consumption factor* 112
- Continous-EP 272
- contour plot* 145
- crossover* **39**
  - *cycle* 46
  - *double-point* 40
  - *edge assembly* 46
  - *multi-point* 40
  - *reduced surrogate* 41
  - *shuffle* 41
  - *single-point* 40
  - *two-point* 40
  - *uniform* 41
- crowding* 274
  
- Darstellung mehrerer Individuen 140
- Darstellung mehrkriterieller Zielfunktionswerte 140
- DARWIN, CHARLES** 2
- Daten zur Visualisierung 117, 152
- DE JONG** 274
  - Funktion 1 149
  - Testfunktionen 274
- deceptive problems* 148
- Definition zusätzliche Parameter 187
- Definitionsreich der Variablen 38, 48, 61, 131, 177
- Degree of Nness* 265
- Degree of Oneness* 265
- Dekodierung binärer Variablen 183

- Der Stärkere überlebt! 7  
 detaillierte Auswertung 153  
 deutlicher Fortschritt 106  
 diagonales Newtonverfahren 140  
 Dieselmotormodell 185  
 diffuse Ausbreitung guter Individuen 125  
 Diffusionsmodell 74, 77  
 Dimension Farbe 138  
 Dimension Zeit 138  
 Dimensionalität eines Problems 191  
 diploid 263, 269
  - Adaption an verändernde Zielfunktion 265
  - Chromosomen 269
  - frühe Arbeiten 264
  - Kodierung 268
  - Repräsentation 269
  - Schemata 270
  - Speicher vergangener Lösungen 266
  - Variablenstruktur 268
  - Verschiedenartigkeit der Population 265
  - Vorteile 265*directed mutation* 268  
 direkte Darstellung der Zielfunktion 145  
 direkte Unterteilung der Population 125  
 diskrete Mutation 269  
 diskrete Rekombination 35, 41, 164, 175, 178  
*dissimilarities* 139  
 Distanz zwischen Individuen 134  
 Distanz zwischen Individuen in ihrer räumlichen Anordnung 136  
 Distanz zwischen Nachbarn 77, 78  
 Distanzkarten 136  
 Distanzverteilung 134  
 Diversität der Population 135, 136, 270  
*diversity* 130, 134  
 DNS 261  
 Dokumentation der Berechnungen 153  
 Dokumentation der Ergebnisse 151  
 dominantes Allel 263  
 Dominanz 268  
 Dominanzmechanismus 263, 269  
 Dominanzschema 264  
 Dominanzwechsel 265  
*double-point crossover* 40  
 dreialeliges Schema 264, 270  
 Dreiecksungleichung 139  
*duplication* 270  
 Durchführung von Optimierungen 185, 192  
 dynamischen Verteilung der Ressourcen 114  
*E. coli* 268  
 EAX 46  
 Eck-Rekombination 35  
*edge assembly crossover* 46  
*edge recombination* 42, 44, 46, 164, 179  
*edge table* 42  
*edge-2* 42  
*edge-3* 42, 46  
*edge-4* 42  
 effektive Verteilung der Ressourcen 102  
 Eigenschaften der Zielfunktion 188  
 Eigenschaften des Systems 192  
 Einbringen von problemspezifischem Wissen 263  
 Einbringen von Vorwissen 62  
 eindimensionale Schnitte 145, 189, 208  
 Einfügemutation 55  
 Einfügen der Nachkommen 165  
 einheitliche Population 88  
 einkriterielle Optimierung 23  
 Einordnung als Optimierungsproblem 187  
 Einordnung des Problems 185  
 Einsatz Evolutionärer Algorithmen
  - diskontinuierliche Zielfunktionen 11
  - multimodale Zielfunktionen 12
  - nichtlineare Zielfunktionen 11
  - Variablen unterschiedlicher Repräsentation 12
 Einsatz konkurrierender Unterpopulationen 108  
 Einschränkung der Wertebereiche 193  
 Einschränkung des Suchbereiches 243  
 elitest Methode 84  
 elitest Selektion 59  
 elitest Strategie 173  
*elitism* 274  
 Endzustand 191  
 Energiebilanz 276  
 Entwicklung Evolutionärer Algorithmen 267  
 Entwicklung spezieller Operatoren 193  
*EP* 275  
*epistatic interactions* 148  
 Erfolg der Unterpopulationen 97, 102, 103, 111  
 erfolglose Unterpopulation 102  
 erfolgreiche Unterpopulation 102  
 Ergänzung von Strategien 101  
 erhöhte Stabilität der Information 266  
 Erkennung von Buchstaben 270  
 Ermittlung guter Strategien 101  
 Erstellung der Anfangspopulation 8  
 erweiterte Initialisierung 62  
 erweiterte Linien-Rekombination 164  
 Erweiterung auf beliebige Dimensionen 80

- 
- Erweiterungen des regionalen  
Populationsmodells 74
- Erweiterungen für Evolutionäre  
Algorithmen 9
- euklidischer Abstand 134, 139, 149
- Evolution 2
- Evolution Strategies* 267, 272
- Evolution von Computerprogrammen 275
- Evolutionäre Algorithmen 2, **275**
- Evolutionäre Programmierung 51, 267,  
271
- evolutionärer Kreislauf 8
- Evolutionary Algorithms* 157
- evolutionary computation* 275
- Evolutionary Programming* 267, 271, 275
- Evolutionary Programming Society* 272
- Evolutionsmechanismen 2
- Evolutionsstrategien 51, 176, 267, 272
- EX 42, 179
- Expertenwissen 62
- extended competition model* 112
- Farbenteppich 121, 123, 128, 136
- farming* Modell 74
- FDC 148
- feine Mutation 176
- feine Suche 173, 193
- feinkörniges Modell 74, 87
- Fernbereich 209
- Festlegung der Eingangsdaten 187
- fine grained model* 74
- finite Automaten 271
- finite-state machines* 271
- fitness distance correlation* 148
- fitness sharing* 23
- Fitneß 16, 18, 24
- Fitneßaufteilung 23
- Fitneß-Distanz-Korrelation 54, 148
- Fitneß-Distanz-Verteilung 148
- Fitneßfunktion 16, **17**
- Probleme 17
  - Skalierungen 17
- fitneßproportionale Auswahl 94
- fitneßproportionale Selektion 83
- Fitneßzuweisung **16**, 97, 103, 163, 172
- mehrkriteriell 21
  - Methode der Wahl 20
  - proportional 17
  - reihenfolgebasiert 18
  - variabel 269
- fließende Unterteilung der Population 77,  
125
- FOGEL, D. 272
- FOGEL, L. 271
- FORSYTH 271
- Fortpflanzungswahrscheinlichkeit 16
- FORTRAN-Gateway-Funktion 207
- Fortschritt der Berechnungen 152
- Fortschrittsgeschwindigkeit 273
- FRASER 268
- FRIEDBERG 270
- GALESIA* 275
- ganzzahlige Repräsentation 182
- ganzzahlige Variablen
- Initialisierung 61
  - Abbruchkriterien 166
  - Beispiele 168
  - Beispieldateien 168
  - Dokumentation 157, 169
  - erweiterter Ergebnisgrafiken 169
  - Fitneßzuweisung 163
  - high-level Funktionen 162
  - Initialisierung 163, 169
  - Konkurrenz 166
  - Migration 166
  - Mutation 165
  - Namenskonventionen 160
  - neue Erkenntnisse 170
  - plattformunabhängig 170
  - problemspezifisches Wissen 169
  - Protokollierung 167
  - Ranking 163
  - regionales Modell 166
  - Schichtenmodell 158
  - Scriptfunktionen 158, 159, 161
  - Standardtestfunktionen 168
  - Terminierung 166
  - Toolboxfunktionen 158, 159, 161
  - Verfügbarkeit 170
  - Version 1.9x 157
  - Version 2.x 157
  - Version 1.95 4, 260
  - Verwendung verschiedener Strategien  
166
  - Visualisierung 166
  - vordefinierte Algorithmen 161
  - Wiedereinfügen 165
  - Zentralfunktion 158, 162
  - Zielfunktionen 168
  - zusätzliche Parameter 168, 169
- GECCO* 275
- generation gap* 52, **57**, 83, 173
- Generationslücke 57, 173
- Generierung von Computerprogrammen  
270

- 
- Genetic Algorithms* 157, 267, 273
  - Genetic Programming* 275
  - Genetische Algorithmen 178, 183, 267, 273
    - erste Benennung 269
    - genetische Drift 23, 88
    - Genetische Programmierung 275
    - genetisches Einfrieren 60
    - Genotype-Space Mapping* 143
    - geographische Schranken 89
    - Gewächshaus 186
      - Anzahl der Variablen 242
      - Bilanzgleichungen im Innenraum 241
      - detaillierte Steuerung 252
      - Effektivität der Pflanzenproduktion 239
      - Einbeziehung übergeordneter Langfristmodelle 252
      - Einhaltung der Beschränkungen 242
      - Einschränkung des Suchbereiches 243
      - Erzielung hoher Erträge 239
      - Gewinn 242
      - hochdimensionale Optimierungsaufgabe 240
      - Initialisierung der Anfangspopulation 243
      - Klimamodell 240
      - Klimavoraussagen 253
      - Kurzfriststeuerung 240
      - Langfristmodell 252
      - Langfriststeuerung 240
      - Maximierung des Gewinns 242
      - mikrometeorologische Bedingungen 241
      - optimale klimatische Wachstumsbedingungen 239
      - Optimierung mit veränderten Preisen 246
      - Optimierungen für durchschnittliche Tage 244
      - Optimierungen mit reale Wetterdaten 247
      - Pflanzenmodell 241
      - repetierende Optimierung 253
      - Repräsentation der Individuen 242
      - Simulationsperiode 242
      - Stellgrößen 241
      - Steuerung der Zustandsgrößen 239
      - Unterteilung der Simulation 242
      - vollständige Steuerung aller Zustandsgrößen 252
      - Vorgabe guter Steuerstrategien 243
      - Vorwissen 243
    - Gewächshausmodell
      - Beschränkung der Temperatur 280
      - Beschränkungen 280
    - Bilanzgleichungen 276
    - Biomasse 279
    - Dampfdichteänderung 277
    - Energiebilanz 276
    - Erlösgröße 280
    - Ertrag 280
    - Gemüsepreis 279
    - Gewinn 279
    - Heizungskosten 279
    - Kohlendioxidbilanz 277
    - Kohlendioxidkonzentration 277
    - Kohlendioxidkosten 279
    - Langfrist-Strategien 280
    - Stellgrößen 280
    - Trockensubstanzgehalt 280
    - Trockensubstanzproduktion 280
    - Vermeidung von Streß 280
    - Wasserdampfbilanz 277
  - gewichtete Summe 23, 213, 230, 243
  - Gewichtungsfunktion 274
  - global orientierte Parameteroptimierung 172, 175
  - global orientierte Suche 211
  - globales Modell 74, 96
  - globales Populationsmodell 75
  - goal* 22, 163
  - goal programming* 22
  - GOLDBERG 274
  - GP* 275
  - GPX 179
  - gray code* 53
  - gray Kodierung* 53
  - Grenzen zwischen Unterpopulationen 125
  - grobe Mutation 176
  - grobe Suche 173, 193
  - großkörniges Modell 74
  - Größe der Nachbarschaft 78
  - Größe der Unterpopulationen 88, 89, 127
  - große Nachbarschaft 125
  - Grundelemente des Evolutionsprozesses 267
  - Gültigkeit des Schema-Theorems 274
  - Gütegebirge 145, 188
  
  - Hammingdistanz 134, 149
  - Handlungsreisender 172, 179
  - haploid 263
  - haploides Schema 270
  - Häufigkeit der Migration 88
  - heterozygote* Allele 263
  - Hinton-Diagramm 126
  - Histogramm 134
  - hochdimensionale Visualisierung 138
  - HOLLAND 273

- HOLLSTIEN 264, 269  
*homozygote* Allele 263  
HTML 169, 170  
hybride Optimierung 44  
hybrides Modell 74  
*hyper sphere* 196  
Hyperellipsoid 201  
Hyperkreuz 81  
Hyperkugel 82  
Hyperstern 82  
– schief 82  
Hypertorus 80  
Hyperwürfel 92
- ICEC* 275  
*ICGA* 275  
*image plot* 121  
Implementierung Evolutionärer Algorithmen 157  
Individuenanzahl 89  
Initialisierung 163  
– Anfangspopulation 243  
– binäre Variablen 61  
– erweitert 62  
– ganzzahlige Variablen 61  
– Individuen 60  
– kombinatorische Probleme 61  
– nicht-zufällig 62  
– reelle Variablen 61  
– zufällig 61  
*inoculation* 61  
*insert mutation* 180  
interaktiver Test 188  
intermediäre Rekombination 36, 164  
*International Conference on Evolutionary Computation* 275  
*International Conference on Genetic Algorithms* 275  
Interpretation des genetischen Kodes 262  
Inversion 268, 270  
Isolation durch Distanz 77, 79, 134  
Isolationszeit 88, 174
- job shop scheduling* 172  
*JSS* 172, 179, 180
- Kanten 80  
*kernel density estimation* 134  
Klassifikation von Populationsmodellen 73, 74  
Klassifikationsprobleme 271  
Klassifizierung der Visualisierung 116
- kleine Nachbarschaft 125  
Knoten 80  
Kodierung 53  
– binär 53, 270  
– binäre Variablen 178  
– Erbinformation 262  
– genetische Information 261  
– *gray* 53, 269
- Koexistenz von Unterpopulationen 103  
Kombination mehrerer Verfahren 74  
Kombination von ganzzahligen und binären Variablen 181  
Kombination von Operatoren 171  
kombinatorische Optimierung 179  
komplexe Anwendung 186  
Konferenzen 275  
Konkurrenz 8, 166  
Konkurrenz zwischen Unterpopulationen 74, 102, 125, 132, 173, 195, 233  
Konkurrenzauswahl 107  
Konkurrenzintervall 105, 125, 175  
Konkurrenzrate 106, 175  
Konvergenz der Population 119  
Konvergenzdiagramm 119  
*k-opt* 56  
Korrelation zwischen Variablen 53, 121, 147, 176, 189, 199, 209  
KOZA 275
- Langzeitspeicher 264  
*lateral control of an autonomous road vehicle* 226  
Lebensdauer eines Individuum 58, 59  
– durchschnittlich 59  
– maximal 60  
Leitertopologie 82, 92  
lineare Ranking-Selektion 29, 33  
lineare Skalierung 53  
lineare Topologie 78  
lineares Ranking 19, 97, 163  
Linien-Rekombination 37, 71, 164  
– erweiterte 38  
*Local Evolutionary Algorithm* 195  
logarithmische Skalierung 53  
lokal orientierte Parameteroptimierung 172, 176  
lokale Minima 145, 147, 196  
lokales Populationsmodell 73, 74, 77, 124, 132, 134, 136  
lokales Wiedereinfügen 84  
Lokalität 8  
loss of diversity 24, 28, 32  
Lösung eines neuen Problems 185  
Lösung realer Probleme 151

- 
- Lösung von Optimierungsaufgaben 186  
low-level Funktionen 162
- Massenmutation 63  
Master-Slave-Struktur 76  
*mating schemes* 269  
Matlab-Toolbox 157  
*maximal preservative crossover* 44, 179  
maximales Lebensalter 60  
mehrdimensionale Funktionen 185  
mehrdimensionale Skalierung 138  
mehrdimensionale Visualisierung 190  
mehrere Extremwerte 193  
mehrkriterielle Fitneßzuweisung 21, 131  
mehrkriterielle Optimierung 21, 70  
mehrkriterielle Probleme 190  
mehrkriterielles Ranking 21, 163  
Merkmalsextraktion 147  
*mesh plot* 145  
*Meta-EP* 272  
*method of inequalities* 22  
Migration 8, **88**, 125, 166  
– Ablauf 94  
Migrationsauslese 88, 94  
Migrationsauswahl 88, **94**  
Migrationsintervall 88, **92**  
Migrationsmodell 74, 87  
Migrationsrate 88, **92**, 94, 174  
Migrationsstruktur 88, 174  
Migrationstopologie 79, 88, 90  
Migrationszeit 174  
Modell der Verbrennung 185  
Modellierung natürlicher Prozesse 8  
MOI 22  
*moved axis parallel hyper-ellipsoid* 201  
MPX 44, 179  
MPX2 44  
*Multi subPopulation Evolutionary Algorithm* 195  
*multidimensional scaling* 138  
multimodal 196  
multimodale Funktion 131  
multimodales Problem 70  
*multiobjective optimization* 21  
*multiple strategies* 96  
*multi-point crossover* 40  
Mustererkennung 270  
Mutation 8, **46**, 165  
– binäre Variablen 53  
– ganzzahlige Variablen 47, 176  
– *insert* 55  
– kombinatorische Probleme 54  
– *move* 55  
– reelle Variablen 47, 51, 176
- *reverse* 56  
– Schrittweitenanpassung 51  
– *scramble* 57  
– *swap* 56, 57  
*mutation range* 48  
Mutationsbereich **48**, 52, 173, 176, 177, 180  
Mutationspräzision **48**, 176, 180  
Mutationsrate **46**, **47**, 49, 55, 63, 176, 179, 180  
– optimal 47  
Mutationsschritt 46  
Mutationsschrittweite 47  
– Anfangsgröße 52  
– Bereich 49  
– minimal 49  
– obere Grenze 48  
– untere Grenze 48  
Mutationswahrscheinlichkeit 46, 270
- Nachbarschaft 8  
Nachbarschaftsmodell 77  
Nachbarschaftstopologie 90, 91  
Nachbarschaftstopologien 79  
Nahbereich 209  
Nahordnung der Datenpunkte 138  
*neural-network control* 226  
Neuro-Fuzzy-Regler 226  
Neustart eines Optimierungslaufes 63  
nichtlineares Ranking 19, 104, 163  
normalverteilte Mutation 273
- offline* Performanz 274  
*online* Performanz 274  
Optimierung  
– eines Gleichstrom-Stellers 141  
– eines Rohrkümmers 272  
– mehrkriteriell 21  
*order crossover-2* **45**  
Ordnung der Unterpopulationen 97, 103  
– gewichtet 98  
OWENS 271  
OX2 45
- panmictic* Population 88  
*Parallel Problems solving from Nature*  
275  
parallele Modelle 74  
parallele Zielfunktionsberechnung 76  
Parallelisierung der Berechnungen 73  
Parallelität 8  
Parameteridentifikation 172, 185, 204

- 
- Parameteroptimierung 172
    - binär 172
    - binäre Variablen 178
    - global orientiert 172, 175
    - lokal orientiert 172, 176
    - verschiedene Repräsentationen 181
  - Pareto-Dominanz 21
  - Pareto-Front 70
  - Pareto-Optimalität 21
  - Pareto-Ranking 21
  - partially matched crossover* 45
  - PCA 140
  - Permutation 54
  - Permutation von Elementen 61
  - Pfad durch den Lösungsraum 140
  - Pfad durch den Suchraum 140
  - Pflanzenmodell
    - Dampfdrucksättigungsdefizit 279
    - Kohlendioxidgaswechsel 278
    - Transpiration 279
    - Zustandsgleichungen 276
  - Plateaus 147
  - Ploidie der Information 263
  - plt* 21
  - pole-placement-Regler* 228
  - polygenic inheritance* 265
  - Polyplidie 263
  - population diversity* 265
  - Populationsgröße 90, 177
  - Populationsmodelle 73
    - global 75
    - lokal 77
    - Reichweite der Selektion 74
    - Unterscheidungsmerkmal 75
  - position crossover* 45
  - Position der Unterpopulationen 127
  - Position einer Unterpopulation 97
  - position recombination* 179
  - PPSN* 275
  - premature convergence* 17, 23, 85
  - principal component analysis* 140
  - Problemklassen 171, 184
  - problemspezifische Informationen 62
  - problemspezifische Visualisierung 150, 151, 185, 188, 214
  - problemspezifisches Wissen 169
  - production scheduling* 172
  - Produktionsreihenfolge 172
  - Programm-Visualisierung 115
  - proportionale Fitneßzuweisung 17
  - proportionale Selektion 274
  - Protokollierung 152, 167
    - Beginn des Laufs 152
    - Daten 152
    - Ende des Laufs 152
  - jede Generation 152
  - spätere Auswertung 152
  - während eines Laufs 152
  - pseudo-Meiosis GA* 265
  - PX** 45
  - QAP** 180
  - Querlenkung 186
    - Abstand zur Straßenmitte 226
    - Abweichung von der Fahrbahnmitte 230
    - Änderung des Lenkungswinkels 226
    - Anforderungen an Regler 227
    - *course angle* 226
    - Drehwinkel 226
    - eines autonomen Straßenfahrzeuges 226
    - kein Kreuzen der Fahrbahnmitte 231
    - Komfort der Fahrzeuginsassen 226
    - kontinuierliches System 229
    - Kurswinkel 226
    - Lenkungswinkel 226
    - Querbeschleunigung 230
    - Schnelligkeit 226
    - Schwimmwinkel 226
    - Sicherheit 226
    - *sideslip angle* 226
    - Simulation des Gesamtsystems 228
    - Stabilitätstest 229, 232
    - *steer angle* 226
    - Struktur des Reglers 228
    - Übertragungsfunktion 227
    - *yaw angle* 226
    - zeitdiskretes System 229
    - Zustandsgleichungen 227
    - Zustandsmodell 227
  - radial fitness plot* 133
  - Radienquotient 86
  - Rang einer Unterpopulation 98
  - Ranking 18, 97, 172, 271
    - mehrkriteriell 21
  - RASTRIGIN's Funktion 122, 196
  - rauhes Gütegebirge 211
  - räumlich beschränkte Selektion 134
  - Raum-Rekombination 36
  - RECHENBERG 272
  - reduced surrogate* 41, 178
  - reelle Kodierung 263
  - reelle Variablen
    - Initialisierung 61
  - regionale Fitneßberechnung 87

- 
- regionales Populationsmodell 73, 74, **87**,  
124, 131, 134, 166, 174  
Reglereinstellung 185  
Reichweite der Selektion 74  
reihenfolgebasierte Fitneßzuweisung 18,  
97  
*reinsertion* **57**
  - *elitest* 58
  - *pure* 58
  - *rate* **57**
  - *uniform* 58
  - *with offspring selection* 58Rekombination 8, **34**, 164
  - alle Variablen 35
  - binäre Variablen 39, 71
  - *cycle crossover* 46
  - diskrete **35**, 41, 71
  - *edge assembly crossover* 46
  - *edge recombination* **42**, 46
  - erweiterte Linien-Rekombination 38
  - ganzzahlige Variablen 35, 39
  - intermediär 36, 71
  - kombinatorisch 42
  - Linien-Rekombination 37, 71
  - *maximal preservative crossover* 44
  - mehr als zwei Eltern 272
  - *order crossover-2* 45
  - *partially matched crossover* 45
  - *position crossover* 45
  - reelle Variablen 35Rekombinationsrate 175, 178  
relative Anordnung 180  
relative Position der Knoten 81  
repetierende Optimierung 253  
Repräsentation
  - binär 149, 183
  - diploid 269
  - ganzzahlig 149, 182
  - Individuum 53
  - nicht-binär 269
  - reell 149, 183
  - strukturiert 275
  - Variablen 117, 130Reproduktion 8  
Ressourcen 102  
Ressourcenverbrauch 105, 112  
*reverse mutation* 180  
rezessives Allel 263  
Rezessivität 268  
Ringtopologie 90, 91  
*RMeta-EP* 272  
robuste Parametrisierung 174  
ROSENBERG 264, 268  
ROSENROCK's Funktion 53, 121, 141,  
198  
Rouletteselektion **25**, 28  
*Rprop*-Verfahren 140  
*rugged fitness landscapes* 148  
*running mean* 67  
  
Sammon-Mapping 140, 190  
*scalarization method* 23  
*scatter plot* 149  
*scheduling* 42, 45  
Schema-Theorem 274
  - Gültigkeit 274schiefer Hyperstern 82  
Schnitte durch die Zielfunktion 145  
Schrittweite 38  
**SCHWEFEL** 272  
Schwere einer Funktion 148  
*scramble mutation* 180  
Selbstadaptation der Strategieparameter 273  
*selection intensity* 25  
*selection variance* 25  
*selective pressure* 24  
Selektion 2, 8, **24**, 164, 172, 267
  - Analyse der Verfahren 28
  - *elitest* 59
  - in Nachbarschaft 77, 83
  - loss of diversity 24
  - optimal (fitneßproportional) 26
  - Rouletteselektion **25**
  - Selektionsdruck 19, 24, 29, 52, 79, 94,  
172, 177, 269, 271
  - Selektionsintensität 25, 28, 31, 86
  - Selektionsparameter 28
  - Selektionspool 16, 59
  - Selektionspool Nachbarschaft 77
  - Selektionsvarianz 25, 28, 32, 88
  - Selektionswahrscheinlichkeit 16, 24
  - *stochastic universal sampling* **25**
  - Truncation-Schwelle **27**
  - Truncation-Selektion **27**
  - Turniergröße 26
  - Turnierselektion **26**
  - Vergleich der Verfahren 31separierbare Funktion 47, 96, 196, 201  
*sequential symbol prediction* 271  
*sharing* 23  
*shuffle crossover* 41  
Simplex-Algorithmus 270  
Simulation 151  
Simulation mehrerer Datensätze 188  
*Single Population Evolutionary Algorithm*  
195  
*single-point crossover* 40, 270  
Skalierbarkeit der Methoden 144  
Skalierbarkeit des Problems 191

- Skalierung
  - linear 53
  - logarithmisch 53
  - Variablen 131
  - Zielfunktionswerte 120, 126
- Softwaretestsysteme 147
- Speicherung der Erbinformation 262
- Spezialistenwissen 62
- spezielle Initialisierung 193
- spezielle Verfahren und Operatoren 179
- spezielles Verfahren 194
- Spieltheorie 269
- spread* 24, 25, 26
- Sprünge 147
- Stabilität 186
- stagnation* 17
- steady-state* Algorithmus 59, 76
- Steuerung des Gewächshausklimas 239
- Steuerung eines dynamischen Systems 239
- Steuerung eines Systems 151
- Steuerung von Gasleitungssystemen 274
- Steuerung von Klimagrößen 186
- stochastic sampling with replacement* 25
- stochastic universal sampling* 25, 28
- stochastisches Verhalten 208
- Struktur Evolutionärer Algorithmen 8
- strukturierte Lösungen 271
- strukturierte Repräsentation 275
- Strukturmodell 185
- Strukturowptimierung einer Zweiphasen-Überschalldüse 272
- Summe der Fehlerquadrate 139
- sus* 25
- swap mutation* 57, 180
- Systematisierung der Visualisierung 116
  
- termination criteria* 63
- Terminierung 166
- Testfunktionen 185, 194
  - binäre Variablen 168
  - dynamische Systeme 168
  - ganzzahlige Variablen 168
  - mehrkriteriell 168
  - n-dimensional 168
  - zwei reelle Variablen 168
- Testsuite 274
- tetraploid 263
- Theorie adaptiver Systeme 274
- Topologie
  - eindimensional 78
  - ganzer Ring 78
  - ganzer Stern 78
  - ganzes Kreuz 78
  - halber Ring 78
  - halber Stern 78
  - halbes Kreuz 78
  - hochdimensional 78
  - Kreis 78
  - Nachbarschaft 77, 78
  - schiefer Stern 78
  - Unterpopulationen 88
  - Vergleich 92
  - Vergleich 80
  - vollständiges Netz 90, 94
  - zweidimensional 78
- Torus 80, 92
- Träger der Gene 261
- Transactions on Evolutionary Computation* 275
- transition table* 271
- Transpiration 279
- triallelic scheme* 264
- triangle inequality* 139
- triploid 263
- truncation selection* 52
- Truncation-Schwelle 27, 30
- Truncation-Selektion 27, 30, 34, 58
- TSP 42, 44, 46, 172, 179, 180
- Turniergröße 26, 29
- Turnierselektion 26, 29, 33
- two-point crossover* 40, 270
  
- Überprüfung der Zielfunktion 151, 216
- Umverteilung der Ressourcen 106
- Umwandlung in binäre Repräsentation 53
- Umwandlung von Variablen 183
- uneingeschränkte Migration 90, 94
- Ungleichheit 139
- uniform crossover* 41
- Unstetigkeiten 145
- Unterpopulationen 87
- Unterpopulationsanzahl 89
- Unterpopulationsminimum 107, 175
- Unterscheidung der Populationsmodelle 74
- Untersuchung des Systemverhaltens 188
- Untersuchung verschiedener Konfigurationen 74
- Untersuchungen zum Systemverhalten 207
- Unterteilung der Population 73
- Unterteilung in mehrere Optimierungsaufgaben 191
  
- Variabilität 263
- variable Fitneßzuweisung 269
- Variation 2, 267
- Variationsdiagramm 145, 208
- verändernde Zielfunktion 265

- Veranschaulichung der Ähnlichkeit 148
- Verbrennungsmodell 185
- Verfahren der Wahl 194
- Verfahren des steilsten Abstiegs 140
- Vergleich der Topologien 80
- Vergleich mehrerer Lösungen 151
- Vergleich mehrerer Optimierungsläufe 140
- Vergleich verschiedener Lösungen 151
- Vergleich von mehreren Läufen 142
- Vergleiche Evolutionärer Algorithmen 186
- Verlauf der besten Zielfunktionswerte 115
- Verlauf der Simulation 151
- verrauschte Zielfunktion 53
- Verringerung der Anzahl der Variablen 191
- Verschiedenartigkeit der Individuen 88, 130, 134
- verteilte Zielfunktionsberechnung 76
- Verteilung der Gütwerte 99
- Verteilung der Rechenressourcen 173
- Verteilung der Ressourcen 102
- Verteilung der Variablenwerte 130
- Verteilungsdichte der Distanzen 134
- Verteilungsdruck 104
- Verwendung verschiedener Strategien 166
- Verwendung von Unterpopulationen 131
- Vielfalt in der Population 63
- virtuelle Inseln 85
- Visualisierung 115, 166, 193
  - abgeleitete Werte 117
  - Anzahl der Individuen 117
  - Aufgaben 118
  - Beschränkung auf drei Dimensionen 138
  - Daten 117
  - direkte Werte 117
  - Distanzkarten 136
  - Distanzverteilung 134
  - Eigenschaften der Zielfunktion 144
  - globale Eigenschaften 119
  - globales Verhalten der Population 119
  - Individuen einer Generation 128
  - Klassifizierung 116
  - lokale Eigenschaften 128
  - Schema der Systematisierung 118
  - Systematisierung 116
  - Variablen der Individuen 128
  - Variablen des besten Individuums 121
  - Zeitintervall der Daten 116
  - Zielfunktionswert des besten Individuums 119
  - Zielfunktionswerte aller Individuen 123
- Zielfunktionswerte einer Population 131
- Zustand der Population 128
- visuelles Ranking der Zielfunktionswerte 140
- vollständige Netztopologie 90, 94
- Vorteil reelle Repräsentation 54
- Voruntersuchungen zum Systemverhalten 185, 186
- vorzeitige Konvergenz 17, 63, 85, 93, 193
  
- WALSH 271
- Wechselwirkungen der Gene 262
- Weg durch den Lösungsraum 190
- Weg durch den Suchraum 190
- weighted sum* 23
- WEINBERG 268
- Widerstandsmiminierung einer Gelenkplatte 272
- Wiedereinfügen 57, 165
  - einfach 58
  - elitest 58
  - in Nachbarschaft 83
  - lokal 84
  - mit Nachkommenauswahl 58
  - von Nachkommen 77
  - zufällig 58
- Wiedereinfügerate 52, 57, 173
- Wissen von Fachleuten 62
- worker/farmer* Algorithmus 74
  
- Zentrum einer Nachbarschaft 83
- Zerklüftung 147
- Zielfunktion 160, 187
  - Beispiele 168
  - verrauscht 53
- Zielfunktionswerte der Individuen einer Population 131
- zufällige Initialisierung 61
- Zufallssuche 270
- Zustandsgleichungen des Pflanzenmodellles 276
- Zustandsgrößen der Simulation 151
- zweidimensionale Nachbarschaft 136
- zweidimensionale Schnitte 145, 189, 197, 208
- zweidimensionale Topologie 78
- zweidimensionales Gitter 92
- zweistufiger genetischer Algorithmus 268