GMD Research Series

GMD –
Forschungszentrum
Informationstechnik
GmbH

Dirk Schlierkamp-Voosen

Populationsbasierte Wettbewerbsmodelle zur Strategieanpassung in Evolutionären Algorithmen

N° 8/1999

GMD – Forschungszentrum Informationstechnik GmbH
Schloß Birlinghoven
D-53754 Sankt Augustin, Germany
Telefon +49 -2241 -14 -0
Telefax +49 -2241 -14 -2618
http://www.gmd.de

Die vorliegende Veröffentlichung entstand im/
The present publication was prepared within:
Institut für Autonome intelligente Systeme (AiS)
Institute for Autonomous intelligent Systems
http://ais.gmd.de/

Anschrift des Verfassers/Address of the author:
Dirk Schlierkamp-Voosen
Ebereschenweg 3
D-53127 Bonn

290

Die Deutsche Bibliothek - CIP-Einheitsaufnahme:
Schlierkamp-Voosen, Dirk:
Populationsbasierte Wettbewerbsmodelle zur Strategieanpassung in Evolutionären Algorithmen / Dirk Schlierkamp-Voosen.
GMD – Forschungszentrum Informationstechnik GmbH. - Sankt Augustin:
GMD – Forschungszentrum Informationstechnik, 1999
(GMD Research Series ; 1999, No. 8)
Zugl.: Dortmund, Univ., Diss., 1998
ISBN 3-88457-357-8

ISSN 1435-2699
ISBN 3-88457-357-8

Abstract

Evolutionary algorithms simulate the adaptation of a population of individuals to its environment. The adaptation can be seen as seeking for an optimum on a fitness landscape. In nature there exists not only a single population, but a variety of spatially structured populations that interact which each other.

In this dissertation, evolutionary algorithms are extended to ecological models that simulate the competition between different populations. A competition concept is defined that controls the size of populations and allows online-self-adaptation of different search strategies during the search process. This concept is applied to continuous parameter optimization. Therefore, competition models are designed that are based on the genetic operators of evolution strategies and the Breeder Genetic Algorithm.

The new concept is compared to classical optimization methods as well as to traditional evolutionary algorithms. A representative set of test functions is chosen, and measures for efficiency, robustness, and scalability are defined. The integration of different search strategies into the competition models leads to synergy effects that increase the efficiency as well as the robustness as compared to single search strategies.

A real world application *optical character recognition* demonstrates the benefit of the competition model that outperforms all standard methods according to the classification quality.

Keywords: Competition model, evolutionary algorithms, Breeder Genetic Algorithm, continuous parameter optimization, optical character recognition

Zusammenfassung

Evolutionäre Algorithmen simulieren den Anpassungsprozeß einer Population von Individuen an ihre Umwelt. Dieser Anpassungsprozeß kann als Optimumsuche auf einer Fitnesslandschaft angesehen werden. In der Natur findet man nicht nur eine einzige Population, sondern eine Vielzahl von räumlich strukturierten Populationen vor, die miteinander interagieren.

Die Zielsetzung dieser Arbeit ist die Erweiterung der Evolutionären Algorithmen zu ökologischen Modellen, die Wettbewerbsinteraktionen zwischen verschiedenen Populationen simulieren. Dabei werden die Populationsgrößen und Suchstrategien selbstorganisierend an die aktuelle Fitnesslandschaft angepaßt. Das Wettbewerbskonzept wird auf die Klasse der statischen, kontinuierlichen Parameteroptimierung ohne Nebenbedingungen angewendet. Dazu werden Wettbewerbsmodelle entworfen, die auf den genetischen Operatoren der Evolutionsstrategien sowie des Breeder Genetic Algorithm basieren.

Diese Wettbewerbsmodelle werden mit klassischen Optimierungsverfahren sowie klassischen Evolutionären Algorithmen verglichen. Dazu wird eine repräsentative Menge von Testfunktionen zusammengestellt und es werden Gütemaße wie Effizienz, Robustheit und Skalierbarkeit definiert. Durch die Integration verschiedener Suchstrategien erzielen die Wettbewerbsmodelle Synergieeffekte, die sowohl zur Effizienzsteigerung als auch zur Erhöhung der Robustheit gegenüber den einzelnen Suchstrategien beitragen.

Am Beispiel der Handschriftenerkennung werden die Einsatzmöglichkeiten des Wettbewerbskonzepts demonstriert, das hinsichtlich des Klassifikationsergebnisses alle anderen zum Vergleich herangezogenen Standardverfahren übertrifft.

Schlagwörter: Wettbewerbsmodelle, Evolutionäre Algorithmen, Breeder Genetic Algorithm, kontinuierliche Parameteroptimierung, Handschriftenerkennung

Inhaltsverzeichnis

1 Einleitung **1**

2 Ableitungsfreie Optimierungsverfahren **5**
2.1 Deterministische Verfahren 5
2.1.1 Quasi-Newton-Verfahren 5
2.1.2 Verfahren der konjugierten Richtungen von Powell 6
2.1.3 Downhill-Simplex-Methode 8
2.2 Stochastische Verfahren 9
2.2.1 Verfahren von Solis und Wets 10
2.2.2 Dynamic-Hill-Climbing 11

3 Evolutionäre Algorithmen **13**
3.1 Allgemeiner Aufbau 14
3.1.1 Struktur eines Evolutionären Algorithmus 14
3.1.2 Formale Definition 15
3.1.3 Genetische und populationsbasierte Operatoren 18
3.1.4 Syntax zur Spezifikation von Evolutionären Algorithmen 19
3.2 Allgemeine Operatoren 21
3.2.1 Kodierung 21
3.2.2 Selektion 22
3.2.3 Paarbildung 23
3.2.4 Genetische Operatoren 25
3.2.5 Initialisierung 30
3.2.6 Terminierung 31
3.3 Evolutionsstrategien 32
3.3.1 Kodierung und Fitnessfunktion 32
3.3.2 Selektion 33
3.3.3 Paarbildung 34
3.3.4 Rekombination 34
3.3.5 Mutation 36
3.3.6 Spezifikation von ES-Objekten 37
3.4 Genetische Algorithmen 38
3.4.1 Kodierung und Fitnessfunktion 38
3.4.2 Selektion 40
3.4.3 Paarbildung 42
3.4.4 Rekombination 42
3.4.5 Mutation 43

3.4.6 Spezifikation von GA-Objekten 44
3.5 Breeder Genetic Algorithm 44
3.5.1 Kodierung und Fitnessfunktion 45
3.5.2 Selektion 45
3.5.3 Paarbildung 46
3.5.4 Rekombination 46
3.5.5 Mutation 47
3.5.6 Spezifikation von BGA-Objekten 54
3.6 Erweiterungen 55
3.6.1 Verteilte Evolutionäre Algorithmen 55
3.6.2 Hybride Evolutionäre Algorithmen 57

4 Untersuchung der BGA-Mutationsverteilung 59
4.1 Definitionen und Notationen 59
4.2 BGA-Mutationsverteilung 61
4.3 Weitere Mutationsverteilungen 68
4.4 Vergleich der Mutationsverteilungen 69
4.4.1 Mutations-Selektions-Verfahren 70
4.4.2 Zielfunktion und Abstandsmaß 70
4.4.3 Effizienz 71
4.4.4 Robustheit 83

5 Wettbewerbsmodelle zur Strategieanpassung 87
5.1 Begriff der Suchstrategie 87
5.2 Ebenen der Strategieanpassung 88
5.2.1 Strategieanpassung auf Individuenebene 88
5.2.2 Strategieanpassung auf Populationsebene 89
5.3 Exkurs: Populationsdynamik 90
5.3.1 Basismodelle 90
5.3.2 Räuber-Beute-Interaktionen 93
5.3.3 Wettbewerbs-Interaktionen 94
5.4 Wettbewerb in Evolutionären Algorithmen 96
5.4.1 Konkurrierende Populationen 97
5.4.2 Bewertung der Populationen 101
5.4.3 Ressourcenzuweisung 102
5.4.4 Berechnung der Populationsgrößen 103
5.4.5 Migrationsverfahren 105
5.4.6 Strategiemodifikation 106
5.4.7 Selektion in Populationen variabler Größe 107
5.5 Instanzen des Wettbewerbsmodells 108
5.5.1 Spezifikation von Wettbewerbsverfahren 108
5.5.2 Spezifikation von Instanzen 109

6 Vergleich der Optimierungsverfahren 125
6.1 Auswertungsmethodik 125
6.2 Künstliche Testfunktionen 127
6.2.1 Kugelmodell (F1) 128

6.2.2 Gewichtetes Kugelmodell (F2) . . . 128
6.2.3 Schwefels Funktion (F3) . . . 129
6.2.4 Rosenbrocks Funktion (F4) . . . 130
6.2.5 Ackleys Funktion (F5) . . . 131
6.2.6 Flechter und Powells Funktion (F6) . . . 131
6.3 Spezifikation der Algorithmen . . . 132
6.4 Testumgebung . . . 136
6.5 Leistungsmaße . . . 137
6.5.1 Konvergenzverhalten . . . 137
6.5.2 Robustheit . . . 144
6.5.3 Skalierung . . . 145
6.6 Simulationsergebnisse . . . 147
6.6.1 Konvergenzverhalten . . . 148
6.6.2 Skalierung . . . 186

7 Mustererkennung als reale Anwendung 193
7.1 Grundlagen der Mustererkennung . . . 193
7.2 Mustermengen . . . 194
7.3 Vorverarbeitung . . . 196
7.4 Merkmalsgewinnung . . . 197
7.5 Klassifikationsansatz . . . 198
7.5.1 Allgemeiner Polynomansatz . . . 198
7.5.2 Polynomansatz mit erweitertem Optimierungsziel . . . 200
7.6 Vergleich zu anderen Verfahren . . . 201
7.6.1 Quadratmittelansatz . . . 201
7.6.2 Nearest-Neighbor-Klassifikatoren . . . 203
7.6.3 Multilayer-Perceptron (MLP) . . . 204
7.6.4 Klassifikationsergebnisse . . . 205

8 Parallelisierungsaspekte 207
8.1 Effizienzmaße der Parallelverarbeitung . . . 207
8.2 Parallelitätsebenen in Evolutionären Algorithmen . . . 208
8.2.1 Inselmodell . . . 208
8.2.2 Farming . . . 208
8.3 Parallele Umgebung . . . 209
8.3.1 Hardware . . . 209
8.3.2 Kommunikationssoftware . . . 209
8.4 Messungen . . . 210

9 Zusammenfassung 213

A Daten für Fletcher und Powells Funktion 215

B Notation 219

Literaturverzeichnis 227

Kapitel 1

Einleitung

Die natürliche Evolution kann als ein fortdauernder Anpassungsprozeß interpretiert werden, in dem Populationen durch die Mechanismen Variation und Selektion an ihre Umwelt angepaßt werden. Der Variationsprozeß setzt sich aus der Rekombination, die die Gene zweier Eltern mischt, und der Mutation, die zufällige Veränderungen auf den Genen bewirkt, zusammen. Die Selektion bevorteilt gut angepaßte Individuen einer Population bei der Produktion von Nachkommen und beeinflußt damit die Richtung der Entwicklung.

Dieser evolutionäre Anpassungsprozeß läßt sich als Optimierung der Individuen einer Population auffassen. Aus der Idee, die Konzepte der Evolution zur Lösung von Optimierungsproblemen einzusetzen, entstanden die Evolutionären Algorithmen. Evolutionäre Algorithmen modellieren den kollektiven Anpassungprozeß einer Population von Individuen. Jedes Individuum wird als ein Punkt im Suchraum der möglichen Realisierungen einer gegebenen Problemstellung interpretiert. Eine Population, die damit eine Menge von Suchpunkten repräsentiert, entwickelt sich durch Rekombination, Mutation und Selektion sukzessive in immer bessere Bereiche des Suchraums.

Die Evolution ist jedoch nicht auf eine einzige Population gleichartiger Individuen beschränkt. In der Natur sind vielmehr eine Vielzahl räumlich strukturierter Populationen und Arten zu beobachten, die miteinander interagieren. Diese Interaktionen, wie Symbiose, Wettbewerb und Räuber-Beute-Beziehungen, beeinflussen die Entwicklung der beteiligten Populationen. Analog zur Entwicklung der Individuen innerhalb einer Population sind auf einer höheren Ebene auch die Populationen einem Anpassungsprozeß ausgesetzt. Die Vielfalt, die durch das Vorhandensein unterschiedlicher Populationen gegeben ist, kann als Risikostreuung interpretiert werden, die den Fortbestand des Lebens sichert. Die Evolution ist so in Lage, robuster auf veränderte Umwelteinflüße zu reagieren.

Die Zielsetzung dieser Arbeit ist die Entwicklung und Integration populationsbasierter Wettbewerbsmodelle in Evolutionäre Algorithmen. Die Modellierung der in der natürlichen Evolution vorkommenden Wettbewerbsinteraktionen zwischen verschiedenen Populationen erlaubt den gleichzeitigen Einsatz unterschiedlicher Suchstrategien. Dabei werden die Populationsgrößen und Suchstrategien selbstorganisierend an die aktuellen Anforderungen, die sich im Laufe der Suche ergeben, angepaßt. Mit der gleichzeitigen Verwendung verschiedener Suchstrategien wird eine Erhöhung der Robustheit des Evolutionären Algorithmus angestrebt. Durch eine geeignete Kombination sich ergänzender Suchstrategien kann die Effizienz des Verfahrens durch Synergieeffekte gesteigert werden. Die Anpassung der Populationsgrößen bewirkt, daß der momentan erfolgversprechendsten Suchstrategie die meiste Rechenzeit zugewiesen wird. Das Wettbewerbskonzept ist unabhängig von der

Klasse, aus der das Optimierungsproblem stammt, auf alle populationsbasierten Suchverfahren anwendbar.

Die Integration des Wettbewerbskonzepts in Evolutionäre Algorithmen wird für die Klasse der statischen, kontinuierlichen Optimierungsprobleme ohne Nebenbedingung durchgeführt. Diese sogenannte kontinuierliche Parameteroptimierung nimmt in der Optimierung einen besonderen Stellenwert ein, da sich viele Problemstellungen auf sie abbilden lassen. Für diese Problemklasse wird der Breeder Genetic Algorithm (BGA) eingeführt, dessen Entwicklung einen weiteren Schwerpunkt dieser Arbeit bildet.

Die Leistungsbewertung der entwickelten Optimierungsverfahren basiert auf einer umfangreichen Simulationsreihe, die auf einer repräsentativen Menge von künstlichen Testfunktionen aus dem Bereich der kontinuierlichen Parameteroptimierung durchgeführt wird. Die Wettbewerbsmodelle werden mit verschiedenen Evolutionären Algorithmen und lokalen Suchverfahren bezüglich ihres Konvergenz- und Skalierungsverhaltens verglichen. Die praktische Bedeutung der hier behandelten Problem- und Algorithmenklasse wird am Beispiel der realen Anwendung 'Erkennung handgeschriebener Zeichen' gezeigt.

Die Entwicklung moderner Optimierungsverfahren ist eng mit der technischen Entwicklung des Computers verknüpft. Durch die Einführung leistungsstarker Rechner wurden in der numerischen Optimierung erhebliche Fortschritte erzielt. In besonderer Weise profitierten Evolutionäre Algorithmen von der Einführung der Parallelrechner. Aufgrund des populationsbasierten Ansatzes von Evolutionären Algorithmen, in denen mehrere Suchpunkte gleichzeitig bearbeitet werden können, kann durch die Verwendung von Parallelrechnern eine beträchtliche Beschleunigung erzielt werden. Am Beispiel der 'Erkennung handgeschriebener Zeichen' wird demonstriert, welche Beschleunigung durch die parallele Auswertung der Individuen eines Evolutionären Algorithmus erreicht werden kann.

Die Arbeit ist folgendermaßen gegliedert.

In Kapitel 2 werden fünf analytische und heuristische Suchverfahren der kontinuierlichen Parameteroptimierung vorgestellt. Die Simulationsergebnisse, die mit diesen Standardsuchverfahren erzielt werden, dienen als Vergleich, um die Leistungsfähigkeit der Evolutionären Algorithmen und insbesondere die des Wettbewerbskonzepts einzuordnen.

Kapitel 3 gibt einen Überblick über Evolutionäre Algorithmen zur kontinuierlichen Parameteroptimierung. Zunächst wird der allgemeine Aufbau eines Evolutionären Algorithmus spezifiziert. Außerdem werden typische Selektions- und Paarbildungsverfahren sowie allgemein verwendbare genetische Operatoren beschrieben. Basierend auf diesem allgemeinen Rahmen werden die beiden bekanntesten Vertreter dieser Algorithmenklasse, die Evolutionsstrategie und der Genetische Algorithmus, sowie der in dieser Arbeit entwickelte Breeder Genetic Algorithm eingeführt. Abschließend werden Erweiterungsmöglichkeiten der Evolutionären Algorithmen, die die parallele Ausführung identischer Populationen und die Integration lokaler Suchverfahren betreffen, aufgezeigt.

Kapitel 4 befaßt sich mit der Untersuchung der im BGA verwendeten Mutationsverteilung. Zunächst werden die Eigenschaften dieser sogenannten BGA-Mutationsverteilung herausgearbeitet. Dann wird sie bezüglich ihrer Effizienz und Robustheit mit anderen Mutationsverteilungen verglichen.

Das Wettbewerbskonzept für populationsbasierte Suchverfahren wird in Kapitel 5 eingeführt und in die im vorhergehenden Kapitel vorgestellten Evolutionären Algorithmen integriert.

Eine Leistungsbewertung der in dieser Arbeit vorgestellten Suchverfahren wird in Kapitel 6 anhand umfangreicher numerischer Simulationen durchgeführt. Dazu werden eine repräsentative Menge von Testfunktionen zusammengestellt und Gütemaße für Effizienz, Robustheit und Skalierbarkeit definiert.

Die Einsatzmöglichkeit des Wettbewerbskonzepts wird in Kapitel 7.1 anhand einer realen Problemstellung demonstriert. Die Problemstellung 'Erkennung handgeschriebener Zeichen' wird auf ein Problem der kontinuierlichen Parameteroptimierung abgebildet und durch einen Evolutionären Algorithmus mit Wettbewerbsmodell gelöst. Die dabei erzielten Resultate werden mit denen anderer Methoden, wie dem klassischen Polynomansatz, Nearest-Neighbour-Klassifikatoren und Neuronalen Netzen, verglichen.

Kapitel 8 geht auf die Parallelisierungsmöglichkeiten populationsbasierter Suchverfahren ein. Anhand des im vorhergehenden Kapitel behandelten Mustererkennungsproblems wird gezeigt, welche Beschleunigung durch die Verwendung eines Parallelrechners erzielt werden kann.

Kapitel 9 faßt die Beiträge dieser Arbeit zusammen.

Kapitel 2

Ableitungsfreie Optimierungsverfahren

2.1 Deterministische Verfahren

2.1.1 Quasi-Newton-Verfahren

Die Quasi-Newton-Verfahren, auch *Strategien variabler Metrik* genannt, zählen zu den verbreitetsten Verfahren im Bereich der kontinuierlichen Parameteroptimierung. Quasi-Newton-Verfahren basieren, wie auch das ursprüngliche Newton-Verfahren, auf der Taylorreihenentwicklung einer quadratischen Funktion. Die Taylorreihenentwicklung einer beliebig oft differenzierbaren Funktion kann folgendermaßen dargestellt werden:

$$f(\vec{x}+\vec{d}) = f(\vec{x}) + \vec{d}^T\vec{g}(\vec{x}) + \frac{1}{2}\vec{d}^T\mathbf{G}(\vec{x})\vec{d} + \ldots \tag{2.1}$$

mit $\vec{x}$ als Parameter- und $\vec{d}$ als Richtungsvektor. $\vec{g}(\vec{x})$ ist der Gradientenvektor und $\mathbf{G}(\vec{x})$ die Hesse-Matrix am Punkt $\vec{x}$. Die Terme höherer Ordnung sind für eine quadratische Funktion Null. Durch Differentiation nach $\vec{d}$ und Nullsetzen der Ableitung lassen sich die stationären Punkte einer Funktion zweiter Ordnung bestimmen.

$$\vec{g}(\vec{x}+\vec{d}) = \vec{g}(\vec{x}) + \mathbf{G}(\vec{x})\cdot\vec{d} \stackrel{!}{=} 0 \tag{2.2}$$

$$\vec{d} = \Leftrightarrow\frac{\vec{g}(\vec{x})}{\mathbf{G}(\vec{x})} \tag{2.3}$$

Gleichung 2.3 wird als *Newton-Bedingung* bezeichnet. Quasi-Newton-Methoden unterscheiden sich von den eigentlichen Newton-Verfahren darin, daß sie nicht mit der Hesse-Matrix $\mathbf{G}(\vec{x})$, sondern mit einer positiv definiten Matrix $\mathbf{H}$ arbeiten, die im Laufe der Suche an $\mathbf{G}^{-1}(\vec{x})$ approximiert wird.

Algorithmus 2.1 *Quasi-Newton-Verfahren*

QN($t^{\max}, \epsilon$, UPDATE)
1 $t \leftarrow 0$
2 **while** $t < t^{\max}$ AND GRADIENT($\vec{x}_t, \epsilon$) $\neq 0$
3 **do** $\vec{g}(\vec{x}_t) \leftarrow$ GRADIENT($\vec{x}_t, \epsilon$)
4 $\vec{d} \leftarrow \Leftrightarrow\mathbf{H_t} \cdot \vec{g}(\vec{x}_t)$ // Bestimmung der Suchrichtung
5 $\vec{x}_{t+1} \leftarrow$ LINMIN($\vec{x}_t, \vec{d}$) // eindimensionale Optimierung
6 $\mathbf{H}_{t+1} \leftarrow$ UPDATE($\mathbf{H}_t$)
7 $t \leftarrow t + 1$
8 **return** $\vec{x}_t$

Der Parameter ϵ legt die Genauigkeit, mit der der Gradient approximiert wird, fest. Die Prozedur LINMIN berechnet zu den beiden Eingabevektoren $\vec{x}$ und $\vec{d}$ und der Funktion f den Skalar λ, der $f(\vec{x} + \lambda \cdot \vec{d})$ minimiert, und liefert als Ergebnis $\vec{x} + \lambda \cdot \vec{d}$. Die verschiedenen Quasi-Newton-Verfahren unterscheiden sich im wesentlichen in der Aktualisierung der Matrix $\mathbf{H}$, also in der Prozedur UPDATE. Zu den bekanntesten zählen das *Davidon-Fletcher-Powell (DFP)* und das *Broyden-Fletcher-Goldfarb-Shanno (BFGS)*-Verfahren.

Die DFP-Formel lautet ([FP63]):

$$\mathbf{H}_{t+1}^{\text{DFP}} = \mathbf{H}_t + \frac{\delta\delta^T}{\delta^T\gamma} \Leftrightarrow \frac{\mathbf{H}_t\gamma\gamma^T\mathbf{H}_t}{\gamma^T\mathbf{H}\gamma} \tag{2.4}$$

Die BFGS-Formel lautet ([Bro70], [Fle70], [Gol70], [Sha70]):

$$\mathbf{H}_{t+1}^{\text{BFGS}} = \mathbf{H}_t + \left(1 + \frac{\gamma^T\mathbf{H}_t\gamma}{\delta^T\gamma}\right)\left(\frac{\delta\delta^T}{\delta^T\gamma}\right) \Leftrightarrow \left(\frac{\delta\gamma^T\mathbf{H}_t + \mathbf{H}_t\gamma\delta^T}{\delta^T\gamma}\right) \tag{2.5}$$

mit $\gamma = g_{t+1} \Leftrightarrow g_t$ und $\delta = x_{t+1} \Leftrightarrow x_t$.

Das BFGS-Verfahren stellt eine Verfeinerung des DFP-Verfahrens dar und gilt laut Fletcher ([Fle87]) als die derzeit beste Methode, um die Hesse-Matrix zu approximieren. Diese Variante wird für die Messungen in dieser Arbeit verwendet.

2.1.2 Verfahren der konjugierten Richtungen von Powell

Das Verfahren von Powell ([Pow64a], [Pow64b]) zählt zu den sogenannten *Direction-Set-Methoden*, die ohne Kenntnis der Ableitungen konjugierte Richtungen aufbauen und entlang dieser eindimensionale Optimierungsverfahren einsetzen.

Richtungen $\vec{d}_i$ werden als konjugiert bezüglich einer positiv definiten $(n \times n)$-Matrix $\mathbf{A}$ bezeichnet, wenn

$$\vec{d}_i^T \cdot \mathbf{A} \cdot \vec{d}_j = 0 \qquad \text{für} \quad 1 \leq i \neq j \leq n \tag{2.6}$$

gilt. Konjugierte Richtungen sind linear unabhängig. Setzt man für $\mathbf{A}$ die *Hesse-Matrix* ein, benötigt ein solches Verfahren zur Minimierung einer quadratischen Funktion den Einsatz von n eindimensionalen Optimierungen entlang der Richtungen $\vec{d}_i$. Powell zeigte, daß n Iterationen der in Algorithmus 2.2 vorgestellten Hauptschleife, und damit $n(n+1)$ eindimensionale Minimierungen, eine quadratische Funktion exakt minimieren ([Pow64a]).

Algorithmus 2.2 *Powell-Verfahren*

POW($t^{\max}, \tau$)
1 $t \leftarrow 0$
2 $\vec{d}_i \leftarrow \vec{e}_i \qquad i = 1, \ldots, n$
3 **while** $t < t^{\max}$
4 **do for** $i \leftarrow 1$ **to** n
5 **do** $\vec{x}_i \leftarrow \text{LinMin}(\vec{x}_{i-1}, \vec{d}_i)$ // eindimensionale Optimierung
6 **if** $2|f(\vec{x}_0) \Leftrightarrow f(\vec{x}_n)| \leq \tau \cdot (|f(\vec{x}_0)| + |f(\vec{x}_n)|)$
7 **then return** $\vec{x}_0$ // Terminierung
8 **for** $i \leftarrow 1$ **to** $n \Leftrightarrow 1$
9 **do** $\vec{d}_i \leftarrow \vec{d}_{i+1}$
10 $\vec{d}_n \leftarrow \vec{x}_n \Leftrightarrow \vec{x}_0$
11 $\vec{x}_0 \leftarrow \text{LinMin}(\vec{x}_n, \vec{d}_n)$ // eindimensionale Optimierung
12 $t \leftarrow t + 1$
13 **return** $\vec{x}_0$

Die Prozedur LinMin bezeichnet ein eindimensionales Minimierungsverfahren. Gegenüber dem Quasi-Newton-Verfahren benötigt das Verfahren von Powell jedoch ein erheblich genaueres Minimierungsverfahren, da die Bildung der konjugierten Richtungen in starkem Maße von dessen Güte abhängt. Powell verwendet eine Folge von quadratischen Interpolationen, die auf jeweils drei Funktionswerten basieren ([Pow64a]).

Durch den Verlust der ersten Richtung $\vec{d}_1$, die durch $\vec{x}_n \Leftrightarrow \vec{x}_0$ ersetzt wird, tendieren die Richtungen dazu, linear abhängig zu werden. Da die Suche in diesem Fall nur noch einen Unterraum von $\mathbb{R}^n$ erreicht, müssen geeignete Maßnahmen getroffen werden, die dies verhindern. Dies kann beispielsweise durch eine Reinitialisierung der Richtungen $\vec{d}_i$ auf die Anfangsrichtungen $\vec{e}_i$ nach jeweils n Iterationen der Hauptschleife geschehen.

Aus diesem Grund modifizierte Powell seinen obigen Basis-Algorithmus in zwei Punkten. Zum einen wird nicht mehr ausschließlich der erste Richtungsvektor $\vec{d}_1$, sondern der Richtungsvektor, mit dem die Zielfunktion den größten Abstieg erreicht, ersetzt. Zum anderen wird unter bestimmten Bedingungen ganz auf die Ersetzung eines vorhandenen Richtungsvektors verzichtet. Die von Powell aufgestellten Bedingungen sind von den Funktionswerten des ersten Punktes $\vec{x}_0$, des n-ten Punktes $\vec{x}_n$ und eines extrapolierten Punktes $2\vec{x}_n \Leftrightarrow \vec{x}_0$ abhängig. Mit den Größen

$$f_0 = f(\vec{x}_0) \qquad f_n = f(\vec{x}_n) \qquad f_e = f(2\vec{x}_n \Leftrightarrow \vec{x}_0) \tag{2.7}$$

werden folgende Bedingungen für den Verzicht auf eine Ersetzung aufgestellt:

1. $f_e \geq f_0$

2. $2(f_0 \Leftrightarrow 2f_n + f_e)\,((f_0 \Leftrightarrow f_n) \Leftrightarrow \Delta f)^2 \geq (f_0 \Leftrightarrow f_e)^2 \Delta f$

wobei Δf den Wert des größten Abstiegs entlang einer einzelnen Richtung durch die letzte Iteration bezeichnet. Weitere Erläuterungen zu diesen Bedingungen sind in [Pow64a] und [PTVF92] zu finden.

2.1.3 Downhill-Simplex-Methode

Die Downhill-Simplex-Methode basiert auf einem 1962 von Spendley, Hext und Himsworth ([SHH62]) entwickelten Grundverfahren, welches 1966 von Nelder und Mead weiterentwickelt wurde ([NM64]). Sie gilt als robustes ableitungsfreies Verfahren für mehrdimensionale Optimierungsprobleme. Im Gegensatz zu den meisten anderen mehrdimensionalen Optimierungsverfahren nutzt diese Methode kein eindimensionales Optimierungsverfahren, sondern stellt ein eigenständiges, in sich abgeschlossenes Verfahren dar. Obwohl die Methode ebenso wie die bekannte *Simplex-Methode* der linearen Programmierung auf dem geometrischen Konzept des Simplex basiert, hat sie mit ihr ansonsten nichts gemein.

Ein (regulärer) Simplex ist eine Menge von $n+1$ (äquidistanten) Punkten im $\mathbb{R}^n$. Für $n=1$ ist der Simplex eine Strecke, für $n=2$ ein (gleichseitiges) Dreieck und für $n=3$ ein Tetraeder. Für ein beliebiges n ist der Simplex ein Polyeder.

Die Downhill-Simplex-Methode startet mit einem beliebigen Punkt im n-dimensionalen Suchraum. Um diesen Startpunkt wird der aus n weiteren Punkten bestehende initiale Simplex gebildet. Die Suche besteht nun darin, den Simplex durch sukzessive Anwendung der drei Operatoren *Reflektion*, *Kontraktion* und *Expansion* zu verformen und ihn so in bessere Bereiche des Suchraums zu bewegen. Jeder der Operatoren ersetzt den Punkt mit dem höchsten (schlechtesten) Funktionswert $\vec{x}_h$. Im folgenden bezeichnet $\vec{x}_l$ den Punkt mit dem niedrigsten (besten) Funktionswert und $\bar{\vec{x}}$ den Schwerpunkt aller Punkte mit Ausnahme von $\vec{x}_h$. Die Reflektions-Operator bildet einen neuen Punkt $\vec{x}^*$, indem er $\vec{x}_h$ um $\bar{\vec{x}}$ reflektiert. Anschließend wird $\vec{x}_h$ durch $\vec{x}^*$ ersetzt.

$$\vec{x}^* = (1+\alpha)\bar{\vec{x}} \Leftrightarrow \alpha\vec{x}_h \tag{2.8}$$

Falls die Reflektion zu einer Verbesserung führt, wird der aus der Reflektion resultierende Punkt $\vec{x}^*$ durch folgende Beziehung expandiert:

$$\vec{x}^{**} = \gamma\vec{x}^* + (1 \Leftrightarrow \gamma)\bar{\vec{x}} \tag{2.9}$$

Falls der neue Punkt besser als der bislang beste Punkt ist, wird der schlechteste durch den neuen ersetzt und der Schritt wiederholt. Führt die Reflektion zu keiner Verbesserung, wird der Simplex durch eine Kontraktion geschrumpft.

$$\vec{x}^{**} = \beta\vec{x}_h + (1 \Leftrightarrow \beta)\bar{\vec{x}} \tag{2.10}$$

Algorithmus 2.3 *Downhill-Simplex-Methode*

DHS$(\alpha, \beta, \gamma, \lambda, t^{\max}, \tau)$

1 $t \leftarrow 0$ // Initialisierung
2 $\vec{x}_i \leftarrow \vec{x}_0 + \lambda \cdot \vec{e}_i \qquad i = 1, \ldots, n$
3 **while** $(t < t^{\max})$ AND $\left(2\frac{|f(\vec{x}_h) - f(\vec{x}_l)|}{|f(\vec{x}_h)| + |f(\vec{x}_l)|} < \tau\right)$
4 **do** $\vec{x}^* \leftarrow (1+\alpha)\bar{\vec{x}} \Leftrightarrow \alpha\vec{x}_h$ // Reflektion
5 **if** $f(\vec{x}^*) < f(\vec{x}_l)$
6 **then** $\vec{x}^{**} = \gamma \cdot \vec{x}^* + (1 \Leftrightarrow \gamma) \cdot \bar{\vec{x}}$ // Expansion
7 **if** $f(\vec{x}^{**}) < f(\vec{x}_l)$
8 **then** $\vec{x}_h \leftarrow \vec{x}^{**}$
9 **else** $\vec{x}_h \leftarrow \vec{x}^*$
10 **else if** $f(\vec{x}^*) > f(\vec{x}_i), \forall i \neq h$
11 **then if** $f(\vec{x}^*) \leq f(\vec{x}_h)$

```
12                          then x_h ← x*
13                        x** ← βx_h + (1 ⇔ β)x̄          // Kontraktion
14                        if f(x**) > f(x_h)
15                          then x_i ← (x_i + x_l)/2     ∀i
16                          else x_h ← x**
17                  else  x_h ← x*
18        t ← t + 1
19  return x_l
```

Nelder und Mead testeten eine Vielzahl von Parameterkombinationen und initialen Simplizes anhand von drei Testfunktionen. Als beste Parameterkombination stellte sich $\alpha = 1.0, \beta = 0.5$ und $\gamma = 2.0$ heraus ([NM64]).

Die Initialisierung des Simplex stellte sich als kritische Größe heraus. Während ein zu großer Simplex aufgrund fehlender Verbesserung zu frühzeitiger Stagnation führt, konvergiert ein zu kleiner Simplex nur sehr langsam.

2.2 Stochastische Verfahren

Im folgenden werden zwei adaptive Zufallssuchverfahren vorgestellt, in denen Schrittweite und Suchrichtung während der Suche an die Topologie des Suchraums angepaßt werden. Erste Verfahren dieser sogenannten *adaptiven Schrittweite-Methoden* wurden 1968 von Schumer und Steiglitz eingeführt ([SS68]).

Der Rahmen einer allgemeinen Zufallssuche ist in Algorithmus 2.4 dargestellt. Die Objektvariablen werden während der Suche mit Hilfe von Zufallsvariablen modifiziert. Die speziellen Zufallssuchverfahren unterscheiden sich in der Art, mit der die Zufallszahlen generiert und die Objektvariablen modifiziert werden.

Algorithmus 2.4 *Konzeptioneller Algorithmus einer Zufallssuche*

```
RandomSearch(D_ξ, D_x, AdaptStepsize, μ_0, TermCond)
1  t ← 0;   x_t ← x_0;   μ_t ← μ_0;                    // Initialisierung
2  while TermCond ≠ true
3     do ξ_t ← D_ξ(μ_t)                  // Generierung von Zufallsvariablen
4        x_{t+1} ← D_x(x_t, ξ_t)             // Setzen der Objektvariablen
5        μ_{t+1} ← AdaptStepsize              // Schrittweitenanpassung
6        t ← t + 1
7  return x_{t-1}
```

Gesucht wird ein Punkt $\vec{x}$ aus dem Suchraum $S \subseteq \mathbb{R}^n$, der eine Funktion f minimiert. $\vec{\mu}_t$ ist ein Wahrscheinlichkeitsmaß einer auf $\mathbb{R}^n$ definierten Wahrscheinlichkeitsverteilung. Die Wahrscheinlichkeitsverteilung, mit der aus $\vec{\mu}_t$ die Zufallsvariable $\vec{\xi}_t$ generiert wird, wird durch die Abbildung $D_\xi : R^n \subseteq \mathbb{R}^n$ spezifiziert. Die Abbildung $D_x : S \times R^n$ definiert, wie aus dem aktuellen Punkt $\vec{x}_t$ und der Zufallsvariablen $\vec{\xi}_t$ ein neuer Punkt $\vec{x}_{t+1}$ bestimmt wird.

2.2.1 Verfahren von Solis und Wets

Die 1981 von Solis und Wets entwickelte Optimierungsmethode ist ein adaptives Zufallssuchverfahren, in dem die Schrittweite während der Suche in Abhängigkeit von Erfolg und Nichterfolg angepaßt wird ([SW81]).

Algorithmus 2.5 *Verfahren von Solis und Wets*

SW$(\rho_0, \rho_{lb}, ex, ct, s_{ex}, f_{ct})$

```
 1  sc ← fc ← t ← 0;                                   // Initialisierung
 2  b⃗ ← 0⃗
 3  while ρ_t > ρ_lb   AND   t < 1000 · n
 4      do t ← t + 1
 5          switch                                     // Schrittweitenanpassung
 6            case sc > s_ex :
 7                 ρ_t ← ρ_{t−1} · ex
 8            case fc > f_ct :
 9                 ρ_t = ρ_{t−1} · ct
10            case default :
11                 ρ_t ← ρ_{t−1}
12          ξ⃗_t ← x⃗_t + b⃗ + U([⇔ρ_t, ρ_t])          // Generierung von Zufallsvariablen
13          switch                                     // Modifikation der Objektvariablen
14            case f(ξ⃗_t) < f(x⃗_t) :
15                 x⃗_{t+1} ← ξ⃗_t
16                 b⃗ ← 0.4 · (ξ⃗ ⇔ x⃗_t) + 0.2 · b⃗
17            case f(2x⃗_t ⇔ ξ⃗_t) < f(ξ⃗_t) :
18                 x⃗_{t+1} ← 2ξ⃗_t ⇔ ξ⃗_t
19                 b⃗ ← b⃗ ⇔ 0.4 (x⃗_t ⇔ x⃗_t)
20                 sc ← sc + 1
21                 fc ← 0
22            case default :
23                 sc ← 0
24                 fc ← fc + 1
25                 b⃗ ← 0.5 · b⃗
26  return x⃗_t
```

Der Bereich der multivariaten Zufallsvariablen $\vec{\xi}_t$ ist durch den Parameter ρ_t festgelegt, der im Laufe der Suche angepaßt wird. Die untere Schranke des Schrittweitenparameters ρ_t ist durch den Parameter ρ_{lb} vorgegeben. In $\vec{b}_t$ werden die Richtungen aus erfolgreichen Schritten akkumuliert. Die beiden Zähler sc und fc halten die Anzahl erfolgreicher und fehlerhafter Schritte fest. Sobald diese Zähler s_{ex} bzw. f_{ct} erreichen, wird die Schrittweite ρ_t um den Faktor ex vergrößert bzw. um ct verringert.

Solis und Wets schlagen für die exogenen Parameter folgende Belegung vor: s_{ex}=5, $f_{ct} = 3$, $ex = 2$, $ct = 0.5$ und $\rho_0 = 1$. Die untere Schranke des Schrittweitenparameters ρ_{lb} wird in Abhängigkeit der gewünschten Genauigkeit, mit der das Optimum approximiert werden soll, gewählt.

2.2.2 Dynamic-Hill-Climbing

Das *Dynamic-Hill-Climbing*-Verfahren wurde 1993 von Yuret und de la Maza ([YdlM93], [dlMY94], [Yur94]) entwickelt. Zur Lösung multimodaler Funktionen schlagen sie vor, das in Algorithmus 2.6 beschriebene lokale Optimierungsverfahren iteriert anzuwenden.

Algorithmus 2.6 *Dynamic-Hill-Climbing*

```
DHC(τ)
 1  t ← 0                                                    // Initialisierung
 2  v_0 ← u_0 ← 0
 3  while |v_t| ≥ τ
 4      do c ← 0
 5          while f(x_t + v_t) ≥ f(x_t)   AND   c < 2 · n
 6              do v_t ← RandomVector(v_t)      // Generierung von Zufallsvariablen
 7                 c ← c + 1
 8                                  // Schrittweitenanpassung und Setzen der Objektvariablen
 9          if f(x_t + v_t) ≥ f(x_t)
10            then
11                 v_{t+1} ← v_t/2
12            else if c = 0
13                   then
14                        x_{t+1} ← x_t + v_t
15                        u_{t+1} ← u_t + v_t
16                        v_{t+1} ← 2 · v_t
17                   else if f(x_t + u_t + v_t) ≤ f(x_t)
18                          then
19                               x_{t+1} ← x_t + u_t + v_t
20                               u_{t+1} ← u + v_t
21                               v_{t+1} ← 2 · u_t
22                          else
23                               x_{t+1} ← x_t + v_t
24                               u_{t+1} ← v_t
25                               v_{t+1} ← 2 · v_t
26          t ← t + 1
27  return x_{t−1}
```

Solange der Schrittvektor $\vec{v}$, der als bester Schätzwert des Gradienten angenommen wird, zu Verbesserungen des Zielfunktionswerts führt, wird seine Länge verdoppelt. Die erfolgreichen Schritte werden in Vektor $\vec{u}$ akkumuliert. Falls der Schrittvektor $\vec{v}$ keine weitere Verbesserung erzielt, geht der Algorithmus in einen Zufallstestmodus über, in dem $\vec{v}$ solange zufällig gesetzt wird, bis eine Verbesserung eintritt oder eine obere Schranke für die Anzahl der Tests erreicht wird. Falls nach mehreren Versuchen eine erfolgreiche Richtung gefunden wurde, wird die Kombination $\vec{u}+\vec{v}$ als neue Schätzung für den Gradienten verwendet. Falls diese neue Schätzung zu einer Verbesserung führt, werden die erfolgten Schritte entlang dieser neuen Richtung $\vec{u}+\vec{v}$ in $\vec{u}$ akkumuliert. Führt ein Schritt $\vec{u}+\vec{v}$ zu keiner Verbesserung, wird der neue Schrittvektor $\vec{v}$ zum Schätzvektor des Gradienten.

Kapitel 3

Evolutionäre Algorithmen

Evolutionäre Algorithmen (EA) stellen eine Klasse von stochastischen Optimierungsverfahren dar, die sich an der Funktionsweise der natürlichen Evolution orientiert. Sie basieren auf einer mehr oder weniger abstrakten Modellierung der evolutionären Mechanismen.

Wie die Evolution läßt sich ein Evolutionärer Algorithmus als ein kollektiver Suchprozeß auffassen, in dem sich eine *Population*, die sich aus mehreren *Individuen* zusammensetzt, in einer bestimmten Umgebung entwickelt.

Ein Individuum repräsentiert dabei eine mögliche Lösung des Problems. Die *Fitness* eines Individuums spiegelt die Qualität der entsprechenden Lösung wider. Durch die stochastischen Prozesse *Rekombination*, *Mutation* und *Selektion* entwickelt sich eine einmal initialisierte Population in neue Bereiche des Suchraums. Das Grundkonzept eines EA basiert darauf, daß die Wahrscheinlichkeit, mit Hilfe der *genetischen Operatoren* Rekombination und Mutation bessere Lösungen zu erzeugen, mit der Fitness der Individuen korreliert ist. Aus diesem Grund wird die Anzahl der Nachkommen eines Individuums in Abhängigkeit von dessen Fitness festgelegt. Dieser Selektionsprozeß, der beispielsweise Umweltselektion oder Geburtenüberschuß modelliert, drängt die Population sukzessive in bessere Bereiche des Suchraums und gibt dem stochastischen Prozeß damit eine Richtung. Die genetischen Operatoren werden aufgrund der Anzahl von Elternindividuen, aus denen ein Nachkomme erzeugt wird, in *asexuelle* und *sexuelle* Operatoren unterschieden.

Mutationsoperatoren, die zur Klasse der asexuellen genetischen Operatoren zählen, modifizieren einzelne Individuen nach einem noch näher zu spezifizierenden stochastischen Verfahren. Rekombinationsoperatoren modellieren die zwei- bzw. mehrgeschlechtliche Vermehrung und werden daher als (multi-) sexuelle genetische Operatoren bezeichnet.

Die populationsbasierte Suche stellt einen der wesentlichen Unterschiede zu anderen Optimierungsverfahren dar. Während die meisten Optimierungsverfahren die Suchpunkte im Laufe der Suche sequentiell bestimmen und auswerten, können die Individuen einer EA-Population gleichzeitig ausgewertet werden. Die daraus resultierenden Parallelisierungsmöglichkeiten von Evolutionären Algorithmen werden in Kapitel 8 ausführlich behandelt.

Historisch betrachtet lassen sich Evolutionäre Algorithmen in die drei Hauptzweige, *Evolutionary Programming* (EP), *Evolutionsstrategien* (ES) und *Genetische Algorithmen* (GA) unterteilen. Diese drei EA-Zweige wurden in den sechziger und siebziger Jahren unabhängig von amerikanischen und deutschen Wissenschaftlern entwickelt. Fogel und andere entwickelten als erste das Evolutionary Programming, das nicht mit *Genetic Programming* zu verwechseln ist ([FOW66]). Fast zeitgleich wurden in Deutschland von Re-

chenberg ([Rec73]) und Schwefel die Evolutionsstrategien und in den USA von Holland ([Hol75]) die Genetischen Algorithmen entwickelt.

Zur Beschreibung der verschiedenen EA-Arten wird zunächst die allen gemeinsame Struktur eines Evolutionären Algorithmus eingeführt und eine Menge von allgemeinen Operatoren beschrieben, die sich keiner einzelnen EA-Art zuordnen lassen. Auf diese Weise können die Gemeinsamkeiten und Unterschiede der verschiedenen EA-Varianten deutlich dargestellt werden.

Basierend auf diesem allgemeinen Evolutionären Algorithmus werden die Evolutionsstrategien und die Genetischen Algorithmen ausführlich beschrieben. Das ursprüngliche Evolutionary Programming wurde aufgrund von Problemen, die die Abhängigkeit der Schrittweite von der Fitness betreffen, von Fogels Sohn ([Fog92]) zu dem heutigen Meta-Evolutionary Programming weiterentwickelt. Aufgrund der starken Ähnlichkeit dieses Verfahrens zu den Evolutionsstrategien wird auf eine Beschreibung dieses Verfahrens verzichtet. Ein Vergleich zwischen den beiden EA-Klassen ist in [BRS93] und [Bäc96] zu finden.

Ebenfalls ausführlich beschrieben wird der von uns entwickelte Breeder Genetic Algorithm (BGA), der Elemente aus der Evolutionsstrategie und dem Genetischen Algorithmus vereinigt ([MSV93b]).

3.1 Allgemeiner Aufbau

3.1.1 Struktur eines Evolutionären Algorithmus

Die Struktur eines allgemeinen Evolutionären Algorithmus ist in Abbildung 3.1 dargestellt. Das Blockdiagramm zeigt die Abfolge von Operationen, die eine Population während einer Optimierung durchläuft.

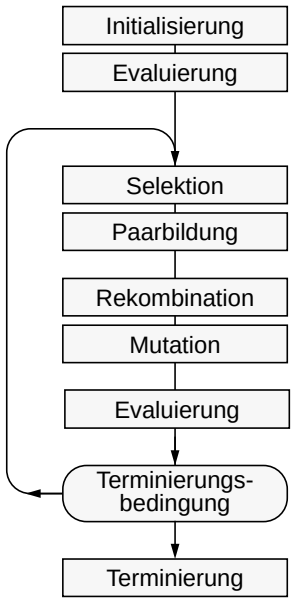

Abbildung 3.1: Struktur eines Evolutionären Algorithmus

1. **Initialisierung:** Die Individuen einer Population werden problem-spezifisch initialisiert.
2. **Evaluierung:** Die Individuen werden bewertet und mit Fitnesswerten versehen.
3. **Selektion:** Die Individuen werden aufgrund ihrer Fitnesswerte ausgewählt, um Nachkommen zu erzeugen.
4. **Paarbildung:** Aus den selektierten Individuen werden Paare (oder Tupel) von Individuen gebildet.
5. **Rekombination:** Aus jedem Paar bzw. Tupel von Elternindividuen wird *ein*[1] neues Individuum zusammengestellt.
6. **Mutation:** Jedes dieser Individuen wird einer zufälligen Änderung unterworfen.
7. **Terminierungsbedingung:** Der Algorithmus testet die Terminierungsbedingung nach Evaluierung der neu generierten Individuen und fährt mit der Selektion fort, falls die Terminierungsbedingung nicht erfüllt ist.

3.1.2 Formale Definition

Die folgende formale Definition eines Evolutionären Algorithmus basiert auf einer Formalisierung, die erstmals von Bäck in seiner Dissertation eingeführt worden ist ([Bäc96]). Mit den hier eingeführten Erweiterungen dieser Formalisierung lassen sich neben den einfachen Evolutionären Algorithmen auch hierachische Mehrpopulationsmodelle und hybride Evolutionäre Algorithmen miteinbeziehen.

Definition 1 (Evolutionärer Algorithmus) *Ein* Evolutionärer Algorithmus (EA) *ist definiert als:*

$$EA = (\mathbb{C}, \mathbb{F}, \mathbb{S}, \mathbb{P}, \mathbb{R}, \mathbb{M}, \mathbb{G}, \mathbb{I}, \mathbb{T}, \lambda) \tag{3.1}$$

wobei

$\mathbb{C} = (I, \mathbb{E}, \mathbb{D})$		*Kodierung*
$\mathbb{F}:$	$I \to \mathbb{R}$	*Fitnessfunktion*
$\mathbb{S}_{\Theta_{\mathbb{S}}}:$	$I^{\lambda+\mu} \to I^{\mu}$	*Selektion*
$\mathbb{P}_{\Theta_{\mathbb{P}}}:$	$I^{\mu} \to (I^{\rho})^{\lambda}$	*Paarbildung*
$\mathbb{R}_{\Theta_{\mathbb{R}}}:$	$(I^{\rho})^{\lambda} \to I^{\lambda}$	*Rekombination*
$\mathbb{M}_{\Theta_{\mathbb{M}}}:$	$I^{\lambda} \to I^{\lambda}$	*Mutation*
$\mathbb{G}:$	$I^{\lambda} \to I^{\lambda}$	*Generationsübergang*
$\mathbb{I}:$	$D \to I^{\lambda}$	*Initialisierung*
$\mathbb{T}:$	$I^{\lambda} \to \{\mathsf{true}, \mathsf{false}\}$	*Terminierungsbedingung*
λ		*Populationsgröße*

bezeichnen.

[1] Einige EA-Implementierungen ([Gre90]) erzeugen aus einem Elternpaar *zwei* Nachkommen. Da die Erzeugung *komplementärer* Nachkommen jedoch zu unbeabsichtigten Nebeneffekten führt, wird dieser Fall hier explizit ausgeschlossen.

Die Kodierung ($\mathbb{C}$) *setzt sich aus der Definition des Raumes der Individuen I, der Kodierungsabbildung* $\mathbb{E} : D \to I$ *und der Dekodierungsabbildung* $\mathbb{D} : I \to D$ *zusammen, wobei* D *der Suchraum des Optimierungsproblems ist. Die* Fitnessfunktion ($\mathbb{F}$) *weist jedem Individuum* $\vec{a} \in I$ *eine reelle Zahl zu, die der Güte der durch das Individuum spezifizierten Lösung des Optimierungsproblems entspricht. Sie ist definiert durch:*

$$\mathbb{F}(\vec{a}) = f(\mathbb{D}(\vec{a})) \quad mit \quad \vec{a} \in I \quad und\ Zielfunktion\ f : D \to \mathbb{R} \tag{3.2}$$

Die Menge aller Nachkommen zum Zeitpunkt t wird als Population $P(t) = (\vec{a}_1(t), \ldots, \vec{a}_\lambda(t))$ *bezeichnet .*

Die Selektion ($\mathbb{S}$) *wählt aus einer Menge von Individuen* μ *Individuen aufgrund ihrer Fitnesswerte aus, um aus ihnen neue Individuen zu generieren. Die Menge, aus der selektiert wird, enthält obligatorisch die Nachkommen der vorhergehenden Generation. Zusätzlich kann ab der zweiten Generation auch das beste oder alle Eltern-Individuen der vorhergehenden Generation enthalten sein. Die Menge der selektierten Individuen wird mit* $P^{\mathbb{S}}(t)$ *bezeichnet. Die* Paarbildung ($\mathbb{P}$) *stellt aus diesen* μ *Individuen* λ ρ*-Tupel von Individuen zusammen. Die* Rekombination ($\mathbb{R}$) *wendet den Rekombinationsoperator*

$$\dot{\mathbb{R}}_{\Theta_{\mathbb{R}}} : I^\rho \to I \tag{3.3}$$

auf jedes der λ ρ*-Tupel an und erzeugt so* λ *neue Individuen. Die* Mutation ($\mathbb{M}$) *wendet den Mutationsoperator*

$$\dot{\mathbb{M}}_{\Theta_{\mathbb{M}}} : I \to I \tag{3.4}$$

auf jedes dieser λ *neu erzeugten Individuen an.* $\Theta_{\mathbb{S}} \subset \mathbb{R}, \Theta_{\mathbb{P}} \subset \mathbb{R}, \Theta_{\mathbb{R}} \subset \mathbb{R}$ *und* $\Theta_{\mathbb{M}} \subset \mathbb{R}$ *sind Parameterlisten, die zur Steuerung der entsprechenden Operatoren verwendet werden.*

Der Generationenübergang ($\mathbb{G}$) *definiert, wie eine Population P durch Anwendung der oben beschriebenen Operatoren verändert wird.*

$$\mathbb{G}(P(t)) = \begin{cases} \mathbb{M}(\mathbb{R}(\mathbb{P}(\mathbb{S}(P(t))))) & : \quad t = 0 \\ \mathbb{M}(\mathbb{R}(\mathbb{P}(\mathbb{S}(P(t) \cup P^{\mathbb{S}}(t \Leftrightarrow 1))))) & : \quad t > 0 \end{cases} \tag{3.5}$$

Die Initialisierung ($\mathbb{I}$) *weist den Individuen der Anfangspopulation P(0) Werte aus I zu. Die* Terminierungsbedingung ($\mathbb{T}$) *entscheidet aufgrund der aktuellen Population, ob die Suche fortgesetzt oder beendet wird.*

Die obige Definition deckt alle EA-Varianten, die auf dem sogenannten *Generationenmodell* basieren, ab. Die Unterschiede zwischen den verschiedenen Varianten bestehen in der Art der Kodierung, der Selektion und den genetischen Operatoren. Die unterschiedliche Notation für die *Blöcke* der genetischen Operatoren ($\mathbb{M}$ und $\mathbb{R}$) und den eigentlichen genetischen Operatoren ($\dot{\mathbb{M}}$ und $\dot{\mathbb{R}}$) wird im folgenden, sofern eine Unterscheidung nicht erforderlich ist, aus Gründen der Übersichtlichkeit vernachlässigt. Der *Mutationsblock* $\mathbb{M}_{\Theta_{\mathbb{M}}}$, der sich auf die gesamte Population bezieht, wendet den Mutationsoperator $\dot{\mathbb{M}}_{\Theta_{\mathbb{M}}}$ auf jedes ihrer Individuen an. Dasselbe gilt für den *Rekombinationsblock* $\mathbb{R}$ und den *Rekombinationsoperator* $\dot{\mathbb{R}}$. Die Populationsgröße zählt ebenso wie die operatorspezifischen Parameter zu den externen Parametern eines EA.

Die Terminierungsbedingung stellt wie überall in der Optimierung einen kritischen Punkt dar, da in der Regel nicht bekannt ist, ob der Algorithmus durch die Ausführung

weiterer Generationen Verbesserungen erzielen kann. Eine einfache Terminierungsbedingung besteht darin, die Anzahl der Generationen festzulegen, nach der der Algorithmus abbricht. Ebenso kann der Abbruch von statistischen Werten, wie der genotypischen Varianz der Population, abhängig gemacht werden. Die Wahl des Terminierungskriteriums ist letztlich von der jeweiligen Anwendung abhängig. Oft ist eine schnelle suboptimale Lösung sinnvoller als eine aufwendig berechnete optimale Lösung des Problems.

Definition 2 (Population) *Als* Population *zum Zeitpunkt t wird die Menge der in der t-ten Generation neu erzeugten Individuen eines EA bezeichnet.*

$$P(t) = \{\vec{a}_i(t)\}_{i=1}^{\lambda} \tag{3.6}$$

Verschiedene Varianten von EA arbeiten mit mehreren Populationen, die miteinander interagieren. Die Gesamtheit aller Individuen kann dann je nach Interpretation und Hierarchiestufe als *Weltpopulation* oder *Inselpopulation* bezeichnet werden. Der Begriff *Subpopulation* bezeichnet dagegen eine Teilmenge einer Population.

Definition 3 (Evolution einer Population) *Sei* $\mathbb{G} : I^{\lambda} \to I^{\lambda}$ *der Generationsübergang eines EA mit der Anfangspopulation* $P(0) \in I^{\lambda}$*. Dann ist*

$$P(t+1) = \mathbb{G}(P(t)) \qquad t = 0, \ldots, \tau \tag{3.7}$$

die Evolution der Population $P(0)$ *bis zur Generation* τ*.*

Die Ausführung eines EA entspricht damit der Evolution einer Anfangspopulation. Die Population $P(\tau)$ ist das Ergebnis der Optimierung. Einem Individuum $\vec{a} \in I$ wird durch die Fitnessfunktion ein Fitnesswert zugeordnet. Um zwischen dem Individuum, das ausschließlich die Objektparameter enthält, und der Einheit von Objektvariablen mit zugehöriger Fitness zu unterscheiden, wird der Begriff des *situierten Individuums* eingeführt. Da das Auswahlkriterium der Selektion auf der Fitness der Individuen basiert, wird die Selektion auf die *situierte Population* angewendet.

Definition 4 (Situiertes Individuum) *Sei* $\vec{a}(t) \in I$ *ein Individuum in Generation* t*. Dann ist*

$$\hat{\vec{a}}(t) = (\vec{a}(t), \mathbb{F}(\vec{a}(t))) \tag{3.8}$$

das situierte Individuum *bzgl.* $\vec{a}(t)$ *zum Zeitpunkt* t*.*

Definition 5 (Situierte Population) *Sei* $P(t) = \{\vec{a}_i(t)\}_{i=1}^{\lambda}$ *die Population in Generation* t *und* $\hat{\vec{a}}_i(t)$ *das situierte Individuum bzgl.* $\vec{a}_i(t)$*. Dann ist*

$$\hat{P}(t) = \left\{\hat{\vec{a}}_i(t)\right\}_{i=1}^{\lambda} \qquad \forall \vec{a}_i(t) \in P(t) \tag{3.9}$$

die situierte Population *zum Zeitpunkt* t*.*

Aus der bisherigen Beschreibung kann ein allgemeiner EA skizziert werden. In Algorithmus 3.1 bezeichnet $P(t) = \{\vec{a}_1(t), \ldots, \vec{a}_\lambda(t)\}$ die Population in Generation t, wobei $\vec{a}_i(t)$ ein Individuum aus I darstellt. $\hat{P}(t)$ ist die situierte Population bzgl. $P(t)$. $\hat{P}^{\mathbb{S}}(t) = \{\hat{\vec{a}}_{i_1}(t), \ldots, \hat{\vec{a}}_{i_\mu}(t)\}$ mit $i_k \in \{1, \ldots, \lambda\}$ ist die Menge der selektierten Individuen, wobei μ die Anzahl der selektierten Individuen angibt. $Q(t) = \{P_1^{\mathbb{P}}(t), \ldots, P_\lambda^{\mathbb{P}}(t)\}$ entspricht der Menge der Eltern-Tupel. Jedes Eltern-Tupel $P_j^{\mathbb{P}}(t) = \{\vec{a}_{i_1}(t), \ldots, \vec{a}_{i_\rho}(t)\}$ mit $i_k \in \{1, \ldots, \mu\}$ setzt sich aus Individuen aus $P^{\mathbb{S}}(t)$ zusammen. ρ gibt die Anzahl der Eltern eines Individuums an.

Algorithmus 3.1 *Skizze eines Evolutionären Algorithmus*

EA(f)
1 $t \leftarrow 0$
2 $P(0) = \mathbb{I}(D)$ Initialisierung
3 $\hat{P}(0) \leftarrow \{\vec{a}_i(0), \mathbb{F}(\vec{a}_i(0))\}_{i=1}^{\lambda}$ Evaluierung
4 $\hat{P}^{\mathbb{S}}(0) = \emptyset$
5 **while** $\mathbb{T}(\hat{P}(t)) \neq$ true
6 **do** $t \leftarrow t + 1$
7 $\hat{P}^{\mathbb{S}}(t) \leftarrow \mathbb{S}(\hat{P}(t \Leftrightarrow 1) \cup \hat{P}^{\mathbb{S}}(t \Leftrightarrow 1))$ Selektion
8 $Q(t) \leftarrow \mathbb{P}(\hat{P}^{\mathbb{S}}(t))$ Paarbildung
9 $P'(t) \leftarrow \mathbb{R}(Q(t))$ Rekombination
10 $P(t) \leftarrow \mathbb{M}(P'(t))$ Mutation
11 $\hat{P}(t) \leftarrow \{\vec{a}_i(t), \mathbb{F}(\vec{a}_i(t))\}_{i=1}^{\lambda}$ Evaluierung
12 **return** $\vec{a}_i(t)$ *mit* $\mathbb{F}(\vec{a}_i(t)) < \mathbb{F}(\vec{a}_j(t)) \quad \forall j \neq i$

3.1.3 Genetische und populationsbasierte Operatoren

Rekombination

Ein Rekombinationsoperator generiert aus mehreren Individuen ein neues Individuum. Man klassifiziert Rekombinationsoperatoren in Abhängigkeit von der Anzahl der Eltern pro Individuum in *sexuelle* und *multisexuelle* Schemata. Als sexuell wird ein Rekombinationsoperator bezeichnet, wenn genau zwei Eltern an einer Paarung beteiligt sind. Wird ein Individuum aus mehr als zwei Eltern erzeugt, handelt es sich um einen multisexuellen Rekombinationsoperator. Der Spezialfall des multisexuellen Rekombinationsschemas, in dem ein Nachkomme aus allen Individuen erzeugt wird, wird als *panmiktisch* bezeichnet. Panmiktische Rekombinationsarten nennt man auch *Genpool*-Rekombination ([Bäc96], [VMC95]).

Rekombinationsoperatoren nutzen implizit die Verteilung der Population, um neue Individuen zu erzeugen. Bezüglich ihrer Wirkungsweise decken sie ein weites Spektrum ab, das von explorierender Suche bis hin zur ausnutzenden (engl.: *exploiting*) Suche reicht. Das Suchverhalten eines Rekombinationsoperators ist immer in starkem Maße durch die Kodierung des Problems bestimmt. So weist ein Rekombinationsoperator eines GA, der in der Lage ist, die Building-Block-Eigenschaft einer Zielfunktion auszunutzen und damit gute Teillösungen zu einer besseren Gesamtlösung zusammenzuführen, ein ausnutzendes Suchverhalten auf. Derselbe Rekombinationsoperator eines GA erzeugt bei der kontinuierlichen Parameteroptimierung aufgrund der Kodierung auch Individuen, die weit von ihren Eltern entfernt sind, und wirkt so auch explorierend.

Mutation

Die Mutation, die auf ein einzelnes Individuum angewendet wird, wird als *asexueller* genetischer Operator bezeichnet. Ein neuer Punkt im Suchraum wird durch die zufällige Modifikation eines einzelnen Individuums generiert. Die Mutation führt damit in der Regel zu einer Erhöhung der Varianz in der Population und ist die *explorierende* Suchkomponente eines EA.

Die Art der Modifikation und die Auswirkungen auf den Phänotyp eines Individuums hängen sehr stark von dem jeweiligen Operator und der zugrundeliegenden Kodierung ab. Damit der EA nicht zu einer puren Zufallssuche verkommt, sollten kleine Änderungen des Phänotyps häufiger vorkommen als große.

Selektion

Die Selektion ist die treibende Kraft eines EA, die der Suche eine Richtung vorgibt. Die Art, in der Individuen aufgrund ihrer Fitness als Eltern der nächsten Generation ausgewählt werden, hat entscheidenden Einfluß auf das Konvergenzverhalten eines Evolutionären Algorithmus.

Selektionsverfahren lassen sich nach drei Kriterien klassifizieren. Das erste Kriterium betrifft die Art der Auswahl, die *deterministisch* oder *probabilistisch* erfolgen kann. Deterministische Selektionsverfahren wählen die Individuen nach einem festgelegten Schema aufgrund ihrer Fitness aus. Probabilistische Verfahren bringen dagegen eine Zufallskomponente in die Selektion ein, um gleichzeitig in den vielversprechenden Bereichen des Suchraums alsauch im restlichen Suchraum zu suchen. Die probabilistischen Verfahren begegnen mit der Zufallskomponente dem Trade-off zwischen Zielgerichtetheit und Robustheit des Selektionsprozesses.

Das zweite Unterscheidungskriterium betrifft die Chance eines beliebigen Individuums, unabhängig von seiner Fitness selektiert zu werden. Selektionsverfahren, die bestimmte Individuen von der Fortpflanzung ausschließen, werden als *ausschließende* (engl.: *extinctive*), Selektionsverfahren, die allen Individuen die Chance zur Fortpflanzung geben, als potentiell *erhaltende* (engl.: *preservative*) bezeichnet.

Ein weiteres Kriterium berücksichtigt den Verlust der derzeit besten gefundenen Lösung. Sogenannte *elitäre* Selektionsstrategien halten die beste bislang gefundene Lösung künstlich in der Population, indem das beste Elternindividuum der letzten Generation selektiert wird, falls es besser als alle Nachkommen ist. Eine Selektion, die ausschließlich aus der Nachkommenpopulation selektiert, wird als *nicht-elitär* bezeichnet.

Paarbildung

Die Paarbildung, die aus der Menge der selektierten Individuen die Paare bzw. Tupel von Eltern zusammenstellt, aus denen ein oder mehrere Nachkommen erzeugt werden, wird in der Literatur oft beiläufig erwähnt, ohne ihr größere Bedeutung zuzumessen. Da sie aufgrund ihres Auswahlprozesses einen impliziten Selektionsvorgang darstellt, dessen Auswirkungen in Abhängigkeit von der verwendeten Selektion nicht unerheblich sind, wird im folgenden auf die verschiedenen Möglichkeiten der Paarbildung hingewiesen.

Paarbildungsverfahren lassen sich nach der Art, mit der Individuen aus der Menge der selektierten Individuen gezogen werden, unterscheiden. So kann das Ziehen mit Zurücklegen, mit Zurücklegen innerhalb eines Tupels oder ohne Zurücklegen erfolgen. Eine detailliertere Beschreibung der möglichen Paarbildungsverfahren und deren Effekte findet sich in Abschnitt 3.2.3.

3.1.4 Syntax zur Spezifikation von Evolutionären Algorithmen

Die im folgenden eingeführte Syntax dient zur eindeutigen Beschreibung eines EA. Aufgrund der vielfältigen Möglichkeiten in der Wahl der Operatoren und deren Parameter ist

eine solche Syntax erforderlich, um eine spezielle Ausprägung eines EA exakt zu beschreiben. Eine solche Sprache ist besonders hilfreich, um Erweiterungen oder Mischformen der hier beschriebenen Evolutionären Algorithmen zu entwickeln und zu spezifizieren. Sie soll insbesondere ein Werkzeug darstellen, um einen beliebigen EA kurz und prägnant zu spezifizieren.

Die Syntax spiegelt die Hierarchie, die Evolutionäre Algorithmen aufweisen, wider. Jede Hierarchiestufe wird durch ein Tupel spezifiziert, das die Objekte der nächstniederen Hierarchiestufe und die Operatoren, die auf diese Objekte angewendet werden, enthält.

Die oberste Hierarchiestufe wird im folgenden als **EA** bezeichnet. Ein **EA** besteht aus κ Populationen und dem Terminierungsverfahren ($\mathbb{T}$). Optional stehen als Operatoren ein Kommunikationsverfahren ($\mathbb{K}$) für Migrationsmodelle und ein Wettbewerbsverfahren ($\mathbb{W}$) für das in dieser Arbeit vorgestellte Wettbewerbskonzept zur Verfügung.

Die zweite Hierachiestufe, die Population (pop), besteht aus λ Individuen (ind) und den populationsbezogenen Operatoren Paarbildung ($\mathbb{P}$) und Selektion ($\mathbb{S}$). Die Initialisierung ($\mathbb{I}$) wird ebenfalls als Teil der Population angesehen. Optional kann ein lokales Optimierungsverfahren ($\mathbb{L}$) eingesetzt werden.

Ein Individuum (ind) wird durch seine Kodierung ($\mathbb{C}$) und die genetischen Operatoren ($\mathbb{R}$, $\mathbb{M}$) spezifiziert. Da spezielle EA-Varianten auf einzelne genetische Operatoren verzichten, werden beide als optional gekennzeichnet. Mit dieser Syntax lassen sich auch Populationen, die aus Individuen unterschiedlichen Typs bestehen, spezifizieren. In diesem Fall sind Kodierung und Rekombination entsprechend zu wählen.

Definition 6 (Syntax zur Spezifikation eines EA)

$$\langle \mathsf{ea} \rangle \quad ::= \quad \mathbf{EA}\,(\{\langle \mathsf{pop} \rangle, \ldots, \langle \mathsf{pop} \rangle\}, \mathbb{T}, [\mathbb{K}, \mathbb{W}]) \tag{3.10}$$

$$\langle \mathsf{pop} \rangle \quad ::= \quad \mathbf{POP}\,(\{\langle \mathsf{ind} \rangle, \ldots, \langle \mathsf{ind} \rangle\}, \mathbb{S}, \mathbb{P}, \mathbb{I}, [\mathbb{L}]) \tag{3.11}$$

$$\langle \mathsf{ind} \rangle \quad ::= \quad \mathbf{IND}\,(\mathbb{C}, [\mathbb{R}, \mathbb{M}]) \tag{3.12}$$

Falls die Populationen eines **EA** vom selben Typ sind, kann $\{\langle \mathsf{pop} \rangle, \ldots, \langle \mathsf{pop} \rangle\}$ durch die kürzere Schreibweise $\langle \mathsf{pop} \rangle^{\kappa}$ ersetzt werden, wobei κ die Anzahl der Populationen bezeichnet. Ebenso kann eine Menge gleichartiger Individuen $\{\langle \mathsf{ind} \rangle, \ldots, \langle \mathsf{ind} \rangle\}$ durch $\langle \mathsf{ind} \rangle^{\lambda}$ spezifiziert werden.

Diese Art der Spezifikation ermöglicht es, in einem Objekt unterschiedliche Objekte der nächstniedrigeren Hierarchiestufe aufzunehmen. So kann eine Population beispielsweise aus unterschiedlichen Individuen oder ein EA aus unterschiedlichen Populationen bestehen. Zur besseren Strukturierung können wie in objektorientierten Sprachen auch Objekte von Objekten derselben Hierarchieebene abgeleitet werden. In diesem Fall ersetzen die Eingabeparameter des neuen Objekts die gleichnamigen Parameter des Elternobjekts. Ein parametrisiertes Objekt kann als Klasse verstanden werden, während ein Objekt, das keine Parameter enthält, eine Instanz der Klasse ist, von der es abgeleitet wurde.

Im folgenden wird ein Beispiel für die hierarchische Spezifikation von EA-Populationen gegeben. Die verwendeten Operatoren sind in den folgenden Abschnitten definiert. Eine Übersicht aller in dieser Arbeit beschriebenen Operatoren ist in Anhang B zu finden.

Beispiel zur Spezifikation von EA-Objekten

Zunächst wird eine allgemeine EA-Population $\mathsf{POP}^{\mathsf{EA}}$ spezifiziert, die die Populationsgröße λ, die Remontierungsrate ϑ und zwei weitere Parameter des Rekombinationsoperators als Parameter hat. Aus dieser allgemeinen EA-Population werden zwei speziellere EA-Populationen, die sich in den Rekombinationsparametern unterscheiden, abgeleitet. Die beiden Populationen $\mathsf{POP}^{\mathsf{EA1}}$ und $\mathsf{POP}^{\mathsf{EA2}}$ erhalten neben den Eigenschaften der Elternklasse zusätzliche Parameter für Selektion und Rekombination.

$$\mathsf{POP}^{\mathsf{EA}}\ (\lambda, \vartheta, \Lambda, ,) ::= \mathbf{POP}\left(\mathbf{IND}\left(\mathbb{C}^{\mathrm{EA}}, \mathbb{R}^{\mathrm{EA}}_{\Lambda,\Gamma,2,\mathrm{R},1.0}, \mathbb{M}^{\mathrm{EA}}_{\mathrm{N},1.0^n,1}\right)^{\lambda}, \mathbb{S}^{\mathrm{EA,TR}}_{\vartheta,p}, \mathbb{P}^{\mathrm{EA,S}}, \mathbb{I}^{\mathrm{EA,U}}\right)$$

$$\mathsf{POP}^{\mathsf{EA1}}(\lambda) ::= \mathsf{POP}^{\mathsf{EA}}(\lambda, 0.25, \mathsf{V}, \mathsf{O})$$

$$\mathsf{POP}^{\mathsf{EA2}}(\lambda) ::= \mathsf{POP}^{\mathsf{EA}}(\lambda, 0.125, \mathsf{L}, \mathsf{U})$$

Aus den Populationen werden abschließend die EA-Instanzen ea1, ea2 und ea3 gebildet. Die ersten beiden Instanzen bestehen aus jeweils einer Population. Sie stammen von unterschiedlichen EA-Populationen ab und unterscheiden sich außerdem in der Terminierungsbedingung. Die dritte Instanz enthält zwei unterschiedliche Populationen, die mit Hilfe des Kommunikationsverfahrens $\mathbb{K}^{\mathrm{B2A}}$ Individuen austauschen.

$$\mathsf{ea1} ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{EA1}}(64), \mathbb{T}^{\mathrm{EA,FIT}}_{10^{-6}}\right)$$

$$\mathsf{ea2} ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{EA2}}(64), \mathbb{T}^{\mathrm{EA,IMP}}_{100,10^{-15}}\right)$$

$$\mathsf{ea3} ::= \mathbf{EA}\left(\left\{\mathsf{POP}^{\mathsf{EA1}}(32), \mathsf{POP}^{\mathsf{EA2}}(32)\right\}, \mathbb{T}^{\mathrm{EA,IMP}}_{100,10^{-15}}, \mathbb{K}^{\mathrm{B2A}}_{10}\right)$$

3.2 Allgemeine Operatoren

Die folgenden Operatoren werden von verschiedenen EA verwendet und sind daher keinem speziellen Verfahren zuzuordnen. Zu diesen *allgemeinen* Operatoren zählen in erster Linie Kodierung, Paarbildung, Selektion, Initialisierung und Terminierung. Außerdem werden allgemeine genetische Operatoren für Gleitkommazahlen beschrieben.

3.2.1 Kodierung

Die einfachste Kodierung eines EA für die kontinuierliche Parameteroptimierung ist die identische Abbildung. In diesem Fall besteht ein Individuum aus n Gleitkommazahlen, wobei n die Anzahl der zu optimierenden Parameter ist.

$$\mathbb{C}^{\mathrm{EA}} = (I^{\mathrm{EA}}, \mathbb{E}^{\mathrm{EA}}, \mathbb{D}^{\mathrm{EA}}) \tag{3.13}$$

Da die Kodierungs- und Dekodierungsfunktion jeweils die identische Abbildung ist, wird zwischen Genotyp und Phänotyp nicht unterschieden.

$$\mathbb{E}^{\mathrm{EA}}(\vec{x}) = \vec{x} \quad mit \quad \vec{x} \in D, I \tag{3.14}$$
$$\mathbb{D}^{\mathrm{EA}}(\vec{a}) = \vec{a} \quad mit \quad \vec{a} \in I, D \tag{3.15}$$

Die Fitnessfunktion $\mathbb{F}^{\mathrm{EA}}$ wertet die Objektvariablen eines Individuums aus und läßt sich daher direkt auf die Zielfunktion $f(\vec{x})$ abbilden.

$$\mathbb{F}^{\mathrm{EA}}(\vec{a}) = f(\mathbb{D}^{\mathrm{EA}}(\vec{a})) = f(\vec{a}) \tag{3.16}$$

3.2.2 Selektion

Schnittselektion $\mathbb{S}^{\mathrm{EA,TR}}_{\vartheta,\eta}$

Dieses Selektionsverfahren entspricht der sogenannten *Schnittselektion*, das Züchter für die Zucht in großen Populationen verwenden ([Fal81]). Hierbei werden alle Individuen streng nach dem Wert des Selektionsmerkmals rangiert. Diese Rangfolge wird an einem Punkt abgeschnitten, so daß ausschließlich Individuen oberhalb dieses Schnitts zur Zucht verwendet werden. Die Proportion der selektierten Individuen wird *Remontierungsrate* genannt und im folgenden mit ϑ bezeichnet. Die selektierte Population in Generation t berechnet sich aus:

$$P^{\mathrm{S}}(t) = \{\vec{a}_i \in P \mid 1 \leq i \leq \mu \ \wedge \ \forall \vec{a}_j \in P \backslash P^{\mathrm{S}}(t) \quad : \quad \mathbb{F}(\vec{a}_i) \leq \mathbb{F}(\vec{a}_j)\} \tag{3.17}$$

mit

$$\mu = \lfloor \lambda \cdot \vartheta \rfloor \tag{3.18}$$

und

$$P = \begin{cases} P(t \Leftrightarrow 1) \cup P^{\mathrm{S}}(t \Leftrightarrow 1) & : \quad \eta = \mathsf{p(lus)} \\ P(t \Leftrightarrow 1) & : \quad \eta = \mathsf{c(omma)} \\ P(t \Leftrightarrow 1) \cup \vec{a}^{*} & : \quad \eta = \mathsf{e(litist)} \end{cases} \tag{3.19}$$

wobei $\vec{a}^{*}$ das beste Individuum der selektierten Individuen der vorhergehenden Generation ist:

$$\vec{a}^{*} = \vec{a}_i \in P^{\mathrm{S}}(t \Leftrightarrow 1) : (\forall \vec{a}_{j,j \neq i} \in P^{\mathrm{S}}(t \Leftrightarrow 1) : \mathbb{F}(\vec{a}_i) \leq \mathbb{F}(\vec{a}_j)) \tag{3.20}$$

Turnierselektion $\mathbb{S}^{\mathrm{EA,TO}}_{q}$

Diese ebenfalls in EA häufig eingesetzte Selektionsart wählt μ-mal das jeweils beste Individuum aus einer zufällig bestimmten Untermenge B_k der Größe q von P aus. Die Menge P, aus der die Individuen selektiert werden, bleibt während der Selektion unverändert. Die Auswahl entspricht damit einem Ziehen mit Zurücklegen.

Proportionale Selektion $\mathbb{S}^{\mathrm{EA,PR}}$

Die von Holland entwickelte *proportionale Selektion* zählt zu den probabilistischen erhaltenden Selektionsverfahren. Die Selektion der Individuen erfolgt bei der ursprünglichen Form der proportionalen Selektion implizit durch die Zusammenstellung der Paare. Jedes Individuum wird mit einer Wahrscheinlichkeit, die von dessen relativer Fitness innerhalb der Population abhängt, als Elter bei der Paarbildung berücksichtigt. Eine Trennung zwischen Selektion und Paarbildung kann erreicht werden, indem in der Selektion zunächst $\lambda \cdot \rho$ Individuen aufgrund ihrer Fitnesswerte ausgewählt und anschließend in der Paarbildung zu Tupeln kombiniert werden.

Die ursprüngliche Form der proportionalen Selektion berechnet die sogenannte *Selektionswahrscheinlichkeit* $p_{\mathbb{S}}$ eines Individuums $\vec{a}_i$ folgendermaßen:

$$p_{\mathbb{S}}(\vec{a}_i) = \frac{\mathbb{F}(\vec{a}_i)}{\sum_{j=1}^{\lambda} \mathbb{F}(\vec{a}_j)} \tag{3.21}$$

Da diese Art der proportionalen Selektion nur für positive Fitnesswerte anwendbar ist, existieren zahlreiche Modifikationen, von denen zwei in Abschnitt 3.4.2 vorgestellt werden.

3.2.3 Paarbildung

Die Paarbildung stellt aus der Menge der selektierten Individuen λ Tupel von Individuen zusammen. In den verschiedenen Evolutionären Algorithmen erfolgt die Zusammenstellung der Tupel in der Regel rein zufällig. Unterschiede zwischen verschiedenen Varianten der zufälligen Paarbildung bestehen im Ziehen der Individuen.

Paarbildung durch Ziehen mit Zurücklegen $\mathbb{P}_{\rho}^{\mathrm{EA,R}}$

Dieses Paarbildungsverfahren stellt die Elterntupel zusammen, indem es aus der Menge der selektierten Individuen $P^{\mathbb{S}}$ λ mal ρ Individuen mit Zurücklegen zieht (Urnenmodell mit Zurücklegen).

$$Q_i = \left\{ \vec{a}_{k_j} \right\}_{j=1}^{\rho} \qquad \text{für} \quad i = 1, \cdots, \lambda \tag{3.22}$$

mit $k_j \sim \mathbf{U}(\{1, \mu\})$ und $\vec{a}_{k_j} \in P^{\mathbb{S}}$.

Q_i bezeichnet das i-te Elterntupel. Jedes Tupel besteht aus ρ Individuen, die zufällig aus der Menge der selektierten Eltern $P^{\mathbb{S}}$ gezogen werden. Die Paarung identischer Individuen (engl.: *self-mating*) ist bei diesem Paarbildungsverfahren nicht explizit ausgeschlossen.

Paarbildung durch Ziehen ohne Zurücklegen innerhalb eines Tupels $\mathbb{P}_{\rho}^{\mathrm{EA,R^*}}$

Die Paarung identischer Individuen läßt sich vermeiden, indem die gezogenen Individuen erst dann zurückgelegt werden, wenn ein Elterntupel vollständig ist.

$$Q_i = \left\{ \vec{a}_{k_j} \right\}_{j=1}^{\rho} \qquad \text{für} \quad i = 1, \cdots, \lambda \tag{3.23}$$

mit $k_j \sim \mathbf{U}(\{1, \mu\})$, $\vec{a}_{k_j} \in P^{\mathbb{S}}$ und $\forall\, l < j k_j \neq k_l$.

Bei diesem Paarbildungsverfahren muß gewährleistet sein, daß die Größe eines Tupels nicht größer ist als die Anzahl der selektierten Individuen ($\rho \leq \mu$).

Paarbildung durch Ziehen ohne Zurücklegen $\mathbb{P}^{\mathrm{EA,S}}_{\rho}$

Das zufällige Zusammenstellen der Elterntupel enthält einen impliziten Auswahlprozeß, der dazu führt, daß ein Anteil der selektierten Individuen in keinem der Eltern-Tupel vorkommt. Dieser Effekt kann vermieden werden, indem aus der Menge der selektierten Individuen P^{s} eine entsprechend vergrößerte Menge $\ddot{P}^{\mathrm{s}}$ mit derselben Proportion gebildet wird, aus der die Paare durch Ziehen *ohne* Zurücklegen zusammengestellt werden.

$$Q_i = \left\{ \vec{a}_{k_j} \right\}_{j=1}^{\rho} \qquad \text{für} \quad i = 1, \ldots, \lambda \tag{3.24}$$

mit $k_j = i \cdot \rho + j$ und $\vec{a}_{k_j} \in \ddot{P}^{\mathrm{s}}$.

$\ddot{P}^{\mathrm{s}}$ ist eine zufällig angeordnete Liste, die aus insgesamt $\rho \cdot \lambda$ Individuen besteht. Falls $\rho \cdot \lambda$ ein ganzzahliges Vielfaches von μ ist, ist in dieser Liste jedes der μ selektierten Individuen genau $\rho \cdot \lambda / \mu$ mal vertreten. Andernfalls werden von jedem Individuum $\lfloor \rho \cdot \lambda / \mu \rfloor$ oder $\lceil \rho \cdot \lambda / \mu \rceil$ Kopien aufgenommen, so daß die Gesamtzahl von $\rho \cdot \lambda$ erreicht wird. Auf diese Weise erhält $\ddot{P}^{\mathrm{s}}$ annähernd die gleiche Proportion wie die Menge der selektierten Individuen.

Das Verhalten der verschiedenen Paarbildungsverfahren ist in Abbildung 3.2 festgehalten. Das Experiment in Abbildung (a) arbeitet mit der $\mathbb{P}^{\mathrm{EA,R}}$-Paarung, während der Algorithmus in Abbildung (b) die $\mathbb{P}^{\mathrm{EA,S}}$-Paarung verwendet.

In einigen Implementierungen werden entgegen der Definition des allgemeinen EAs (Definition 1) aus Effizienzgründen aus einem Elternpaar *zwei* Nachkommen erzeugt (GENESIS 5.0, [Gre90]). Der Rekombinationsoperator, der nun einer Abbildung $\mathbb{R} : I^2 \to I^2$ entspricht, erzeugt *komplementäre* Nachkommen. In Abbildung 3.2 ist der Einfluß dieses Aspektes anhand dreier GA-Varianten dargestellt, die sich ausschließlich in der Paarbildung unterscheiden. Die GA in den Abbildungen (b) und (c) verwenden die $\mathbb{P}^{\mathrm{EA,S}}$-Paarung. Sie unterscheiden sich lediglich in der Anzahl der Nachkommen eines Paares.

Der durchschnittliche Verlauf des besten Fitnesswertes von 100 Experimenten zeigt, daß die Erzeugung zweier komplementärer Nachkommen einer frühzeitigen Stagnation entgegenwirkt (Abbildung 3.2 c)).

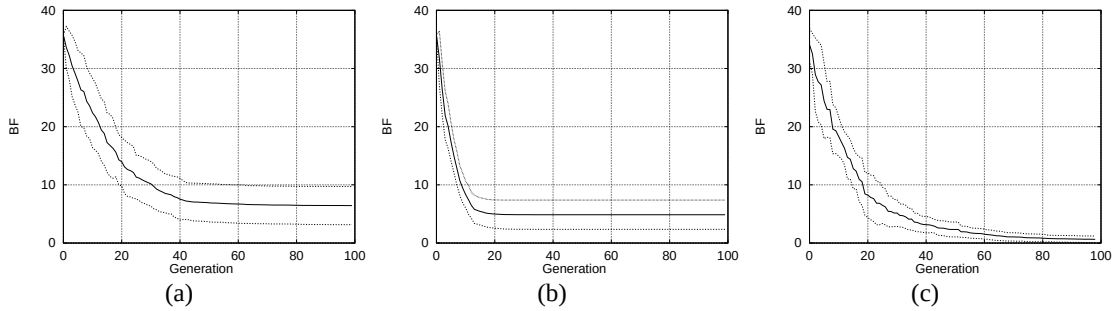

Abbildung 3.2: Vergleich zwischen den beiden Paarbildungsverfahren $\mathbb{P}^{\mathrm{EA,R}}$ (a) und $\mathbb{P}^{\mathrm{EA,S}}$ (b) und einer Modifikation von $\mathbb{P}^{\mathrm{EA,S}}$, bei der aus zwei Eltern zwei komplementäre Nachkommen erzeugt werden (c), anhand des Kugelmodells ($n = 10$). Dargestellt ist die Entwicklung des besten Fitnesswertes (BF) über die Generationen. Der den Abbildungen zugrundeliegende Algorithmus entspricht einem typischen GA ohne Mutation: $\mathbf{EA}\left(\mathbf{POP}\left(\mathbf{IND}(\mathbb{C}^{\mathrm{GA}}_{\mathrm{G},10}, \mathbb{R}^{\mathrm{GA,2P}}_{0.6})^{50}, \mathbb{S}^{\mathrm{GA}}_{\mathrm{R,B},1.25}, \mathbb{P}^{*}, \mathbb{I}^{\mathrm{EA,U}}\right)\mathbb{T}^{\mathrm{IMP}}_{\tau,\epsilon}\right)$. In den Abbildungen sind jeweils die Mittelwerte und die Standardabweichungen der besten Fitnesswerte von 100 Läufen angegeben.

3.2.4 Genetische Operatoren

Rekombinationsoperator für reelle Kodierungen $\mathbb{R}^{\mathrm{EA}}_{\Lambda,\Gamma,\Upsilon,\Omega,\delta,[k]}$

Die Möglichkeiten, aus einer Menge reellkodierter Individuen ein neues Individuum zu erzeugen, sind unerschöpflich. Neben der Anzahl der beteiligten Eltern bietet die Wahl der zugrundeliegenden Wahrscheinlichkeitsverteilung beliebige Variationsmöglichkeiten, die von einer richtungsorientierten bis zur volumenorientierten Suche reichen. Die Varianzentwicklung einer Population wird ebenfalls durch die konkrete Wahrscheinlichkeitsverteilung beeinflußt.

In [SRB95] wurde erstmals ein allgemeiner Rekombinationsoperator formal eingeführt, aus dem durch geeignete Parametrisierung alle in Evolutionsstrategien eingesetzten Rekombinationsvarianten modelliert werden können. Der hier eingeführte *Rekombinationsoperator für reelle Kodierungen* stellt eine Verallgemeinerung dieses Rekombinationsoperators dar, da hier verschiedene Eigenschaften eines Rekombinationsoperators miteinander zu einem speziellen Rekombinationsoperator kombiniert werden können. Durch geeignete Parametrisierung kann so ein Großteil der in der EA-Welt verwendeten Rekombinationsoperatoren modelliert werden.

Da das sexuelle Rekombinationsschema das verbreitetste ist und die Variationsmöglichkeiten für den multisexuellen Fall eine strukturierte Darstellung erschweren, wird dieser allgemeine Rekombinationsoperator zunächst nur für den sexuellen Fall eingeführt. Folgende Gleichung stellt die Grundstruktur des allgemeinen Rekombinationsoperators dar:

$$\boxed{x'_i = \bar{x}_i + \Delta_i \cdot \left(\alpha^{\Gamma} \cdot \frac{\delta}{\omega_\Omega} + \beta_i^{\Upsilon}\right)} \quad \forall i = 1, \ldots, n \tag{3.25}$$

mit $\bar{x}_i = \frac{x_i^{\mathrm{P}1} + x_i^{\mathrm{P}2}}{2}$ und $\Delta_i = x_i^{\mathrm{P}2} \Leftrightarrow x_i^{\mathrm{P}1}$.

Der neue Punkt $\vec{x}'$ wird durch die beiden Positionen der beiden Eltern $\vec{x}^{\mathrm{P}1}$ und $\vec{x}^{\mathrm{P}2}$ bestimmt. Die Art der Verteilung setzt sich aus Zufallsvariablen α und β sowie zwei weiteren Faktoren zusammen. Die auf den Positionen der Eltern basierte Wahrscheinlichkeitsverteilung ist der Kern der populationsbasierten Suche und wird im folgenden als Merkmal für die Strukturierung reellbasierter Rekombinationsoperatoren verwendet.

Strukturierung des Rekombinationsoperators. Diese allgemeine Form des Rekombinationsoperators kann bezüglich folgender Merkmale klassifiziert werden:

- **Suchprinzip:** volumen- oder richtungsorientiert (Λ)
- **Verteilung:** der Schritte pro Dimension ($, , \Upsilon$)
- **Berechnungsart:** relativ zu den Eltern oder absolut (Ω)
- **Schrittweitenfaktor:** pro Dimension (δ)

Suchprinzip: Das Suchprinzip des Rekombinationsoperators kann *volumen-* oder *richtungsorientiert* sein. Eine volumenorientierte Suche bedeutet, daß der Rekombinationsoperator einen Punkt in einem Bereich des Suchraums erzeugt, der durch die Eltern definiert

ist. Dies wird erreicht, indem die Zufallsvariable α aus Gleichung 3.25 für jede Komponente des Parametervektors neu bestimmt wird. Wird derselbe Wert der Zufallsvariablen α für alle Komponenten verwendet, liegt der resultierende Punkt auf der durch die Eltern definierten Geraden. In diesem Fall spricht man von einer richtungsorientierten Suche.

Das Suchprinzip wird durch den Parameter $\Lambda \in \{V, L\}$ spezifiziert: V steht für volumenorientiert und L für richtungsorientiert.

$$\alpha_i = \begin{cases} \tilde{\alpha}_i \sim , & : \quad \Lambda = V \\ \tilde{\alpha} \sim , & : \quad \Lambda = L \end{cases} \qquad \forall i = 1, \ldots, n \tag{3.26}$$

Verteilung: Die in der Rekombination verwendete Verteilung pro Dimension setzt sich aus den beiden Zufallsvariablen α^{Γ} und β^{Υ} zusammen. Die externen Parameter des Rekombinationsoperators , und Υ spezifizieren Art und Modalität der Verteilung.

In Rekombinationsoperatoren häufig verwendete Verteilungen sind die Einpunkt- (O), Gleich- (U), Normal- (N) und die Dreiecksverteilung (T). Andere Verteilungen wie Exponential-, Cauchy- oder Hyperbelverteilung sind ebenfalls möglich. Eine Übersicht zu den verschiedenen Verteilungen findet sich in Abschnitt 4.3 auf Seite 68.

$$\alpha^{\Gamma} \sim \begin{cases} \mathbf{B}(k,1) & : \ , = B & \text{BGA-Mutationsverteilung} \\ \mathbf{C}(1) & : \ , = C & \text{Cauchyverteilung} \\ \mathbf{E}(1) & : \ , = E & \text{Exponentialverteilung} \\ 0.0 & : \ , = O & \text{Einpunktverteilung} \\ \mathbf{N}(0,1) & : \ , = N & \text{Normalverteilung} \\ \mathbf{T}(1) & : \ , = T & \text{Dreiecksverteilung} \\ \mathbf{U}(\{0,1\}) & : \ , = U & \text{Gleichverteilung} \end{cases} \tag{3.27}$$

Die diskrete Zufallsvariable β^{Υ} definiert die Modalität der Verteilung sowie die Lage der Häufungspunkte. Die beiden unimodalen Alternativen unterscheiden sich in der Lage ihres Häufungspunktes, der in der Mitte der beiden Eltern ($\Upsilon = 1$) oder im besseren Elternindividuum ($\Upsilon = 3$) liegen kann. Die bimodale Alternative hat ihre Häufungspunkte an den Positionen der beiden Eltern ($\Upsilon = 2$). Die Erweiterung auf multimodale Verteilungen ist besonders für multisexuelle Rekombinationsschemata denkbar.

$$\beta^{\Upsilon} = \begin{cases} 0.0 & : \quad \Upsilon = 1 \\ \mathbf{U}(\{\Leftrightarrow 0.5, 0.5\} & : \quad \Upsilon = 2 \\ \begin{cases} \Leftrightarrow 0.5 & : \quad \mathbb{F}(\vec{a}^{\,\mathrm{P}\,1}) < \mathbb{F}(\vec{a}^{\,\mathrm{P}\,2}) \\ 0.5 & : \quad sonst \end{cases} & : \quad \Upsilon = 3 \end{cases} \tag{3.28}$$

Berechnungsart: Die Berechnungsart gibt an, ob die Schrittweite relativ zu dem Abstand der Eltern oder absolut berechnet werden soll.

$$\omega^{\Omega} = \begin{cases} 1 & : \quad \Omega = R & \text{relativ} \\ \|x_i^{\mathrm{P}\,1}, x_i^{\mathrm{P}\,2}\| & : \quad \Omega = A & \text{absolut} \end{cases} \tag{3.29}$$

Schrittweitenfaktor: Der Schrittweitenfaktor δ dehnt oder staucht die Verteilungsfunktion. Abbildung 3.3 zeigt ein zweidimensionales Schema der verschiedenen Varianten des Rekombinationsoperators. Für einige Varianten sind in Abbildung 3.4 die zweidimensionalen Dichtefunktionen dargestellt.

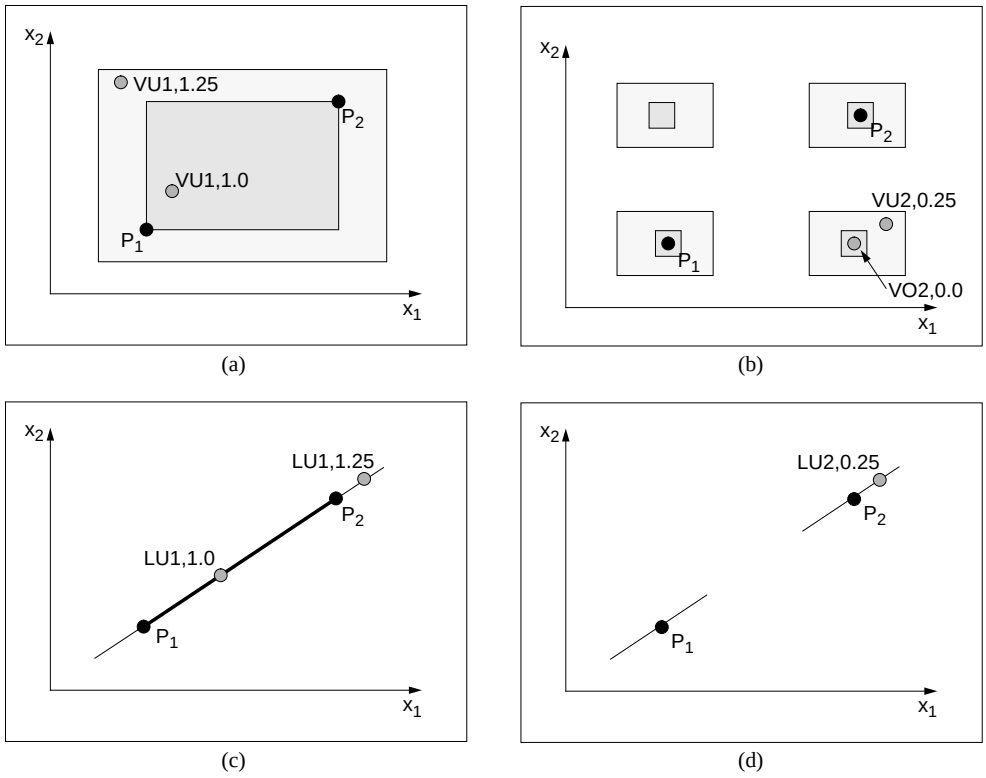

Abbildung 3.3: 2-dimensionales Schema der verschiedenen Rekombinationsmechanismen. Die beiden Elternindividuen sind durch P_1 und P_2, die Nachkommen durch die entsprechenden Parameter des Rekombinationsoperators $\mathbb{R}^{\mathrm{EA}}$ gekennzeichnet. Abbildungen (a) und (b) zeigen eine unimodale ($\Upsilon = 1$) und bimodale ($\Upsilon = 2$) volumenorientierte Suche ($, = V$). Die Abbildungen (c) und (d) geben die entsprechenden richtungsorientierten Varianten wieder ($, = L$).

Der hier vorgestellte allgemeine Rekombinationsoperator ist nur für den sexuellen Fall ($I^2 \to I$) definiert. Eine Erweiterung auf den multisexuellen Fall, in dem mehr als zwei Eltern an der Erzeugung eines neuen Nachkommens beteiligt sind, kann auf verschiedene Arten definiert werden. Die einfachste Möglichkeit besteht darin, jede Objektvariable mit Hilfe des allgemeinen Rekombinationsoperators aus zwei zufällig ausgewählten Elternindividuen zu generieren. In diesem Fall kann die Grundstruktur des allgemeinen Rekombinationsoperators weiter verwendet werden. Eine allgemeinere Möglichkeit, aus ρ Elternindividuen einen Nachkommen zu generieren, basiert auf einer Wahrscheinlichkeitsverteilung, die auf allen ρ Elternindividuen basiert.

Mutationsoperator für reelle Kodierungen $\mathbb{M}^{\mathbb{EA}}_{\Gamma,\vec{p},\nu,[k]}$

Der Entwurf eines auf reellen Objektvariablen operierenden Mutationsoperators, der Korrelationen zwischen Objektvariablen unberücksichtigt läßt, wirft folgende Entscheidungs-

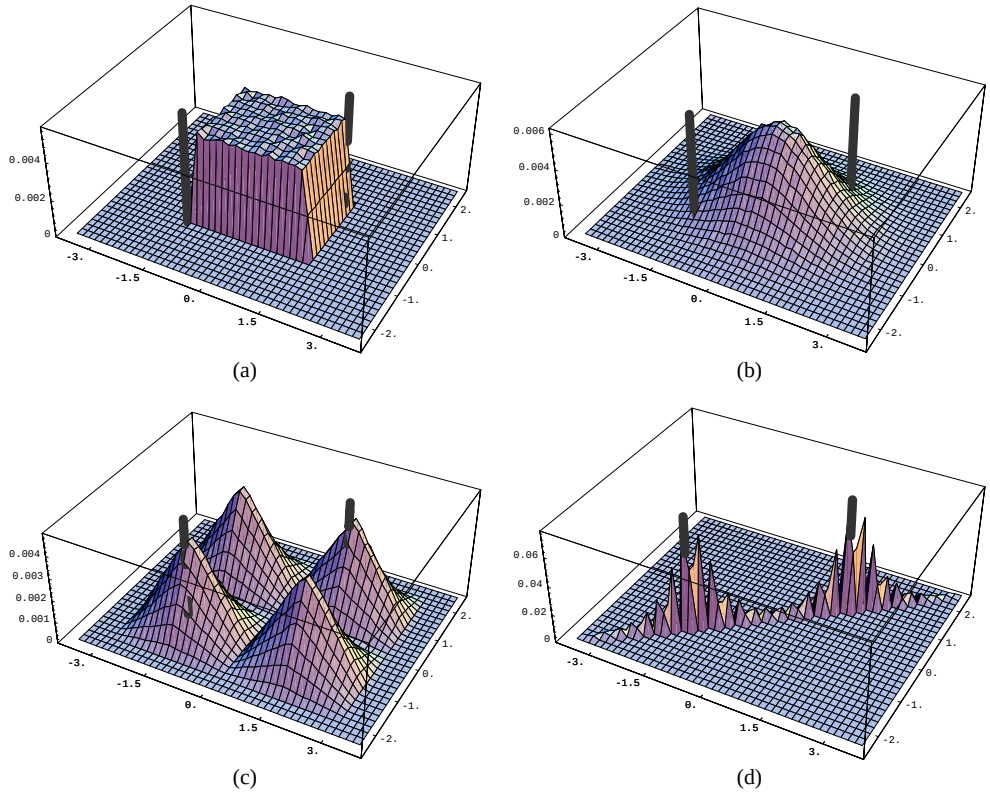

Abbildung 3.4: Zweidimensionale Dichtefunktionen für verschiedene Varianten des Rekombinationsoperators. Die Schnittpunkte der Geraden mit der Ebene kennzeichnen die Positionen der beiden Eltern ($(-1.5,-1)$ und $(1.5,1)$). (a): $\mathbb{R}^{\mathrm{EA}}_{\mathrm{VU1R},1.0}$, (b): $\mathbb{R}^{\mathrm{EA}}_{\mathrm{VN1R},0.5}$, (c): $\mathbb{R}^{\mathrm{EA}}_{\mathrm{VT2R},0.5}$, (c): $\mathbb{R}^{\mathrm{EA}}_{\mathrm{LE2R},0.15}$

fragen auf:

- Vorzeichen pro Objektvariable
- Mutationsverteilung pro Objektvariable
- Auswahl der zu mutierenden Objektvariablen

Da keine apriori-Information über eine zu bevorzugende Richtung vorliegt, wählt man jede Richtung mit gleicher Wahrscheinlichkeit. Die Wahl der Mutationsverteilung pro Objektvariable ist nicht ganz so einfach. Für den Raum der reellen Zahlen kommt jede stetige Verteilung in Frage. Beispiele sind Gleichverteilung, Normalverteilung, Cauchyverteilung und Exponentialverteilung. Die Auswahl der zu mutierenden Objektvariablen kann deterministisch oder probabilistisch erfolgen. Ein Mutationsoperator modifiziert den Wert einer reellen Objektvariablen auf folgende Weise:

$$\boxed{x_i' = x_i + \beta \cdot \rho_i \cdot \alpha^{\Gamma}} \tag{3.30}$$

wobei $\beta \sim \mathbf{U}(\{\Leftrightarrow 1, 1\})$ die Zufallsvariable für die Richtung und α^{Γ} die in Gleichung 3.27 definierte Zufallsvariable der Mutationsverteilung angibt. ρ_i ist ein Schrittweitenparameter, der für jede Objektvariable spezifiziert werden kann. Der dritte externe Parameter ν gibt die Anzahl der gleichzeitig zu mutierenden Objektparameter an. Für $\nu < n$ werden ν Objektparameter durch Ziehen ohne Zurücklegen zufällig bestimmt und mutiert.

Behandlung von Schranken

Schranken, die den Suchbereich einer Zielfunktion einschränken, sind Nebenbedingungen, die sich als Ungleichheitsrestriktionen formulieren lassen. Grundsätzlich können Evolutionäre Algorithmen zur Behandlung solcher Nebenbedingungen auf allgemeine Methoden der numerischen Optimierung zurückgreifen. Während GA aufgrund ihrer Binärkodierung implizit eine untere und obere Schranke für die Entscheidungsvariablen aufweisen, arbeiten Suchverfahren mit reellwertigen Variablen in der Regel ohne Schranken. Durch die Ausnutzung von problemspezifischem Wissen kann der Suchbereich jedoch häufig eingeschränkt und die Suche damit erleichtert werden. Die Art, mit der auf eine Verletzung von Schranken reagiert wird, kann einen wesentlichen Einfluß auf Erfolg und Effizienz der Suche haben. Für die Behandlung der Verletzung des Definitionsbereichs bieten sich folgende Methoden an. Der Definitionbereich des aus den genetischen Operatoren resultierenden i-ten Objektparameters x_i' ist durch $x_i^{\min}$ und $x_i^{\max}$ spezifiziert.

Methode 1 (Letalität): Die nächstliegende Methode besteht darin, das entsprechende Individuum als nicht lebensfähig (letal) zu betrachten und von der Selektion und damit von der Fortpflanzung auszuschließen.

Methode 2 (Beibehaltung des ursprünglichen Wertes): Eine weitere einfache Alternative besteht darin, einen ungültigen Wert abzulehnen und stattdessen den ursprünglichen Wert des Parameters beizubehalten.

$$x_i'' = \begin{cases} x_i' & : \quad x_i^{\min} \leq x_i' \leq x_i^{\max} \\ x_i & : \quad sonst \end{cases} \tag{3.31}$$

Da Funktionsauswertungen für ungültige Individuen entfallen, profitiert diese Methode besonders von Zielfunktionen, deren Optimum in der Nähe eines ungültigen Bereichs liegt.

Methode 3 (Reparatur): Falls die Nebenbedingungen durch den Definitionsbereich festgelegt sind, kann eine ungültige Objektvariable repariert werden. Die Reparatur besteht in der Zuweisung des ähnlichsten gültigen Wertes der Objektvariablen. Der ungültigen Objektvariablen wird der minimale bzw. maximale Wert des Definitionsbereichs zugewiesen, je nachdem, ob der Definitionsbereich unter- oder überschritten wurde.

$$x_i'' = \begin{cases} x_i' & : \quad x_i^{\min} \leq x_i' \leq x_i^{\max} \\ x_i^{\min} & : \quad x_i' < x_i^{\min} \\ x_i^{\max} & : \quad x_i' > x_i^{\max} \end{cases} \tag{3.32}$$

Methode 4 (Straffunktion): Eine Abschwächung der Letalitätsmethode (Methode 1) wird durch die Verwendung einer Straffunktion erreicht. Hierbei führt eine Verletzung der Schranke nicht zum Ausschluß des Individuums, sondern lediglich zur Verschlechterung seiner Fitness. Dazu wird die eigentliche Zielfunktion in eine Hilfszielfunktion transformiert, die neben der Zielfunktion auch Verletzungen der Nebenbedingungen berücksichtigt. Gegenüber der Letalitätsmethode erhält das Suchverfahren hier Informationen, um

den ungültigen Bereich gezielt zu verlassen. Die Modellierung der Straffunktion ist nicht trivial und hängt letztlich von dem zu lösenden Problem ab. Hierbei ist zwischen der Wichtigkeit der Nebenbedingungen und dem erreichtem Zielfunktionswert zu gewichten. Zudem kann problemspezifisches Wissen in die Modellierung der Straffunktion einfließen, so daß die Suche möglichst schnell in einen gültigen Bereich führt.

Die Wahl der richtigen Methode hängt sehr stark von der Art des Evolutionären Algorithmus ab. So sind beispielsweise die Methoden 2 und 3, in denen die resultierenden Werte nicht ausschließlich durch die genetischen Operatoren erzeugt werden, für eine Evolutionsstrategie mit Schrittweitensteuerung gänzlich ungeeignet, da die Beziehung zwischen gemachtem Schritt und zugehöriger Schrittweite verloren geht.

3.2.5 Initialisierung

Die Initialisierung spielt in der Optimierung eine nicht zu unterschätzende Rolle. Während iterative Optimierungsverfahren, die einen neuen Punkt nach dem anderen generieren, ihre Suche an einem einzigen Punkt beginnen, bietet sich den EA die Möglichkeit, die Individuen der Anfangspopulation in dem Suchraum zu verteilen. Ist der Definitionsbereich des Problems bekannt und keine weitere Information über die Funktion verfügbar, basiert die Initialisierung der Objektvariablen in der Regel auf einer n-dimensionalen Wahrscheinlichkeitsverteilung.

Neben den Objektvariablen arbeiten viele Optimierungsverfahren mit weiteren Hilfsvariablen, die während der Suche modifiziert werden. Die Initialisierung dieser Hilfsvariablen, die oft einen erheblichen Einfluß auf die Suche haben, wird in der Regel durch das Suchverfahren vorgegeben.

Gleichverteilte Initialisierung $\mathbb{I}^{\mathrm{EA,U}}_{\vec{x}^{\min},\vec{x}^{\max}}$

Diese Art der Initialisierung erzeugt die Individuen der ersten Generation gleichverteilt in dem durch die Vektoren $\vec{x}^{\min}$ und $\vec{x}^{\max}$ begrenzten Bereich. Die Grenzen sind meist durch die Problemstellung vorgegeben. Falls der Suchraum des Problems unbegrenzt ist, müssen für die Initialisierung plausible Grenzen festgelegt werden.

$$x_i = \mathbf{U}([x_i^{\min}, x_i^{\max}]) \qquad \forall i = 1, \ldots, n \tag{3.33}$$

Punktuelle Initialisierung $\mathbb{I}^{\mathrm{EA,P}}_{\vec{x}^0,\sigma}$

Die punktuelle Initialisierung entspricht im wesentlichen der Initialisierung traditioneller Optimierungsverfahren, die nur einen Startpunkt vorgeben. Als Startpunkt wird hier der beste bekannte Punkt verwendet. Um zu vermeiden, daß alle Individuen identisch sind, werden die initialen Werte aller Individuen, mit Ausnahme des ersten, um den vorgegebenen Startpunkt normalverteilt gestreut. Das erste Individuum erhält den vorgegebenen Startpunkt $\vec{x}^0$.

$$x_i = \begin{cases} x_i^0 & : \quad \vec{a}_i = \vec{a}_0 \\ \mathbf{N}(x_i^0, \sigma) & : \quad sonst \end{cases} \qquad \forall i = 1, \ldots, n \tag{3.34}$$

3.2.6 Terminierung

Auf den ersten Blick erscheint die Frage nach der Terminierungsbedingung als nebensächlich und wird in der Beschreibung von Suchverfahren häufig vernachlässigt. Dennoch ist die Entscheidung, wann die Suche abzubrechen ist, für die Optimierung realer Problemstellungen von eminenter Bedeutung.

Da das Optimum realer Probleme nicht bekannt ist und Suchverfahren nicht in jeder Generation einen Fortschritt erzielen, kann nicht mit Sicherheit gesagt werden, ob die Fortsetzung der Suche eine weitere Verbesserung bringt oder nicht.

Aus diesem Grund werden in Evolutionären Algorithmen verschiedene pragmatische Terminierungsbedingungen verwendet. Welche Terminierungsbedingung im konkreten Fall eingesetzt wird, hängt letztlich von der jeweiligen Problemstellung ab. Bei der Wahl der Terminierungsbedingung muß die Qualität der Lösung mit dem dafür erforderlichen Aufwand gewichtet werden.

In der Praxis ist es oft angebracht, verschiedene Terminierungsbedingungen zu kombinieren.

Terminierung $\mathbb{T}_{\epsilon}^{\mathrm{EA,FIT}}$

Ist die Qualität des Optimierungsproblems f^* vorgegeben, wird die Suche fortgesetzt, bis das beste Individuum $\vec{a}^*(t)$ der Population diese Qualität zum ersten Mal erreicht.

$$\mathbb{T}(P(t)) = \begin{cases} \mathsf{true} & : \; = \mathbb{F}(\vec{a}^*(t)) \Leftrightarrow f^* < \epsilon \\ \mathsf{false} & : \; sonst \end{cases} \tag{3.35}$$

Terminierung $\mathbb{T}_{\tau}^{\mathrm{EA,GEN}}$

Ein einfaches Terminierungskriterium ist die Vorgabe der Generationen. In diesem Fall bricht die Suche nach τ Generationen ab. Diese Terminierungsbedingung kann abgewandelt werden, indem man die Anzahl der Funktionsauswertungen oder die Rechenzeit als Terminierungskriterium zugrunde legt.

$$\mathbb{T}(P(t)) = \begin{cases} \mathsf{true} & : \; t = \tau \\ \mathsf{false} & : \; t < \tau \end{cases} \tag{3.36}$$

Terminierung $\mathbb{T}_{\tau,\epsilon}^{\mathrm{EA,IMP}}$

Diese Terminierungsbedingung tritt ein, falls in den letzten τ Generationen keine Verbesserung des Funktionswertes um mindestens ϵ eingetreten ist. Ziel dieses Kriteriums ist es, auf eine andauernde Stagnation des Suchprozesses zu reagieren. Eine kurzfristige Stagnation führt bei dieser Bedingung nicht zu einer Terminierung der Suche.

$$\mathbb{T}(P(t)) = \begin{cases} \mathsf{true} & : \; \mathbb{F}(\vec{a}_i^*(t \Leftrightarrow \tau)) \Leftrightarrow \mathbb{F}(\vec{a}_i^*(t)) < \epsilon \\ \mathsf{false} & : \; sonst \end{cases} \tag{3.37}$$

3.3 Evolutionsstrategien

Die Evolutionsstrategien (ES) wurden in den 60er Jahren gemeinsam von Rechenberg und Schwefel an der Technischen Universität Berlin (TUB) entwickelt ([Rec65], [Sch65], [Sch68]). Die ersten Anwendungen waren hydrodynamische Probleme aus dem Bereich der experimentellen Optimierung wie die Strukturoptimierung einer Zweiphasendüse und die Formoptimierung eines gebogenen Rohres.

Die erste Version einer Evolutionsstrategie ist Rechenbergs sogenannte (1+1)-ES. Der Name kennzeichnet die Art der Selektion und die Anzahl der Individuen. Die (1+1)-ES ist ein aus einem Individuum bestehendes Mutations-Selektions-Verfahren, dessen mittlere Mutationsschrittweite durch die bekannte *1/5-Regel* eingestellt wird (siehe [Rec73]). Mit der Erweiterung dieses Mutations-Selektions-Verfahrens zu der sogenannten $(\mu + 1)$-ES führte Rechenberg das Populationsprinzip und die Rekombination in die Evolutionsstrategie ein. Aus den μ Eltern wird ein Nachkomme erzeugt, der gegebenenfalls den schlechtesten Elter ersetzt. Durch die Hinzunahme weiterer Nachkommen erweiterte Schwefel diese erste populationsbasierte Evolutionsstrategie zu der bekannten $(\mu+\lambda)$-Evolutionsstrategie bzw. (μ, λ)-Evolutionsstrategie ([Sch75]).

Heute hat sich die (μ, λ)-Evolutionsstrategie mit Selbstanpassung ihrer Strategieparameter als Standard durchgesetzt ([Sch95], [Rec94]).

3.3.1 Kodierung und Fitnessfunktion

Die Individuen einer Evolutionsstrategie beinhalten neben einer Repräsentation der Objektparameter sogenannte *Strategieparameter*, die zur Steuerung der Schrittweiten und Orientierung des Mutationsoperators verwendet werden. Die Strategieparameter beinhalten die Varianzen und Kovarianzen der n-dimensionalen Dichte der Gaussverteilung, die der Mutation zugrunde liegt. Da sie im Laufe der Suche an die Fitnesslandschaft adaptiert werden, werden sie auch als *internes Modell* der Fitnesslandschaft bezeichnet.

Die Evolutionsstrategien unterscheiden drei Varianten der Schrittweitensteuerung. Die einfachste Evolutionsstrategie verwendet einen Strategieparameter für alle Objektvariablen. Die zweite Variante besitzt für jede Objektvariable einen eigenen Strategieparameter. Die aufwendigste Evolutionsstrategie verwendet pro Individuum $n \cdot (n + 1)/2$ freie Parameter, wobei n die Dimension der Zielfunktion ist.

Die jeweilige Kodierung einer Evolutionsstrategie wird durch die Anzahl der Standardabweichungen $n_{\sigma \in \{1,\ldots,n\}}$ und die Anzahl der Rotationswinkel n_α spezifiziert. Falls keine Rotationswinkel verwendet werden, ist $n_\alpha = 0$. Ansonsten ist n_α eine Funktion von n und n_σ. Für $n_\sigma = n$ ist $n_\alpha = n \cdot (n \Leftrightarrow 1)/2$.

$$\mathbb{C}^{\mathrm{ES}}_{n_\sigma, n_\alpha} = \left(I^{\mathrm{ES}}_{n_\sigma, n_\alpha}, \mathbb{E}^{\mathrm{ES}}_{n_\sigma, n_\alpha}, \mathbb{D}^{\mathrm{ES}}_{n_\sigma, n_\alpha} \right) \tag{3.38}$$

Der Raum der Individuen I kann in Abhängigkeit von n_σ und n_α verschiedene Formen annehmen. Im folgenden werden die vier am häufigsten verwendeten Varianten spezifiziert.

$$I^{\mathrm{ES}}_{n_\sigma, n_\alpha} = \begin{cases} \mathbb{R}^n \times \mathbb{R} & : \quad n_\sigma = 1, n_\alpha = 0 \\ \mathbb{R}^n \times \mathbb{R}^n & : \quad n_\sigma = n, n_\alpha = 0 \\ \mathbb{R}^n \times \mathbb{R}^{n_\sigma} \times \mathbb{R}^{n_\alpha} & : \quad 1 < n_\sigma < n, n_\alpha = (2n \Leftrightarrow n_\sigma)(n_\sigma \Leftrightarrow 1)/2 \\ \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{n \cdot (n-1)/2} & : \quad n_\sigma = n, n_\alpha = n \cdot (n \Leftrightarrow 1)/2 \end{cases} \tag{3.39}$$

Im folgenden wird $I^{\mathrm{ES}}_{n_\sigma,n_\alpha}$ durch I abgekürzt. Ein Individuum $\vec{a} \in I$ setzt sich aus den Objektvariablen $\vec{x}$, den Standardabweichungen $\vec{\sigma}$ und den Rotationswinkeln $\vec{\alpha}$ zusammen.

$$\vec{a} = \begin{cases} (\vec{x},\sigma) \in I & : \quad n_\sigma = 1, n_\alpha = 0 \\ (\vec{x},\vec{\sigma}) \in I & : \quad n_\sigma = n, n_\alpha = 0 \\ (\vec{x},\vec{\sigma},\vec{\alpha}) \in I & : \quad 1 < n_\sigma < n, n_\alpha = (2n \Leftrightarrow n_\sigma)(n_\sigma \Leftrightarrow 1)/2 \\ (\vec{x},\vec{\sigma},\vec{\alpha}) \in I & : \quad n_\sigma = n, n_\alpha = n \cdot (n \Leftrightarrow 1)/2 \end{cases} \tag{3.40}$$

Die Kodierungs- und Dekodierungsfunktion entsprechen jeweils der identischen Abbildung zwischen Objektvariablen und dem Teil des Individuums, das den Objektvariablen entspricht.

$$\mathbb{E}^{\mathrm{ES}}(\vec{x}) = \vec{a} \qquad mit \quad \vec{a} \in I, \vec{x} \in D \tag{3.41}$$

$$\mathbb{D}^{\mathrm{ES}}(\vec{a}) = \vec{x} \qquad mit \quad \vec{a} \in I, \vec{x} \in D \tag{3.42}$$

Die Fitnessfunktion $\mathbb{F}^{\mathrm{ES}}$ wertet die Objektvariablen eines Individuums aus und läßt sich daher direkt auf die Zielfunktion $f(\vec{x})$ abbilden.

$$\mathbb{F}^{\mathrm{ES}}(\vec{a}) = f(\mathbb{D}^{\mathrm{ES}}(\vec{a})) = f(\vec{x}) \tag{3.43}$$

3.3.2 Selektion

Die Evolutionsstrategien verwenden zwei deterministische, sogenannte ausschließende, Selektionsverfahren, von denen das eine elitär und das andere nicht-elitär ist. Beide Selektionsverfahren arbeiten wie die in Abschnitt 3.2.2 vorgestellte Schnittselektion. Aus einer Menge von Individuen werden die besten als Eltern der nächsten Generation ausgewählt. Die beiden Verfahren unterscheiden sich ausschließlich in den Mengen, aus denen selektiert wird.

Die elitäre Selektionsart wählt die besten Individuen aus der Menge der Eltern und Nachkommen der letzten Generation aus, während die nicht-elitäre Selektion ausschließlich aus der Menge der Nachkommen selektiert. Nach Schwefel wird die elitäre Selektion als $(\mu + \lambda)$-*Selektion* und die nicht-elitäre als (μ, λ)-*Selektion* bezeichnet ([Sch77]). μ und λ bezeichnen die Anzahl der Eltern bzw. der Nachkommen einer Evolutionsstrategie.

Eine Verallgemeinerung der $(\mu\overset{+}{,}\lambda)$-Selektion stellt die $(\mu, \kappa, \lambda, \rho)$-Selektion dar, die von Schwefel, Rudolph und Bäck eingeführt wurde ([SRB95]). Hierbei spezifiziert κ die maximale Lebenszeit eines Individuums und ρ die Anzahl der Eltern pro Nachkomme. Durch den Parameter κ können die beiden Standard-Selektionsverfahren spezifiziert werden. Mit $\kappa = 1$ ergibt sich die Komma- und mit $\kappa = \infty$ die Plus-Strategie.

$$\mathbb{S}^{\mathrm{ES,P}}_{\mu,\lambda} \quad : \quad I^{\mu+\lambda} \rightarrow I^{\mu} \tag{3.44}$$

$$\mathbb{S}^{\mathrm{ES,C}}_{\mu,\lambda} \quad : \quad I^{\lambda} \rightarrow I^{\mu} \tag{3.45}$$

Um zu verhindern, daß ungeeignete Strategieparameter, die zufällig zu einem gutem Objektparameter und damit zu einer guten Fitness führten, zu lange in der Population bleiben, verwenden die Evolutionsstrategen die nicht-elitäre (μ, λ)-Selektion. Die Stärke der Selektion hängt vom Verhältnis μ/λ ab, dessen optimaler Wert für das Kugelmodell von Schwefel mit $1/7$ bestimmt wurde ([Sch95]). Typische $\mu \Leftrightarrow \lambda$-Kombinationen sind $(8, 50)$ und $(15, 100)$.

3.3.3 Paarbildung

Die Evolutionsstrategien bilden ein Elterntupel, indem sie aus der Menge der selektierten Eltern rein zufällig ρ Individuen auswählen und sie anschließend wieder zurücklegen. Diese Paarung entspricht damit der in Abschnitt 3.2.3 beschriebenen Paarung durch Ziehen mit Zurücklegen.

$$\mathbb{P}_{\rho}^{\mathrm{ES}} = \mathbb{P}_{\rho}^{\mathrm{EA,R}} \tag{3.46}$$

3.3.4 Rekombination

Die Evolutionsstrategien arbeiten mit verschiedenen Rekombinationsoperatoren, die sich zum einen in der Art, in der eine Gleitkommazahl aus den entsprechenden Elementen der Elternindividuen konstruiert wird, und zum anderen in der Anzahl der Eltern, aus denen ein Nachkomme erzeugt wird (siehe ρ in Definition 1), unterscheiden ([BS93]). Die Rekombination wird nicht nur auf die Objektvariablen, sondern auch auf die Strategieparameter angewandt. Damit läßt sich der Rekombinationsoperator einer ES als Tripel von Rekombinationsoperatoren darstellen.

$$\mathbb{R}^{\mathrm{ES}} = (\mathbb{R}^{\mathrm{ES,x}}, \mathbb{R}^{\mathrm{ES},\sigma}, \mathbb{R}^{\mathrm{ES},\alpha}) : \quad I^{\rho} \to I, \tag{3.47}$$

wobei ρ die Anzahl der Elternindividuen angibt. Diese ausführliche Schreibweise läßt sich durch eine kompaktere, in der die Parameter zusammengefaßt und durch Kommata getrennt werden, verkürzen:

$$\mathbb{R}^{\mathrm{ES}} = (\mathbb{R}_{\Theta_x}^{\mathrm{ES,x}}, \mathbb{R}_{\Theta_\sigma}^{\mathrm{ES},\sigma}, \mathbb{R}_{\Theta_\alpha}^{\mathrm{ES},\alpha}) = \mathbb{R}_{\Theta_x,\Theta_\sigma,\Theta_\alpha}^{\mathrm{ES}} \tag{3.48}$$

Evolutionsstrategien verwenden eine *diskrete Rekombination* und eine *intermediäre Rekombination* ([Sch95]). Jede Rekombinationsart kann als sogenannter *sexueller* Operator, in der ein Nachkomme aus *zwei* Individuen erzeugt wird, oder als *multisexueller* Operator, in der ein Nachkomme aus mehreren Individuen der Population erzeugt wird, verwendet werden ([Sch95] S.146). Die diskrete Rekombination entscheidet für jede Komponente des Parametervektors zufällig, von welchem Elter der entsprechende Wert genommen wird. Die intermediäre Rekombination generiert einen Punkt, dessen einzelne Komponenten zwischen den entsprechenden Komponenten der jeweiligen Eltern liegen.

In der heutigen Evolutionsstrategie werden vier Rekombinationsarten verwendet ([BS95], [SRB95]). Die Rekombinationsarten heißen *global intermediär* (g), *allgemein lokal intermediär* (a), *lokal intermediär* (i) und *diskret* (d) und werden durch den Parameter ω spezifiziert. Im folgenden wird stellvertretend für alle Komponenten der drei Vektoren dargestellt, wie die i-te Komponente des Objektparameters x_i' durch Rekombination gebildet wird. In gleicher Weise werden die σ- und α-Teile der Individuen rekombiniert. Der Rekombinationsoperator, der auf jeden der drei Teile eines Individuums angewandt wird, ist folgendermaßen parametrisiert:

$$\mathbb{R}_{\omega,\varrho}^{\mathrm{ES,x}} : \qquad \text{mit} \quad 2 \leq \varrho \leq \rho \quad \text{und} \quad \omega \in \{g, a, i, d\}. \tag{3.49}$$

$$x_i' = \begin{cases} \frac{1}{\rho}\sum_{k=1}^{\rho} x_i^{\mathrm{P}k} & : \quad \omega = g \\ x_i^{\mathrm{P}1} + \chi_i \cdot (x_i^{\mathrm{P}2} \Leftrightarrow x_i^{\mathrm{P}1}) & : \quad \omega = a \\ x_i^{\mathrm{P}1} + 0.5 \cdot (x_i^{\mathrm{P}2} \Leftrightarrow x_i^{\mathrm{P}1}) & : \quad \omega = i \\ x_i^{\mathrm{P}i} & : \quad \omega = d \end{cases} \tag{3.50}$$

mit $\chi_i \sim \mathbf{U}([0,1])$ und $P_1, P_2 \sim \mathbf{U}(\{1,\ldots,\varrho\})$ für jeden Nachkommen bzw.
$P_i \sim \mathbf{U}(\{1,\ldots,\varrho\})$ für jedes i. ϱ bezeichnet die Anzahl der Eltern eines Nachkommen. $\vec{x}^{\mathrm{P}j}$ ist das j-te Elternindividuum aus dem jeweiligen ρ-Tupel.

Diese Rekombinationsoperatoren lassen sich für den sexuellen Fall ($\rho = 2$) folgendermaßen aus dem in Abschnitt 3.2.4 vorgestellten allgemeinen Rekombinationsoperator $\mathbb{R}^{\mathrm{EA}}$ modellieren:

$$\begin{aligned} \mathbb{R}_g^{\mathrm{ES}} &= \mathbb{R}^{\mathrm{EA}}_{*\mathrm{O1R}^*} \\ \mathbb{R}_a^{\mathrm{ES}} &= \mathbb{R}^{\mathrm{EA}}_{\mathrm{VU1R1.0}} \\ \mathbb{R}_i^{\mathrm{ES}} &= \mathbb{R}^{\mathrm{EA}}_{\mathrm{VU1R0.0}} \\ \mathbb{R}_d^{\mathrm{ES}} &= \mathbb{R}^{\mathrm{EA}}_{\mathrm{VO1R}^*} \end{aligned}$$

Da auf jede der Komponenten eines Individuums ein unterschiedlicher Rekombinationsoperator angewendet werden kann, stellt sich die Frage nach der optimalen Kombination der Rekombinationsoperatoren. Bäck und Schwefel schlagen für die Evolutionsstrategie vor, die Objektvariablen sexuell diskret und die Strategieparameter panmiktisch lokal intermediär zu rekombinieren ([Sch77], [BS95]):

$$\{\mathbb{R}_{d,2}^{\mathrm{ES},\mathrm{x}}, \mathbb{R}_{i,\mu}^{\mathrm{ES},\sigma}, \mathbb{R}_{-,0}^{\mathrm{ES},\alpha}\} \tag{3.51}$$

Die Wahl der Rekombinationsoperatoren hat erheblichen Einfluß auf die Wirkungsweise der Evolutionsstrategie. Dies machte Kursawe deutlich, indem er empirisch nachwies, daß die Eignung der Rekombinationsarten stark von der jeweiligen Zielfunktion und den Änderungsraten der Varianzen abhängt ([Kur95]). Er zeigte, daß die Evolutionsstrategie, in der Objektvariablen panmiktisch diskret und Strategieparameter panmiktisch lokal intermediär rekombiniert werden, beim hochdimensionalen Kugelmodell ($n = 100$) divergiert. Bei den Messungen, die im Rahmen der vorliegenden Arbeit gemacht wurden, trat dieser Divergenzeffekt bei einigen hochdimensionalen Zielfunktionen auch mit der Standardrekombination aus obiger Gleichung auf. Durch die ausschließliche Verwendung sexueller Rekombinationsoperatoren konnte dieser Effekt bei den meisten hier untersuchten Zielfunktionen beseitigt werden. Aus diesem Grund wird im folgenden ein rein sexueller Rekombinationsoperator verwendet, der im Unterschied zu dem in Gleichung 3.51 empfohlenen Rekombinationsoperator nur zwei Eltern für die Rekombination der Strategieparameter verwendet.

$$\mathbb{R}^{\mathrm{ES}} = \{\mathbb{R}_{d,2}^{\mathrm{ES},\mathrm{x}}, \mathbb{R}_{i,2}^{\mathrm{ES},\sigma}, \mathbb{R}_{-,0}^{\mathrm{ES},\alpha}\} \tag{3.52}$$

Die obige Spezifikation kann durch eine kompaktere Schreibweise vereinfacht werden. Zwischen sexueller und panmiktischer Rekombination wird durch Groß-/Kleinschreibung unterschieden.

$$\mathbb{R}^{\mathrm{ES}} = \{\mathbb{R}_{di\_}^{\mathrm{ES}}\} \tag{3.53}$$

Die in dieser Arbeit verwendete Evolutionsstrategie entspricht der in [BS95] beschriebenen Evolutionsstrategie. Die Resultate sind aufgrund der unterschiedlichen Behandlung von Strategieparametern nicht mit den Messungen von Schwefel in [Sch77] zu vergleichen. Schwefel, der in der ursprünglichen Evolutionsstrategie für die Mutation der Strategieparameter nur einen generellen Multiplikator einsetzte und die Ausdifferenzierung zu individuellen Strategieparametern ausschließlich der Rekombination überließ, zeigte, daß selbst bei $n = 1000$ Objektvariablen keine Divergenz eintrat ([Sch77]).

3.3.5 Mutation

Die Mutation wird, ebenso wie die Rekombination, auf Objekt- und Strategieparameter angewendet. Der Mutationsoperator hängt von der jeweiligen Kodierung und von bis zu zwei externen Parametern ab ([BS93], [Bäc96]).

$$\mathbb{M}^{\mathrm{ES}}_{n_\sigma, n_\alpha, \tau_1, \tau_0, \tau, \beta} : I \to I \tag{3.54}$$

Im folgenden wird mit $\mathbf{N}(0,1)$ eine eindimensionale normalverteilte Zufallsvariable mit Mittelwert 0 und Standardabweichung 1 bezeichnet. Der optionale Index deutet an, daß die Zufallsvariable für jeden Wert des Index neu generiert wird. $\vec{\mathbf{N}}(\vec{0}, \vec{\sigma}, \vec{\alpha})$ steht für die Realisierung der n-dimensionalen Normalverteilung mit Erwartungsvektor $\vec{0}$, Standardabweichungen $\vec{\sigma}$ und Rotationswinkeln $\vec{\alpha}$.

Falls $n_\sigma = 1$ und $n_\alpha = 0$:

$$\begin{aligned} \sigma_i' &= \sigma_i \cdot \exp(\tau_1 \cdot \mathbf{N}(0,1)) \\ x_i' &= x + \sigma_i' \cdot \mathbf{N}_i(0,1) \end{aligned} \tag{3.55}$$

Falls $n_\sigma = n$ und $n_\alpha = 0$:

$$\begin{aligned} \sigma_i' &= \sigma_i \cdot \exp(\tau_0 \cdot \mathbf{N}(0,1) + \tau \cdot \mathbf{N}_i(0,1)) \\ x_i' &= x + \sigma_i' \cdot \mathbf{N}_i(0,1) \end{aligned} \tag{3.56}$$

Falls $n_\sigma = n$ und $n_\alpha = n \cdot (n \Leftrightarrow 1)/2$:

$$\begin{aligned} \sigma_i' &= \sigma_i \cdot \exp(\tau_0 \cdot \mathbf{N}(0,1) + \tau \cdot \mathbf{N}_i(0,1)) \\ \alpha_j' &= \alpha_j + \beta \cdot \mathbf{N}_j(0,1) \\ \vec{x}' &= \vec{x} + \vec{\mathbf{N}}(\vec{0}, \vec{\sigma}', \vec{\alpha}') \end{aligned} \tag{3.57}$$

Die Konstanten τ_1, τ_0, τ und β in Gleichung 3.58 beeinflussen die Geschwindigkeit, mit der die Schrittweiten angepaßt werden. Die optimalen Werte der Proportionalitätsfaktoren sind von der jeweiligen Zielfunktion abhängig. Sie gelten jedoch als äußert robust. Schwefel schlägt folgende Belegungen vor ([Sch95] S.144, [Bäc96] S.72).

$$\begin{aligned} \tau_1 &\sim \left(\sqrt{n}\right)^{-1} \\ \tau_0 &\sim \left(\sqrt{2n}\right)^{-1} \\ \tau &\sim \left(\sqrt{2\sqrt{n}}\right)^{-1} \\ \beta &\approx 0.0873 \end{aligned} \tag{3.58}$$

Der Wert von β entspricht einem Winkel von 5°. Um eine Stagnation infolge zu kleiner Werte der Strategieparameter zu verhindern, werden zwei Randbedingungen eingeführt. Die erste Randbedingung gibt den kleinsten Wert für einen Strategieparameter vor, der nicht unterschritten werden darf. Die andere Randbedingung betrifft das Verhältnis zwischen dem Wert der Objektvariablen und dem dazugehörigen Strategieparameter ([Sch95], S. 113).

$$\sigma_i \geq \epsilon_a \qquad \text{und} \qquad \sigma_i \geq \epsilon_b \cdot |x_i| \tag{3.59}$$

Schwefel verwendet in der Implementierung OptimA als Voreinstellung die Werte $\epsilon_a = 10^{-30}$ und $\epsilon_b = 10^{-9}$ ([Sch95]).

3.3.6 Spezifikation von ES-Objekten

Eine Evolutionsstrategie läßt sich durch die in Abschnitt 3.1.4 eingeführte EA-Syntax folgendermaßen spezifizieren: Zunächst wird eine allgemeine Population einer ES eingeführt, aus der dann Populationen der speziellen ES-Instanzen abgeleitet werden.

Spezifikation 1 $\mathsf{POP}^{\mathsf{ES}}$ – *ES-Population.*

$$\mathsf{POP}^{\mathsf{ES}}\ (\lambda, \mathbb{S}, n_\sigma, n_\alpha, \mathbb{R}_x\mathbb{R}_\sigma\mathbb{R}_\alpha, [\tau_1, \tau_0, \tau, \beta]) ::=$$
$$\mathbf{POP}\left(\mathbf{IND}\left(\mathbb{C}^{\mathrm{ES}}_{n_\sigma, n_\alpha}, \mathbb{R}^{\mathrm{ES}}_{\mathbb{R}_x\mathbb{R}_\sigma\mathbb{R}_\alpha}, \mathbb{M}^{\mathrm{ES}}_{n_\sigma, n_\alpha, [\tau_1, \tau_0, \tau, \beta]}\right)^{\lambda}, \mathbb{S}, \mathbb{P}^{\mathrm{EA,R}}, \mathbb{I}^{\mathrm{EA,U}}\right)$$

Basierend auf dieser Spezifikation werden nun drei typische ES-Varianten beschrieben, die sich in der Anzahl der Strategieparameter unterscheiden. Die einfachste Variante einer Evolutionsstrategie `es1` arbeitet mit nur einem Strategieparameter, der für alle Objektvariablen verwendet wird. Die zweite Evolutionsstrategie `es2` hat für jede Objektvariable einen eigenen Strategieparameter. Die dritte und aufwendigste Evolutionsstrategie `es3` arbeitet mit Rotationswinkeln und kann sich so auch an beliebig orientierte Täler anpassen.

Spezifikation 2 $\mathsf{POP}^{\mathsf{ES1}}$ – *ES-Population mit einem Strategieparameter.*

$$\mathsf{POP}^{\mathsf{ES1}}(\lambda, \mathbb{S}, [\tau_1]) ::= \mathsf{POP}^{\mathsf{ES}}\ (\lambda, \mathbb{S}, 1, 0, di\_\, [\tau_1])$$

$$\mathsf{ES1}(\lambda, \mu, \mathbb{T}) ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{ES1}}(\lambda, \mathbb{S}^{\mathrm{ES,C}}_{\mu}), \mathbb{T}\right)$$

Spezifikation 3 $\mathsf{POP}^{\mathsf{ES2}}$ – *ES-Population mit n Strategieparametern.*

$$\mathsf{POP}^{\mathsf{ES2}}(\lambda, \mathbb{S}, [\tau_0, \tau]) ::= \mathsf{POP}^{\mathsf{ES}}\ (\lambda, \mathbb{S}, n, 0, di\_\, [\tau_0, \tau])$$

$$\mathsf{ES2}(\lambda, \mu, \mathbb{T}) ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{ES2}}(\lambda, \mathbb{S}^{\mathrm{ES,C}}_{\mu}), \mathbb{T}\right)$$

Spezifikation 4 $\mathsf{POP}^{\mathsf{ES3}}$ – *ES-Population mit Rotationswinkeln.*

$$\mathsf{POP}^{\mathsf{ES3}}(\lambda, \mathbb{S}, [\tau_0, \tau, \beta]) ::= \mathsf{POP}^{\mathsf{ES}}\left(\lambda, \mathbb{S}, n, \tfrac{n\cdot(n-1)}{2}, di\_, [\tau_0, \tau, \beta]\right)$$

$$\mathsf{ES3}(\lambda, \mu, \mathbb{T}) ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{ES3}}(\lambda, \mathbb{S}_{\mu}^{\mathrm{ES,C}}), \mathbb{T}\right)$$

Die Parameter τ_1, τ_0 und τ sind optional. Falls sie nicht explizit angegeben sind, erhalten sie die in Gleichung 3.58 angegeben Werte. Bei der Wahl der geeigneten Evolutionsstrategie ist zu beachten, daß mit zunehmender Anzahl von Strategieparametern gleichzeitig die Mächtigkeit des Verfahrens, aber auch die Länge der Adaptionsphase zunehmen.

3.4 Genetische Algorithmen

Die Genetischen Algorithmen zählen zu den populärsten Vertretern Evolutionärer Algorithmen. Nach Einführung des Evolutionary Programming wurden sie etwa zeitgleich mit den Evolutionsstrategien von Holland in den U.S.A. entwickelt ([Hol75], [DJ75]). Da die Entwicklung beider Evolutionärer Algorithmen unabhängig voneinander stattfand, unterscheiden sich die Genetischen Algorithmen in Kodierung und Operatoren grundlegend von den Evolutionsstrategien. Obwohl die Intention der Entwickler der Entwurf eines Verfahrens für allgemeine Adaptationsprozesse war, liegt das Hauptanwendungsgebiet der Genetischen Algorithmen heutzutage im Bereich der numerischen Optimierung.

Im folgenden wird die Grundform des Genetischen Algorithmus, die als sogenannter *kanonischer* Genetischer Algorithmus bezeichnet wird, für das Problem der kontinuierlichen Parameteroptimierung beschrieben. Hierbei werden auch einige Erweiterungen, die für den praktischen Einsatz unabdingbar sind, vorgestellt. Der resultierende Genetische Algorithmus entspricht im wesentlichen der GA-Implementierung GENESIS von Grefenstette ([Gre90]).

3.4.1 Kodierung und Fitnessfunktion

Genetische Algorithmen kodieren die Individuen als Bitstrings fester Länge. Ein Bitstring repräsentiert damit eine mögliche Lösung des zu optimierenden Problems. Im folgenden werden zwei mögliche Kodierungen für die kontinuierliche Parameteroptimierung beschrieben.

$$\mathbb{C}_{\mathrm{K,l}}^{\mathrm{GA}} = (I_{\mathrm{l}}^{\mathrm{GA}}, \mathbb{E}_{\mathrm{K,l}}^{\mathrm{GA}}, \mathbb{D}_{\mathrm{K,l}}^{\mathrm{GA}}) \tag{3.60}$$

$$I_{\mathrm{l}}^{\mathrm{GA}} \in \mathbb{B}^m \tag{3.61}$$

mit $m = n \cdot l$ und $\vec{a} = (a_1, \ldots, a_m) \in I$.

Im folgenden wird I^{GA} als I abgekürzt. Ein Individuum $\vec{a} \in I$ repräsentiert n Objektparameter, die jeweils durch einen Teilstring $\vec{b} \in \mathbb{B}^l$ der Länge l von $\vec{a}$ dargestellt werden.

$$\vec{a} = (\vec{b}_1, \ldots, \vec{b}_n) \tag{3.62}$$

mit $\vec{b}_i \in \mathbb{B}^l$ und $\vec{b}_i = (a_{(i-1)\cdot l+1}, \ldots, a_{i\cdot l})$.

Für die in dieser Arbeit behandelten Problemstellungen der kontinuierlichen Parameteroptimierung wird jeder binäre Teilstring $\vec{b}_i$ als Kodierung einer ganzen Zahl aufgefaßt, aus der dann der reelle Wert des Objektparameters berechnet wird. Zur Berechnung des reellen Wertes aus der ganzen Zahl muß für jeden Parameter x_i eine obere und untere Schranke mit $u_i \leq v_i$ angegeben werden. Der durch diese Schranken spezifizierte Bereich wird nun in diskrete Teilbereiche zerlegt, deren Größe Δx_i durch die Schranken und die größte ganze Zahl, die durch den Bitstrings $\vec{b}_i$ dargestellt werden kann, bestimmt ist:

$$\Delta x_i = \frac{v_i \Leftrightarrow u_i}{2^l \Leftrightarrow 1} \tag{3.63}$$

Für eine beliebige Binärkodierung $K : \mathbb{B}^l \rightarrow \mathbb{Z}$ ergibt sich folgende Dekodierungsfunktion $\mathbb{D}^{\mathrm{GA}}_{\mathrm{K},l}$:

$$\mathbb{E}^{\mathrm{GA}}_{\mathrm{K},l} \quad : \quad \vec{b}_i = K^{-1}\left(\frac{x_i \Leftrightarrow u_i}{\Delta x_i}\right) \tag{3.64}$$

$$\mathbb{D}^{\mathrm{GA}}_{\mathrm{K},l} \quad : \quad x_i = u_i + \Delta x_i \cdot K(\vec{b}_i) \tag{3.65}$$

Als Kodierungsfunktion kann die Binärkodierung zur Basis Zwei verwendet werden:

$$K_{\mathrm{S}}(\vec{b}) = \sum_{j=0}^{l-1}\left(b_j \cdot 2^j\right) \tag{3.66}$$

Diese Binärkodierung hat den Nachteil, daß benachbarte ganzzahlige Werte große Hammingdistanzen aufweisen können. Die Graykodierung, bei der die Bitstrings benachbarter ganzzahliger Werte Hammingdistanz eins haben, entschärft das Problem dieser sogenannten *Hamming-Cliffs*. Gleichwohl kann die Änderung eines einzelnen Bits des Bitstrings zu einer beliebig großen Änderung des entsprechenden ganzzahligen Wertes führen. Die Gray-Code basierte Kodierungsfunktion ist folgendermaßen spezifiziert:

$$K_{\mathrm{G}}(\vec{b}) = \sum_{j=0}^{l-1}\left(\gamma^{-1}(b_j) \cdot 2^j\right) \tag{3.67}$$

mit der Funktion γ^{-1}, die einen Bitstring der Standardkodierung in einen entsprechenden Bitstring der Graykodierung konvertiert. Die Konvertierungsfunktionen zwischen Standard- und Graykodierung sind in folgender Gleichung angegeben. $\vec{a}$ und $\vec{b}$ stellen dabei Bitstrings der Länge n der Standardkodierung bzw. Graykodierung dar. Der Operator $\oplus : \mathbb{B}^n \rightarrow \mathbb{B}$ beschreibt eine Addition aller Operanden mit anschließender Modulo 2 Berechnung ([Wri91]).

$$\gamma : b_i \quad = \quad \begin{cases} a_i & : \quad i = 1 \\ a_{i-1} \oplus a_i & : \quad i > 1 \end{cases} \qquad \text{Standard nach Gray} \tag{3.68}$$

$$\gamma^{-1} : a_i \quad = \quad \bigoplus_{j=1}^{i} b_j \qquad \text{Gray nach Standard} \tag{3.69}$$

Im folgenden sind die Dekodierungsfunktionen für Standard- und Graykodierung explizit angegeben:

$$\mathbb{D}^{\mathrm{GA}}_{\mathrm{S},l} \quad : \quad x_i = u_i + \Delta x_i \cdot \left(\sum_{j=0}^{l-1} b_{i_j} \cdot 2^j\right) \tag{3.70}$$

$$\mathbb{D}_{\mathrm{G,l}}^{\mathrm{GA}} \quad : \quad x_i = u_i + \Delta\, x_i \cdot \left(\sum_{j=0}^{l-1} \left(\bigoplus_{k=1}^{l} b_{i_k} \right) \cdot 2^j \right) \tag{3.71}$$

3.4.2 Selektion

Im Gegensatz zu den Evolutionsstrategien verwenden die Genetische Algorithmen ein stochastisches Selektionsschema. Die sogenannte *proportionale Selektion* wurde von Holland entwickelt und basiert darauf, daß die Wahrscheinlichkeit für ein Individuum, Nachkommen zu erzeugen, mit zunehmender Fitness steigt ([Hol75]). Die Wahrscheinlichkeit ist dabei proportional zu einer Größe, die die Fitness des jeweiligen Individuums widerspiegelt. In der ursprünglichen Form ist die sogenannte *Selektionswahrscheinlichkeit* oder *Überlebenswahrscheinlichkeit* eines Individuums proportional zu seiner relativen Fitness. Rudolph zeigte in seiner Dissertation, daß GA aufgrund dieser Selektionsart, sofern sie keine elitäre Komponente enthalten, niemals konvergieren, sondern am Ende flukturierend stagnieren ([Rud96]).

Fitnessbasierte proportionale Selektion $\mathbb{S}_{\omega,\epsilon}^{\mathrm{GA,F}}$

Die fitnessbasierte proportionale Selektion weist jedem Individuum eine *Selektionswahrscheinlichkeit* $p_{\mathbb{S}}$ zu, die folgendermaßen definiert ist:

$$p_{\mathbb{S}}(\vec{a}_i(t)) = \frac{\mathbb{F}(\vec{a}_i(t))}{\sum_{j=1}^{\lambda} \mathbb{F}(\vec{a}_j(t))} \tag{3.72}$$

Da dieser Mechanismus nur für positive Fitnesswerte und für Maximierungsprobleme verwendet werden kann, müssen die Fitnesswerte für den allgemeinen Fall skaliert werden. In der Literatur findet man verschiedene Skalierungsfunktionen $\delta : \mathbb{R} \rightarrow \mathbb{R}^+$, die einen beliebigen Fitnesswert auf eine positive reelle Zahl abbilden. Die Verfahren reichen von linearen über logarithmische bis zu exponentiellen Skalierungsfunktionen. Die Skalierungsfunktion hat, da die Überlebenswahrscheinlichkeit in Gleichung 3.72 nur auf Basis der skalierten Fitnesswerte berechnet wird, erheblichen Einfluß auf den Selektionsprozeß. Die Skalierungsfunktion läßt sich damit einschließlich ihrer eigenen Parameter als externer Parameter der Selektion auffassen.

Im folgenden wird die sogenannte *lineare dynamische* Skalierungsmethode von Grefenstette verwendet, die auch in seiner Simulationsumgebung GENESIS implementiert ist ([Gre90]). Hierbei wird der Fitnesswert skaliert, indem seine Differenz zu dem schlechtesten Fitnesswert der letzten ω Generationen gebildet und anschließend durch einen *Normalisierungsfaktor* gewichtet wird. Im folgenden bezeichnet f den Fitnesswert des Individuums und $f_{\max}$ die maximale (schlechteste) Fitness aller Individuen der letzten ω Generationen.

$$\delta_{\mathrm{LD},\omega}(f, \{P(u)\}_{u=t}^{t-\omega}, \epsilon) \quad = \quad (\hat{f}_{max} \Leftrightarrow f) \cdot \gamma \tag{3.73}$$

$$\hat{f}_{\max} \quad = \quad \begin{cases} \epsilon & : \quad f_{\max} = 0 \\ f_{\max} \cdot (1+\epsilon) & : \quad f_{\max} > 0 \\ f_{\max} \cdot (1 \Leftrightarrow \epsilon) & : \quad f_{\max} < 0 \end{cases} \tag{3.74}$$

$$f_{\max} = \max\left\{f_i \mid \vec{a}_i \in \{P(u)\}_{u=t}^{t-\omega}\right\} \tag{3.75}$$

$$\gamma = \frac{1}{\hat{f}_{\max} \Leftrightarrow \bar{f}} \tag{3.76}$$

Durch die Selektionsgleichung (Gleichung 3.72) wird jedem Individuum implizit ein Erwartungswert für die Anzahl seiner Nachkommen zugewiesen. Dieser Erwartungswert ist definiert als

$$\eta(\vec{a}_i) = \lambda \cdot p_{\mathbb{S}}(\vec{a}_i(t)). \tag{3.77}$$

An dieser Stelle tritt das Problem auf, daß die ermittelten reellwertigen Erwartungswerte in ganze Zahlen, die die tatsächliche Anzahl der Nachkommen eines jeden Individuums angeben, umgerechnet werden müssen. Um den bei dieser Umrechnung nicht vermeidbaren Fehler möglichst gering zu halten, sind eine Vielzahl von sogenannten *Spinning-Wheel*- oder *Roulette-Wheel*-Verfahren entwickelt worden ([Gol89]).

Ein Verfahren, das aufgrund seiner Effizienz und Genauigkeit in der Praxis häufig verwendet wird, ist das von Baker entwickelte *Stochastic-Universal-Sampling* (SUS)([Bak87]). Der folgende Algorithmus beschreibt das SUS-Verfahren, das in GENESIS und PeGAsuS implementiert wurde.

Algorithmus 3.2 *Stochastic Universal Sampling*

SUS($\mathbf{P}(t)$)
1 $P^{\mathbb{S}}(t+1) \leftarrow \emptyset$
2 $sum \leftarrow 0$
3 $ptr \leftarrow \mathrm{U}([0,1])$
4 **for** $i \leftarrow 0$ **to** λ
5 **do while** $sum > ptr$
6 **do** $sum \leftarrow sum + \eta(\vec{a}_i)$
7 $P^{\mathbb{S}}(t+1) \leftarrow P^{\mathbb{S}}(t+1) \cup \vec{a}_i$
8 **return** $P^{\mathbb{S}}(t+1)$

Im Gegensatz zu den üblichen Spinning-Wheel-Verfahren, in der ein einziger Zeiger den Gewinner identifiziert, verwendet SUS λ Zeiger, die jeweils 1.0 voneinander entfernt sind. Dadurch wird erreicht, daß die wirkliche Anzahl der Nachkommen eines Individuums entweder die nächstkleinere oder nächstgrößere ganze Zahl des Erwartungswertes ist. Bei der Implementierung ist zu beachten, daß die zu selektierende Population zufällig angeordnet ist.

Die hier vorgestellte proportionale Selektion, die auch *Erwartungswertmodell* genannt wird ([Bak85]), hat den Nachteil, daß ein *Superindividuum* aufgrund seiner relativ guten Fitness alle anderen Individuen verdrängt und den Fortschritt der Population bremst. Aus diesem Grund wurden die sogenannten rangbasierten Selektionsverfahren eingeführt. Der Erwartungswert der Nachkommen eines Individuums basiert hierbei auf dem Rang seiner Fitness.

Rangbasierte proportionale Selektion $\mathbb{S}^{\mathrm{GA,R}}_{\mathrm{max}}$

Mit dem von Baker stammenden Verfahren ([Bak85]), das auch in GENESIS und PeGAsuS implementiert ist, wird der Erwartungswert eines Individuums mit Rang ρ nach folgender Gleichung bestimmt[2]:

$$\eta(\rho) = min + \rho \cdot inc \tag{3.78}$$

mit $min = 2.0 \Leftrightarrow max, inc = \frac{2.0 \cdot (max - 1.0)}{\lambda}$ und $low = \frac{inc}{2}$.

Der Eingabeparameter max liegt im Intervall $[1, 2]$ und bestimmt die Härte der Selektion. Als Voreinstellung schlägt Baker den Wert 1.1 vor. Die untere Grenze für den Erwartungswert η ist low. Die Fläche unter der durch $\eta(\rho)$ bestimmten Geraden entspricht im Intervall $[0, \lambda]$ der Populationsgröße λ.

3.4.3 Paarbildung

GA ordnen jedes der selektierten Individuen zufällig in ein Elternpaar, so daß alle selektierten Individuen in eine Paarung einbezogen werden. Die hier verwendete Paarbildung entspricht damit dem in Abschnitt 3.2.3 beschriebenen Paarbildungsverfahren $\mathbb{P}^{\mathrm{EA,S}}_{\rho}$, das die Paare durch Ziehen ohne Zurücklegen zusammenstellt.

$$\mathbb{P}^{\mathrm{GA}} = \mathbb{P}^{\mathrm{EA,S}}_{2} \tag{3.79}$$

Der Parameter ρ des Paarbildungsverfahrens $\mathbb{P}^{\mathrm{EA,S}}$ spezifiziert die Anzahl der Eltern eines Nachkommen. Da GA in der Regel sexuelle Rekombinationsoperatoren verwenden, wird er auf 2 gesetzt.

3.4.4 Rekombination

Der Rekombinationsoperator, der in GA Crossoveroperator genannt wird, gilt hier als der zentrale genetische Operator. Die Idee, die den genetischen Algorithmen zugrunde liegt, besteht darin, daß die Kombination guter Teillösungen zu besseren Gesamtlösungen führt. Diese Idee basiert auf der Hypothese sogenannter *Building Blocks*.

Rekombinationsoperatoren in GA generieren aus den Bitstrings, die die Elternindividuen repräsentieren, einen neuen Bitstring. Der Rekombinationsoperator wird in der Regel mit einer bestimmten Wahrscheinlichkeit p_r auf jedes Elternpaar angewandt. Der Wert der Rekombinationswahrscheinlichkeit ist in der Literatur nicht eindeutig festgelegt. Man findet Werte zwischen $p_r = 0.6$ ([DJ75]) und $p_r = 0.95$ ([Gre86]).

Im folgenden werden die drei wichtigsten Rekombinationsoperatoren, die für die kontinuierliche Parameteroptimierung verwendet werden, vorgestellt. Alle Operatoren werden mit einer Wahrscheinlichkeit von p_r auf ein Elternpaar angewendet.

One-Point-Crossover $\mathbb{R}^{\mathrm{GA,1P}}_{p_r}$

Das *One-Point-Crossover* ist der älteste Rekombinationsoperator der GA und wurde von Holland eingeführt ([Hol75]). Dieser Operator wählt zufällig eine Position in den Bitstrings, die die beiden Elternindividuen repräsentieren, aus. Der Nachkomme wird nun aus dem

[2] Die Ränge reichen von 0 bis $\lambda - 1$. Das Individuum mit dem besten Fitnesswert hat den höchsten Rang $\lambda - 1$.

ersten Teil des einen und aus dem zweiten Teil des anderen Elternteils zusammengestellt. Im folgenden ist dargestellt, wie aus den beiden Individuen $\vec{x}$ und $\vec{y}$ der Nachkomme $\vec{z}$ erzeugt wird.

$$\vec{z} = (x_1, \ldots, x_{\chi-1}, x_\chi, y_{\chi-1}, \ldots, y_m) \tag{3.80}$$

mit $\chi \sim \mathbf{U}(\{1, m \Leftrightarrow 1\})$.

Multi-Point-Crossover $\mathbb{R}_{p_r}^{\mathrm{GA,mP}}$

Das Multi-Point-Crossover stellt eine Verallgemeinerung des One-Point-Crossovers dar. Statt einer Position werden hier mehrere Positionen bestimmt, zwischen denen die Teilstrings abwechselnd aus dem einen oder anderen Elternindividuum ausgewählt werden.

$$z_i = \begin{cases} x_i & : \quad \forall_i(\chi_k < i \leq \chi_{k+1}), k \leq m, k \quad \text{ungerade} \\ y_i & : \quad sonst \end{cases} \tag{3.81}$$

mit $\chi \sim \mathbf{U}(\{1, m\})$. Das oben vorgestellte One-Point-Crossover läßt sich als Extremfall des Multi-Point-Crossover, mit $m = 1$, auffassen.

Uniform-Crossover $\mathbb{R}_{p_r}^{\mathrm{GA,U}}$

Das Uniform-Crossover entscheidet für jedes Bit des zu erzeugenden Bitstrings mit gleicher Wahrscheinlichkeit, ob es den Wert von dem einen oder anderen Elter übernimmt.

$$z_i = \begin{cases} x_i & : \quad \chi > 0.5 \\ y_i & : \quad \chi \leq 0.5 \end{cases} \tag{3.82}$$

mit $\chi \sim \mathbf{U}([0, 1])$. Das Uniform-Crossover erhält keine Nachbarschaften in den Bitstrings und stellt damit den allgemeinsten der drei Rekombinationsoperatoren dar. Die Güte des jeweiligen Rekombinationsoperators hängt stark von der Kodierung und dem zu lösenden Optimierungsproblem ab, so daß keine allgemeine Aussage über den Nutzen dieser Operatoren gemacht werden kann. GENESIS 5.0 arbeitet standardmäßig mit einem Two-Point-Crossover, der mit einer Wahrscheinlichkeit von 0.6 auf jedes Elternpaar angewendet wird.

3.4.5 Mutation

Der Mutationsoperator wurde lange Zeit in den Genetischen Algorithmen als Hintergrundoperator angesehen, der im Vergleich zur Rekombination sehr selten angewendet wird. Die Auswirkung einer Mutation besteht in der Invertierung eines einzelnen Bits[3]. Der hier verwendete Mutationsoperator $\mathbb{M}_{p_m}^{\mathrm{GA}}$ invertiert jedes Bit mit der Wahrscheinlichkeit p_m.

$$a_i' = \begin{cases} a_i & : \quad \chi_i > p_m \\ 1 \Leftrightarrow a_i & : \quad \chi_i \leq p_m \end{cases} \tag{3.83}$$

mit $\chi_i \sim \mathbf{U}([0, 1])$.

[3] Grefenstette versteht unter einer Bitmutation das Ersetzen eines Bits durch einen zufälligen Wert aus dem zugrundeliegenden Alphabet ([Gre90]). Sein Argument ist, daß er bei einer Mutationswahrscheinlichkeit von 1.0 ein neues zufälliges Individuum erzeugt, während ansonsten ein zu dem Originalindividuum komplementäres Individuum entstehen würde.

3.4.6 Spezifikation von GA-Objekten

Im folgenden wird eine GA-Population spezifiziert, aus der sich typische GA-Objekte ableiten lassen. Die in dieser Arbeit verwendeten GA-Instanzen arbeiten mit einer Gray-Kodierung der Länge 30 und einem Two-Point-Crossover. Die beiden GA unterscheiden sich lediglich in der Art der Selektion. Die GA-Klasse ga1 verwendet eine rangbasierte, die GA-Klasse ga2 eine fitnessbasierte Selektion.

Spezifikation 5 $\mathsf{POP}^{\mathsf{GA}}$ – *GA-Population.*

$$\mathsf{POP}^{\mathsf{GA}}\ (\lambda, K, l, \mathbb{S}, \mathbb{R}, p_m) ::= \mathbf{POP}\left(\mathbf{IND}\left(\mathbb{C}^{\mathrm{GA}}_{\mathrm{K},\,l}, \mathbb{R}, \mathbb{M}^{\mathrm{GA}}_{p_m}\right)^{\lambda}, \mathbb{P}^{\mathrm{EA,S}}, \mathbb{S}, \mathbb{I}^{\mathrm{EA,U}}\right)$$

Spezifikation 6 $\mathsf{POP}^{\mathsf{GA1}}$ – *GA-Population mit rangbasierter Selektion.*

$$\mathsf{POP}^{\mathsf{GA1}}(\lambda, G, l, p_m) ::= \mathsf{POP}^{\mathsf{GA}}\ (\lambda, G, l, \mathbb{R}^{\mathrm{GA,2P}}_{0.6}, \mathbb{S}^{\mathrm{GA,R}}_{1.1}, p_m)$$

$$\mathsf{GA1}(\lambda, \mathbb{T}) ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{GA1}}(\lambda, G, 30, 0.001), \mathbb{T}\right)$$

Spezifikation 7 $\mathsf{POP}^{\mathsf{GA2}}$ – *GA-Population mit fitnessbasierter Selektion.*

$$\mathsf{POP}^{\mathsf{GA2}}(\lambda, G, l, p_m) ::= \mathsf{POP}^{\mathsf{GA}}\left(\lambda, G, l, \mathbb{R}^{\mathrm{GA,2P}}_{0.6}, \mathbb{S}^{\mathrm{GA,F}}_{LD,5}, p_m\right)$$

$$\mathsf{GA2}(\lambda, \mathbb{T}) ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{GA2}}(\lambda, G, 30, 0.001), \mathbb{T}\right)$$

3.5 Breeder Genetic Algorithm

Der *Breeder Genetic Algorithm (BGA)* ist ein Evolutionärer Algorithmus,der 1993 von Mühlenbein und Schlierkamp-Voosen ([MSV93b]) für die kontinuierliche Parameteroptimierung entwickelt wurde.

Der erste Teil des Namens *Breeder* (dtsch.: *Züchter*) betont die Beziehung dieses Algorithmus zur Vorgehensweise menschlicher Züchter. Evolutionäre Algorithmen haben ebenso wie Züchter das Ziel, Individuen bezüglich bestimmter Merkmale zu optimieren und bedienen sich dazu des Instruments der Selektion. Der BGA basiert, wie die ES, auf der von den menschlichen Züchtern entwickelten Schnittselektion und deren Theorie, die auf Vererbbarkeit und Selektionsintensität beruht ([MSV93b], [MSV94]). Durch die Schnittselektion, die der natürlichen Umweltselektion entspricht, unterscheidet sich der BGA von dem traditionellen GA, dessen proportionale Selektion eher die natürliche Paarungsselektion modelliert.

Der zweite Teil des Namens, *Genetic Algorithm*, verrät die Herkunft des heutigen BGA. Der Prototyp des heutigen BGA, der *Parallel Genetic Algorithm (PGA)*, arbeitete direkt auf der Bitkodierung für Gleitkommazahlen und verwendete dazu die in GA üblichen genetischen Operatoren ([MSB91]). Während der Entwicklung des BGA wurde

die Bitkodierung aufgegeben und die entsprechenden Operatoren auf Gleitkommaebene nachgebildet ([MSV93b]). Die resultierende diskrete Mutationsverteilung wurde in einer späteren Phase kontinuisiert ([SVM94]).

In der heutigen Form ist der BGA den Evolutionsstrategien wesentlich ähnlicher als dem GA. Er verwendet dieselbe Selektionsart und arbeitet direkt auf Gleitkommazahlen. Die Art der Mutationsverteilung stellt den wesentlichsten Unterschied zu den Evolutionsstrategien dar.

3.5.1 Kodierung und Fitnessfunktion

Der BGA verwendet die in Abschnitt 3.2.1 eingeführte identische Abbildung als Kodierungsfunktion. Ein Individuum des BGA besteht damit aus n Gleitkommazahlen.

$$\mathbb{C}^{\mathrm{BGA}} = \mathbb{C}^{\mathrm{EA}} \tag{3.84}$$

Die Fitnessfunktion kann damit direkt aus der Zielfunktion gewonnen werden.

$$\mathbb{F}^{\mathrm{BGA}}(\vec{a}) = f(\mathbb{D}^{\mathrm{BGA}}(\vec{a})) \tag{3.85}$$

3.5.2 Selektion

In Analogie zu der von menschlichen Züchtern verwendeten künstlichen Selektion arbeitet der BGA mit einer *Schnittselektion* (engl.: *truncation selection*) wie sie in Abschnitt 3.2.2 eingeführt wurde.

Schnittselektion $\mathbb{S}_{\vartheta}^{\mathrm{BGA}}$

Der BGA wählt aus der Menge der Nachkommen einschließlich dem besten Elter die besten ϑ Prozent aus. Die Selektion des BGA zählt damit zu den sogenannten *elitären* Selektionsverfahren, da das bislang beste gefundene Individuum solange in der Population bleibt, bis ein besserer Nachkomme erzeugt wird.

$$\mathbb{S}_{\vartheta}^{\mathrm{BGA}} = \mathbb{S}_{\vartheta,e}^{\mathrm{EA,TR}} \tag{3.86}$$

Diese Selektion entspricht einer modifizierten (μ, λ)-Selektion der ES, in der das beste Individuum künstlich erhalten bleibt. Die Lebensdauer eines Individuums ist damit nicht begrenzt.

Die Verwendung einer elitären Selektionsmethode wirft Probleme bei der Optimierung dynamischer oder verrauschter Funktionen auf. Da diese Funktionen denselben Argumenten zu verschiedenen Zeitpunkten unterschiedliche Funktionswerte zuordnen, kann der Fall eintreten, daß das bislang beste Individuum in der Population gehalten wird, obwohl sich seine Fitness verschlechtert hat. Die Lösung dieses Problems besteht in der erneuten Evaluierung des als bestes gekennzeichneten Individuums, bevor es in die Menge der zu selektierenden Individuen aufgenommen wird.

3.5.3 Paarbildung

Die Paare werden zufällig gleichverteilt aus diesen selektierten Individuen zusammengestellt. Die Fitness der Individuen hat keinen Einfluß auf die Zusammenstellung der Paare. Die einzige Restriktion besteht darin, daß ein Paar nicht aus zwei identischen Individuen bestehen darf. Ein Individuum kann in mehreren Paaren vorkommen. Damit entspricht die Paarbildung des BGA der in Abschnitt 3.2.3 eingeführten Paarbildung durch Ziehen ohne Zurücklegen innerhalb eines Tupels.

$$\mathbb{P}^{\mathrm{BGA}} = \mathbb{P}^{\mathrm{EA}}_{\mathrm{R}^*} \tag{3.87}$$

3.5.4 Rekombination

Der BGA verwendet verschiedene Arten von Rekombinationsoperatoren, zu denen neben der bekannten diskreten Rekombination weitere richtungs- und volumenorientierte Verfahren zählen. Grundsätzlich stehen alle Varianten des in Abschnitt 3.2.4 definierten allgemeinen Rekombinationsoperators $\mathbb{R}^{\mathrm{EA}}$ zur Verfügung. Die am häufigsten gebrauchten Rekombinationsoperatoren werden im folgenden mit eigenem Namen versehen und durch den allgemeinen Rekombinationsoperator modelliert.

Fuzzy-Rekombination $\mathbb{R}^{\mathrm{BGA,F}}_{\delta}$

Die *Fuzzy-Rekombination* ist ein volumenorientierter Rekombinationsoperator, der im Gegensatz zur diskreten Rekombination eine Dreiecksverteilung über die Ecken des durch die Eltern definierten Hypercubes legt und so die Diskretisierung aufhebt ([Voi92], [VMC95]).

Die Fuzzy-Rekombination, deren Dichtefunktion in Abbildung 4.4 auf Seite 71 dargestellt ist, läßt sich aus dem in Abschnitt 3.2.4 vorgestellten allgemeinem Rekombinationsoperator wie folgt ableiten.

$$\mathbb{R}^{\mathrm{BGA,F}}_{\delta} = \mathbb{R}^{\mathrm{EA}}_{\mathrm{VT2R}\delta} \tag{3.88}$$

BGA-Line-Rekombination $\mathbb{R}^{\mathrm{BGA,L}}_{\mathrm{k},\rho}$

Diese Rekombinationsart zählt zu den richtungsorientierten Rekombinationsverfahren, bei denen die Nachkommen auf der durch die beiden Eltern spezifizierten Geraden erzeugt werden.

Die Line-Rekombination unterscheidet sich in zwei Punkten von anderen häufig in EA verwendeten intermediären Rekombinationsverfahren. Zum einen werden die Nachkommen entlang der durch die Eltern spezifizierten Geraden nicht gleichverteilt, sondern durch eine spezielle für den BGA entwickelte Mutationsverteilung generiert (siehe Abschnitt 3.5.5). Zum anderen ist die Schrittweite nicht von der Entfernung der beiden Eltern abhängig. Die Positionen der beiden Eltern werden nur für die Orientierung, nicht aber für die Größe der Veränderung verwendet.

$$\boxed{x'_i = x_i^{\mathrm{P}1} + \beta \cdot \delta \cdot \frac{x_i^{\mathrm{P}2} \Leftrightarrow x_i^{\mathrm{P}1}}{\|\vec{x}^{\mathrm{P}1}, \vec{x}^{\mathrm{P}2}\|}} \tag{3.89}$$

wobei $\|\vec{x}, \vec{y}\|$ den euklidischen Abstand zwischen $\vec{x}$ und $\vec{y}$ bezeichnet. Die Zufallsvariablen $\delta \sim \mathbf{B}(k, \rho_i)$ und $\beta \sim \mathbf{U}(\{\Leftrightarrow 1, 1\})$, die Schrittweite und Richtung bestimmen, sind

nicht indiziert, da sie für alle Elemente des Objektvariablenvektors identisch sind. $\mathbf{B}(k, \rho_i)$ bezeichnet die in Abschnitt 3.5.5 eingeführte BGA-Mutationsverteilung. Die Wahl der Richtung muß bei diesem Operator nicht zufällig erfolgen, da aus den Fitnesswerten der beiden Eltern eine zu bevorzugende Richtung gewonnen werden kann. Falls das Optimum nicht zwischen beiden Eltern liegt, ist ein Schritt in Richtung des besseren Elter erfolgversprechender.

Die Line-Rekombination wird durch den in Abschnitt 3.2.4 eingeführten allgemeinen Rekombinationsoperator folgendermaßen modelliert.

$$\mathbb{R}^{\mathrm{BGA,L}}_{k,\rho} = \mathbb{R}^{\mathrm{EA}}_{\mathrm{LH3A},k,\rho} \tag{3.90}$$

3.5.5 Mutation

Der Mutationsoperator des BGA basiert auf einer Mutationsverteilung, die sich fundamental von anderen in Evolutionären Algorithmen verwendeten Mutationsverteilungen unterscheidet. Diese Verteilung, die sich als beidseitig beschränkte Hyperbelverteilung charakterisieren läßt, wird im folgenden als *BGA-Mutationsverteilung* bezeichnet.

Die Anzahl der gleichzeitig zu mutierenden Objektvariablen hängt von der jeweiligen Variante der BGA-Mutation ab. In der ursprünglichen Form wird jede Objektvariable mit einer Wahrscheinlichkeit von $p = 1/n$ mutiert, wodurch während einer Mutation durchschnittlich eine Objektvariable mutiert wird.

Die Mutation einer Objektvariablen besteht aus der Wahl der Richtung und der Wahl der Schrittweite. Die Richtung wird durch eine diskrete Zufallsvariable, die mit gleicher Wahrscheinlichkeit positiv oder negativ ist, bestimmt. Die Wahl der Schrittweite wird durch die zugrunde liegende Mutationsverteilung bestimmt.

BGA-Mutationsverteilung

Die Mutationsverteilung des BGA wurde aus der diskreten Mutationsverteilung des auf Bitstrings operierenden Prototypen des heutigen BGA, dem PGA ([MSB91]), entwickelt. Der PGA basiert wie der traditionelle Genetische Algorithmus auf einer Bitkodierung und verwendet auch dessen Operatoren. Die Bitkodierung des PGA entspricht der binären Gleitkommadarstellung der Rechnerarchitektur, so daß die explizite Kodierung und Dekodierung von Individuen entfällt.

Die Umstellung von dieser binären auf eine reelle Kodierung erfolgte, da die Anwendung bitorientierter Rekombinationsoperatoren auf die reellwertigen Objektparameter zu einer Nachbarschaftsbeziehung führt, die die Rekombination zu einem stark zufallsgetriebenen Operator macht. Um die Eigenschaften des bewährten Mutationsoperators zu erhalten, wurde der bitorientierte Mutationsoperator des PGA für Gleitkommazahlen modelliert ([MSV93b]).

Die Transformationsfunktion, die aus einem Vektor von Zufallszahlen eine Zufallszahl der diskreten Mutationsverteilung $\mathbf{B}^{\mathrm{d}}(k)$ erzeugt, lautet:

$$g^{\mathrm{d}}_k(\vec{\alpha}) = \sum_{j=0}^{k-1} \alpha_j \cdot 2^{-j} \tag{3.91}$$

wobei $\vec{\alpha}$ ein Vektor von Zufallszahlen mit der Wahrscheinlichkeitsfunktion $P(\alpha_j = 0) =$

$\frac{k-1}{k}$ und $P(\alpha_j = 1) = \frac{1}{k}$ ist. Diese diskrete Mutationsverteilung, die gleichverteilt Punkte aus der Menge $\{2^{-(k-1)}, \ldots, 2^{-1}, 2^0\}$ generiert, erfordert k Aufrufe des Zufallszahlgenerators. Der Parameter k, der den minimalen Wert der Verteilung festlegt, wird im folgenden als *Präzisionsparameter* bezeichnet. Da die Diskretisierung des Suchraums zu ungewünschten Nebeneffekten führt, wurde die Mutationsverteilung des BGA in einer weiteren Entwicklungsphase kontinuisiert und erhielt das folgende Aussehen:

Definition 7 (BGA-Mutationsverteilung) *Eine Zufallszahl der* BGA-Mutationsverteilung $\mathbf{B}(k, \rho)$ *wird aus einer gleichverteilten Zufallszahl* $u \sim \mathbf{U}([0, 1])$ *durch folgende Transformationsfunktion generiert:*

$$\boxed{g_{k,\rho}(u) = \rho \cdot 2^{-k \cdot u}} \tag{3.92}$$

Die BGA-Mutationsverteilung hat die Eigenschaft, daß sie wesentlich häufiger kleinere als größere Werte im Intervall $[\rho \cdot 2^{-k}, \rho]$ generiert. ρ wird in der Regel so gewählt, daß der Mutationsoperator den gesamten Bereich der jeweiligen Objektvariablen überdecken kann. Für jede Objektvariable wird in Abhängigkeit ihres Definitionsbereiches ein eigenes ρ festgelegt. Sind die Grenzen der i-ten Objektvariablen durch $x_i^{\min} \leq x_i \leq x_i^{\max}$ gegeben, wird $\rho_i = x_i^{\max} \Leftrightarrow x_i^{\min}$ gesetzt. Da der Parameter k für ein festes ρ die minimale Schrittweite vorgibt, muß er so gewählt werden, daß die geforderte Genauigkeit erfüllt wird. k wird der *Präzisionsparameter* und ρ der *Grundintervallparameter* der BGA-Mutationsverteilung genannt. Da der Grundintervallparameter die Schrittweite der Mutation beeinflußt, wird er auch als Schrittweitenparameter bezeichnet. Die Voreinstellung des Präzisionsparameters ist $k = 32$. $\mathbf{B}(k, 1)$ wird als *standardisierte* BGA-Mutationsverteilung bezeichnet und durch $\mathbf{B}(k)$ abgekürzt. Eine detailierte Untersuchung der BGA-Mutationsverteilung, in der die Bedeutungen des Präzisionsparameters und Schrittweitenparameters herausgestellt werden, ist in Abschnitt 4.2 zu finden.

Eine Mutation besteht aus der Generierung der Schrittweite und des Vorzeichens. Da die oben eingeführte BGA-Mutationsverteilung nur positive Schritte erzeugt, ist ein weiterer Aufruf des Zufallszahlgenerators notwendig. Der zweimalige Aufruf läßt sich vermeiden, indem die Zufallsvariablen β und $\mathbf{B}(k, \rho)$ zu einer Zufallszahl $\hat{\mathbf{B}}(k, \rho)$ zusammengesetzt werden. Die Verteilung dieser Zufallszahl wird im folgenden als *symmetrische BGA-Mutationsverteilung* bezeichnet.

Definition 8 (Symmetrische BGA-Mutationsverteilung) *Eine Zufallszahl der symmetrischen BGA-Mutationsverteilung* $\hat{\mathbf{B}}(k, \rho)$ *wird aus einer gleichverteilten Zufallszahl* $u \sim \mathbf{U}([\Leftrightarrow 1, 1])$ *durch folgende Transformationsfunktion generiert:*

$$\boxed{\hat{g}_{k,\rho} = \rho \cdot sign(u) \cdot 2^{-k \cdot |u|}} \tag{3.93}$$

Im folgenden werden drei Mutationsoperatoren, die auf der oben vorgestellten BGA-Mutationsverteilung basieren, eingeführt. Sie unterscheiden sich lediglich in der Wahl der zu mutierenden Objektvariablen.

Standard-BGA-Mutation $\mathbb{M}_{k,\rho}^{\mathrm{BGA,STD}}$

Der Standard-Mutationsoperator des BGA basiert auf der in Gleichung 3.92 eingeführten stetigen BGA-Mutationsverteilung. Eine Objektvariable x_i wird durch eine Mutation auf folgende Weise modifiziert:

$$\boxed{x_i' = x_i + \hat{\mathbf{B}}(k, \rho_i)} \tag{3.94}$$

Es wird genau *eine* zufällig ausgewählte Objektvariable mutiert, so daß der Operator nur zwei Aufrufe des Zufallszahlgenerators erfordert. Der resultierende Operator erzeugt neue Punkte ausschließlich entlang der Koordinatenachsen.

Mehrachsen-BGA-Mutation $\mathbb{M}_{k,\rho,\nu}^{\mathrm{BGA,MA}}$

Die Mehrachsen-BGA-Mutation ist eine spezielle Erweiterung der Standard-BGA-Mutation für Zielfunktionen, in denen benachbarte Variablen eine stärkere Korrelation aufweisen als weiter entfernte. Zunächst wird wie bei der Standard-BGA-Mutation eine Objektvariable x_i zufällig ausgewählt und mutiert. Zusätzlich werden die benachbarten Objektvariablen von x_i in Abhängigkeit von ihrer Position zu x_i modifiziert. Je größer die Indexentfernung zur zufällig bestimmten Variablen ist, umso geringer ist die Veränderung.

$$\boxed{\begin{array}{rcl} x_i' & = & x_i + \rho_i \cdot \hat{\delta} \\ x_{i-j}' & = & x_{i-j} + \rho_{i-j} \cdot \hat{\delta} \cdot \nu^{j-1} \\ x_{i+j}' & = & x_{i+j} + \rho_{i+j} \cdot \hat{\delta} \cdot \nu^{j-1} \end{array}} \quad j = 1, \ldots, \nu \quad i \Leftrightarrow j > 0 \quad i + j \leq n \Leftrightarrow 1 \tag{3.95}$$

$\hat{\delta} \sim \hat{\mathbf{B}}(k)$ wird einmalig bestimmt und für alle Objektvariablen verwendet. Für $\nu = 0$ entspricht diese Mutationsvariante der Standard-BGA-Mutation, in der genau eine Objektvariable mutiert wird. Diese spezielle Variante der BGA-Mutation kann nur dann sinnvoll eingesetzt werden, wenn bekannt ist, daß die Zielfunktion die entsprechende Eigenschaft aufweist.

BGA-Mutation mit Hauptachsentransformation $\mathbb{M}_{k,\rho}^{\mathrm{BGA,PCA}}$

Die oben eingeführte Standard-BGA-Mutation entspricht einem achsenorierentieren Suchverfahren, da in einer Mutation nur eine Objektvariable modifiziert wird. Der Bereich, in dem neue Punkte generiert werden, ist damit stark eingeschränkt. Weist die Fitnesslandschaft eine Orientierung auf, die nicht parallel zu den Hauptachsen verläuft, ist diese Suche äußerst ineffizient. Dieser Problematik kann durch die in Abschnitt 3.5.4 vorgestellten BGA-Line-Rekombination, die einer BGA-Mutation entlang der durch die beiden Elternindividuen definierten Richtung entspricht, begegnet werden.

Die Verallgemeinerung dieses Operators, in der die Mutationsrichtung nicht nur durch die beiden Eltern, sondern durch mehrere Individuen bestimmt wird, kann durch eine Transformation des Koordinatensystems erreicht werden. Die Population enthält aufgrund der Selektion Information über die Orientierung der Fitnesslandschaft. Aus dieser Information kann ein neues Koordinatensystem, das einem Modell der Fitnesslandschaft entspricht, generiert werden. Das Basissystem wird durch eine orthogonale Reihenentwicklung erzeugt, die den mittleren quadratischen Abstand aller Individuen von allen anderen maximiert. Als Abstandsmaß dient das Quadrat des euklidischen Abstands. Nachdem das

neue Basissystem einmalig für die gesamte Population berechnet worden ist, wird jedes Individuum analog zu dem oben definierten Mutationsoperator bezüglich des neuen Basissystems modifiziert. Diese Transformation wird als *Hauptachsentransformation* (engl.: *principal components analysis*) oder auch als *Karhunen-Loève-Transformation* bezeichnet. Das Basissystem wird durch die Eigenvektoren der durch ausgewählte Individuen definierten Kovarianzmatrix aufgespannt. Wenn $\mathbf{X}$ die $m \times n$-Matrix ist, die sich aus m ausgewählten Individuen $\vec{x}_k$ zusammensetzt, dann sind die Einträge der Kovarianzmatrix $\mathbf{C}$ durch die empirischen Kovarianzen von $\mathbf{X}$ definiert.

$$c_{ij} = \frac{1}{m \Leftrightarrow 1} \sum_{k=1}^{m} = (x_{ik} \Leftrightarrow \bar{\vec{x}}_i) \cdot (x_{jk} \Leftrightarrow \bar{\vec{x}}_j) \tag{3.96}$$

Die Eigenvektoren der Kovarianzmatrix $\mathbf{C}$ bilden das Basissystem der Transformation $\mathbf{V} = (v_{ij})$. Da der zum größten Eigenwert gehörige Eigenvektor die Richtung mit der größten Varianz definiert, kann diese Information genutzt werden, um bestimmte Richtungen zu bevorzugen. Um einen Schritt in Richtung des i-ten Eigenvektors zu machen, werden alle Objektvariablen auf folgende Weise modifiziert:

$$\boxed{x'_j = x_j + \hat{\mathbf{B}}(k, \rho) \cdot v_{ij}} \tag{3.97}$$

Die Lösung des Eigenwertproblems hat die Komplexität $\mathcal{O}(n^3)$. Um eine aussagekräftige Schätzung der Kovarianzmatrix zu gewährleisten, wird empfohlen, die Anzahl der ausgewählten Individuen m auf $\approx c \cdot n^2$ zu setzen. Die m Individuen, aus deren Lage das neue Basissystem berechnet wird, werden aus der Population ausgewählt. Als Berechnungsgrundlage für die Kovarianzmatrix dienen die durch die Schnittselektion mit Remontierungsrate ϑ selektierten Individuen. Aus der aktuellen Population werden also die besten $\vartheta \cdot 100$ Prozent in die Menge $\mathbf{X}$ aufgenommen. Jede Generation steuert folglich $\lambda \cdot \vartheta$ Individuen zu der Menge $\mathbf{X}$ bei. Ist die Anzahl der selektierten Individuen zu klein ($\lambda \cdot \vartheta < m$) und können infolgedessen nicht alle m Einträge von $\mathbf{X}$ aus einer Generation gewonnen werden, wird ein Speicher eingeführt, der m ausgewählte Individuen der letzten $\frac{m}{\lambda \cdot \vartheta}$ Generationen beinhaltet. Um die aufwendigen Berechnungen des Basissystem zu reduzieren, wird es nicht in jeder, sondern nur in jeder n-ten Generation neu berechnet, wobei n die Dimension des Problems ist.

Die BGA-PCA-Mutation besteht aus zwei Teilen. Der erste Teil, die Berechnung des neuen Basissystems, wird alle n Generationen einmalig für die gesamte Population durchgeführt. Die Berechnung des neuen Basissystem ist in Algorithmus 3.3 beschrieben.

Algorithmus 3.3 *BGA-PCA-ComputeBase*

BGA-PCA-ComputeBase($\mathbf{X}, frq$)
1 $\mathbf{X} \leftarrow$ m Individuen
2 $\mathbf{C} \leftarrow$ CovarianceMatrix ($\mathbf{X}$)
3 $\mathbf{V} \leftarrow$ EigenVectors ($\mathbf{C}$)
4 $\vec{d} \leftarrow$ EigenValues ($\mathbf{C}$)
5 **return** $\mathbf{V}, \vec{d}$

Der zweite Teil entspricht im wesentlichen der Standard-BGA-Mutation. Die Mutation erfolgt entlang einer Achse des neuen Basissystems. Zusätzlich können hier Informationen

über die Wichtigkeit der einzelnen Koordinatenachsen, die das resultierende Eigensystem beinhaltet, genutzt werden, um bestimmte Achsen zu bevorzugen. So kann eine Achse mit hohem Eigenwert häufiger ausgewählt oder stärker modifiziert werden. Der genaue Ablauf der BGA-PCA-Mutation ist in Algorithmus 3.4 dargestellt.

Algorithmus 3.4 *BGA-PCA-Mutation*

$\mathbb{M}_{k,\rho}^{\mathrm{BGA,PCA}}(\vec{x}, \mathbf{V}, \vec{d})$

```
1  i ← ChooseAxis (n, d)                    // Wahl der Achse
2  w ← WeightStep (d, i)                    // Gewichtung des Schritts
3  δ ← B̂(k);
4  for j ← 1 to n
5      do x_j ← x_j + w · v_ij · B̂(k, ρ)
6  return x
```

Die Algorithmen 3.5 und 3.6 beschreiben, wie einzelne Achsen aufgrund ihrer Eigenwerte $\vec{d}$ bevorzugt werden. Der Algorithmus ChooseAxis zur unterschiedlichen Gewichtung für die Auswahl von Achsen ähnelt dem Roulette-Wheel-Verfahren der Genetischen Algorithmen. Aus Tabelle 3.1 lassen sich die Wahrscheinlichkeiten für die Wahl der einzelnen Achsen ablesen.

Algorithmus 3.5 *ChooseAxis*

ChooseAxis$(n, \vec{d})$

```
1  max ← sum ← 0
2  for i ← 1 to n
3      do max ← max + (i + 1)^-1
4  i ← 1
5  while sum < U([0, 1]) · max
6      do sum ← sum + (i + 1)^-1
7          i ← i + 1
8  return AxisIndex(d, i ⇔1)
```

Die Prozedur AxisIndex liefert den Index des Eigenvektors mit dem it-höchsten Eigenwert. Der Algorithmus WeightStep gewichtet die Schrittlänge einer Achse aufgrund des Verhältnisses ihres Eigenwertes zum höchsten Eigenwert.

Algorithmus 3.6 *WeightStep*

WeightStep$(\vec{d}, i)$

```
1  w ← (d_0 / d_i)^-0.5
2  return w
```

In Abbildung 3.5 ist die Funktionsweise der BGA-Mutation mit Hauptachsentransformation anhand der zweidimensionalen Funktion von Rosenbrock dargestellt. Die aus $\lambda = 512$ Individuen bestehende Anfangspopulation wurde normalverteilt um den Punkt

i	$(i+1)^{-1}$	$\frac{\sum_{j=1}^{i}(j+1)^{-1}}{\sum_{i=1}^{n}(j+1)^{-1}}$	P(X=i)
1	0.5	0.248	0.247
2	0.333	0.413	0.165
3	0.25	0.536	0.124
4	0.2	0.635	0.099
5	0.167	0.718	0.083
6	0.143	0.789	0.071
7	0.125	0.850	0.062
8	0.111	0.905	0.055
9	0.1	0.955	0.050
10	0.091	1.000	0.045

Tabelle 3.1: Durch Algorithmus 3.5 generierte Wahrscheinlichkeiten für die Auswahl einer Achse aufgrund ihres Eigenwertes für $n = 10$. $P(X = i)$ bezeichnet die Wahrscheinlichkeit, daß die Achse mit dem it-höchsten Eigenwert ausgewählt wird.

$(\Leftrightarrow 2, 2)$ gestreut. Mit einer Remontierungsrate $\vartheta = 0.5$ ergeben sich $\mu = 256$ Eltern. Das Optimum dieser Funktion liegt im Punkt $(1, 1)$ und ist durch ein Kreuz gekennzeichnet. Die sechs Bilder zeigen die Lage der Nachkommen und die Achsen des transformierten Basissystems in verschiedenen Generationen. Hier ist deutlich zu erkennen, wie sich das Basissystem im Laufe der Suche aufgrund der Lage der Population an die aktuelle Situation anpaßt. In Abbildung 3.6 sind nochmals die Achsen des Basissystems in drei verschiedenen Generationen abgebildet. Die fett eingezeichneten Achsen kennzeichnen die jeweilige Achse mit dem höchsten Eigenwert. In der rechten Abbildung ist die Entwicklung der Eigenwerte dargestellt. Sie zeigt das Verhältnis des kleineren zum größeren Eigenwert. Mit der Anpassung des Basissystems an das schmale Tal steigt auch das Verhältnis der beiden Eigenwerte. Dieses Indiz für die Wichtigkeit der Basisachse mit dem größeren Eigenwert wird in der Bestimmung der zu mutierenden Achse sowie deren Schrittweite verwendet. Sobald sich die Suche zwischen Generationen 30 und 40 dem Ursprung nähert, gleicht sich das Verhältnis aus. Es steigt erst wieder an, wenn sich das Basissystem an die neue Orientierung angepaßt hat.

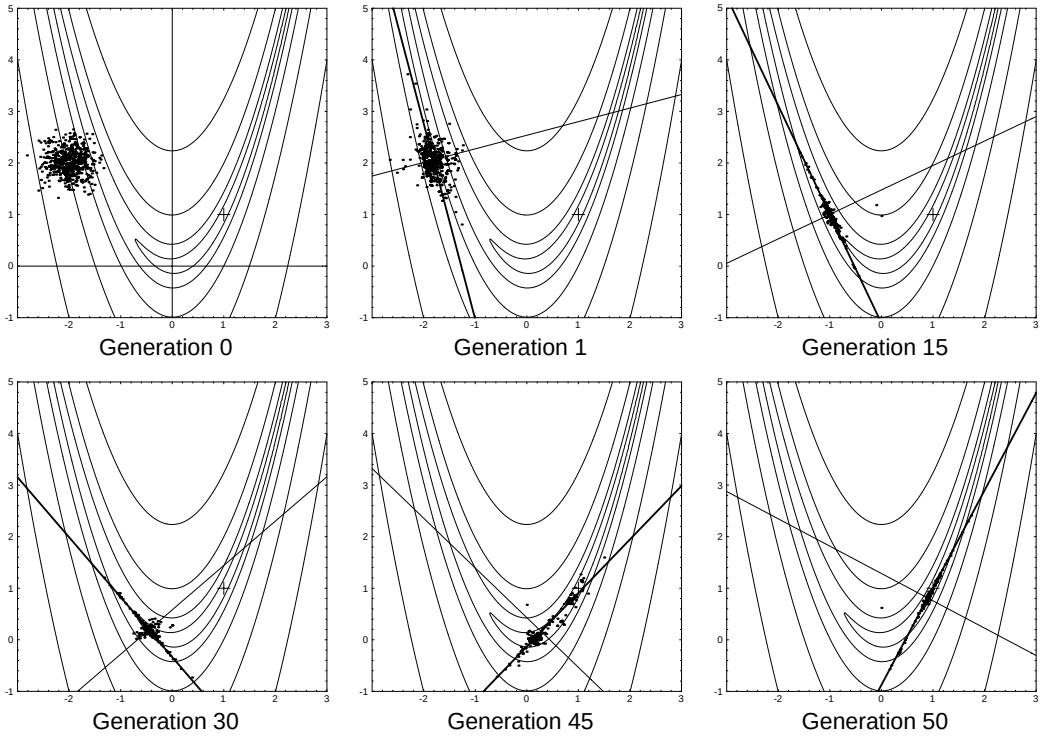

Abbildung 3.5: BGA-Mutation mit Hauptachsentransformation angewandt auf Rosenbrocks Funktion. Die Anfangspopulation wurde um den Punkt $(-2, 2)$ gestreut. Ab Generation 1 sind die Koordinatenachsen des neuen Basissystems mit dem Schwerpunkt der Elterngeneration als Ursprung eingezeichnet. Die fett eingezeichnete Achse gibt den Eigenvektor mit höherem Eigenwert an.

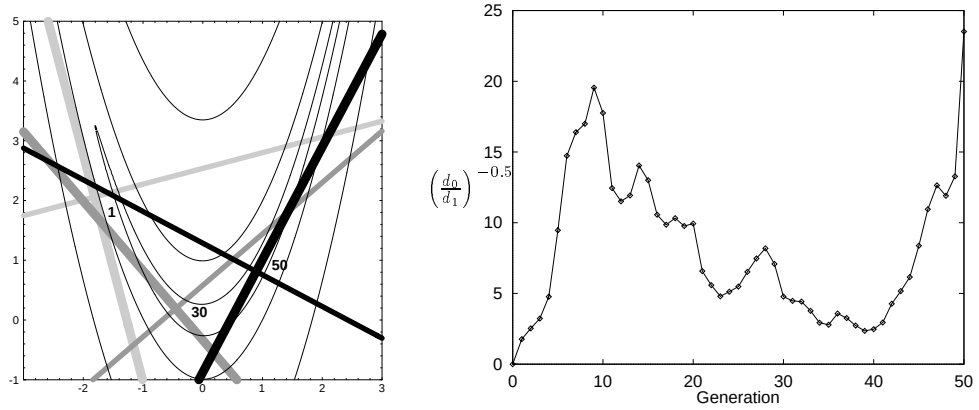

Abbildung 3.6: Die linke Abbildung zeigt die Basissysteme aus Abbildung 3.5 in den Generationen 1, 30 und 50. Die rechte Abbildung zeigt die Entwicklung des Verhältnisses zwischen den Eigenwerten $\left(\frac{d_0}{d_1}\right)^{-0.5}$.

3.5.6 Spezifikation von BGA-Objekten

Ein BGA besteht aus einer Kombination der in diesem Abschnitt vorgestellten Operatoren und zugehöriger Parameter. Im folgenden werden fünf Varianten des BGA, die sich in der Art und der Komplexität der Suche unterscheiden, mit Hilfe der in Abschnitt 3.1.4 eingeführten Syntax spezifiziert. Die Spezifikation der BGA-Instanzen erfolgt in drei Schritten. Zunächst wird die Population eines allgemeinen BGA beschrieben. Basierend auf dieser allgemeinen BGA-Population werden die speziellen BGA-Populationen spezifiziert. Zur Beschreibung der eigentlichen BGA-Instanzen werden diese speziellen Populationen herangezogen.

Spezifikation 8 $\mathsf{POP}^{\mathsf{BGA}}$ – *BGA-Population.*

Die allgemeine BGA-Population hat als Parameter die Populationsgröße, die Selektion und die beiden genetischen Operatoren.

$$\mathsf{POP}^{\mathsf{BGA}}\ (\lambda, \mathbb{S}, \mathbb{R}, \mathbb{M}) ::= \mathbf{POP}\left(\mathbf{IND}\left(\mathbb{C}^{\mathrm{BGA}}, \mathbb{R}, \mathbb{M}\right)^{\lambda}, \mathbb{S}, \mathbb{P}^{\mathrm{EA,S}}, \mathbb{I}^{\mathrm{EA,U}}\right)$$

Spezifikation 9 $\mathsf{BGA}^{\mathsf{BM}}$ – *BGA mit Mutation.*

Diese einfachste BGA-Variante verwendet als genetischen Operator ausschließlich die BGA-Mutation. Infolge des Verzichts auf sexuelle Rekombination erübrigt sich die Paarbildung.

$$\mathsf{POP}^{\mathsf{BGA,BM}}(\lambda, \mathbb{S}, k, \rho) ::= \mathsf{POP}^{\mathsf{BGA}}\left(\lambda, \mathbb{S}, \mathbb{M}_{k,\rho}^{\mathrm{BGA,STD}}\right)$$

$$\mathsf{BGA}^{\mathsf{BM}}(\lambda, \vartheta, k, \rho, \mathbb{T}) ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{BGA,BM}}(\lambda, \mathbb{S}_{\vartheta}^{\mathrm{BGA}}, k, \rho), \mathbb{T}\right)$$

Spezifikation 10 $\mathsf{BGA}^{\mathsf{PCA}}$ – *BGA mit Hauptachsentransformation.*

Diese BGA-Variante verwendet ebenfalls ausschließlich die BGA-Mutation. Die Mutationen erfolgen jedoch nicht entlang der Achsen des Basiskoordinatensystems, sondern entlang eines transformierten Koordinatensystems.

$$\mathsf{POP}^{\mathsf{BGA,PCA}}(\lambda, \mathbb{S}, k, \rho) ::= \mathsf{POP}^{\mathsf{BGA}}\left(\lambda, \mathbb{S}, \mathbb{M}_{k,\rho}^{\mathrm{BGA,PCA}}\right)$$

$$\mathsf{BGA}^{\mathsf{PCA}}(\lambda, \vartheta, k, \rho, \mathbb{T}) ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{BGA,PCA}}(\lambda, \mathbb{S}_{\vartheta}^{\mathrm{BGA}}, k, \rho), \mathbb{T}\right)$$

Spezifikation 11 $\mathsf{BGA}^{\mathsf{BLR}}$ – *BGA mit Line-Rekombination und Mutation.*

Diese BGA-Variante führt ebenso wie die zuvor beschriebene BGA-Variante eine gerichtete Suche aus. Sie unterscheidet sich von dem BGA mit Hauptachsentransformation durch einen geringeren Berechnungs- und Implementierungsaufwand.

$$\mathsf{POP}^{\mathsf{BGA,BLR}}(\lambda, \mathbb{S}, k, \rho) ::= \mathsf{POP}^{\mathsf{BGA}}\left(\lambda, \mathbb{S}, \mathbb{R}_{k,\rho}^{\mathrm{BGA,L}}, \mathbb{M}_{k,\rho}^{\mathrm{BGA,STD}}\right)$$

$$\mathsf{BGA}^{\mathsf{BLR}}(\lambda, \vartheta, k, \rho, \mathbb{T}) ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{BGA,FR}}(\lambda, \mathbb{S}_{\vartheta}^{\mathrm{BGA}}, k, \rho), \mathbb{T}\right)$$

Spezifikation 12 $\mathsf{BGA}^{\mathsf{FR}}$ – *BGA mit Fuzzy-Rekombination und Mutation.*
Der BGA mit Fuzzy-Rekombination führt eine Breitensuche aus.

$$\mathsf{POP}^{\mathsf{BGA,FR}}(\lambda, \mathbb{S}, k, \rho) ::= \mathsf{POP}^{\mathsf{BGA}}\left(\lambda, \mathbb{S}, \mathbb{R}_{1.0}^{\mathrm{BGA,F}}, \mathbb{M}_{k,\rho}^{\mathrm{BGA,STD}}\right)$$

$$\mathsf{BGA}^{\mathsf{FR}}(\lambda, \vartheta, k, \rho, \mathbb{T}) ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{BGA,FR}}(\lambda, \mathbb{S}_{\vartheta}^{\mathrm{BGA}}, k, \rho), \mathbb{T}\right)$$

Die Spezifikationen der verschiedenen BGA-Klassen haben als Parameter die Populationsgröße λ, den Schrittweitenparameter ρ und den Präzisionsparameter k der BGA-Mutationsverteilung. Die Standardbelegungen dieser Parameter sind $\lambda = 64$, $\rho = x_{\max} \Leftrightarrow x_{\min}$ und $k = 32$. Die Terminierung ist in der Regel von dem zu lösenden Problem abhängig.

3.6 Erweiterungen

Die bisher vorgestellten Evolutionären Algorithmen basieren auf der rein genetischen Suche in einer Population. In der Literatur findet man eine Reihe von Erweiterungen oder Modifikationen dieser Basisalgorithmen. Zu den wesentlichsten Erweiterungen zählen die Einführung verteilter Evolutionärer Algorithmen sowie die Integration lokaler Suchverfahren.

3.6.1 Verteilte Evolutionäre Algorithmen

Verteilte Evolutionäre Algorithmen arbeiten im Gegensatz zu den *einfachen* Evolutionären Algorithmen mit mehreren, zunächst voneinander unabhängigen, Populationen. Jede dieser Populationen entwickelt sich für einen festgelegten *Isolationszeitraum* wie in Definition 1 festgelegt. Nach diesem Zeitraum findet eine sogenannte Migration, ein Austausch von Individuen, zwischen benachbarten Populationen statt.

Eine Migration besteht in der Übertragung eines Individuums oder der Kopie eines Individuums von einer Population in eine andere. Im ersten Fall wird das emigrierende Individuum aus der Population, aus der es stammt, herausgenommen. Im zweiten Fall verbleibt es in der Population. Die Emigranten, die in den Zielpopulationen als Immigranten bezeichnet werden, werden gemeinsam mit den Nachkommen der Zielpopulation der nachfolgenden Selektionsprozedur des Evolutionären Algorithmus unterzogen und dadurch in die Zielpopulation integriert.

Definition 9 (Verteilter Evolutionärer Algorithmus) *Ein* verteilter Evolutionärer Algorithmus (DEA) *ist definiert als*

$$DEA = (EA^{\iota}, T, \mathbb{K}) \tag{3.98}$$

wobei

EA:	*Evolutionärer Algorithmus*
ι:	*Anzahl der Inseln*
T:	*Topologie der Inseln*
$\mathbb{K}$:	*Migrationsschema*

bezeichnen.

Diese Art der Parallelität, in der mehrere eigenständige Populationen nebeneinander existieren, wird als *Island-Model* oder *Migrationsmodell* bezeichnet. Die Struktur eines verteilten EA ist in Abbildung 3.7 dargestellt. Die Anzahl der Inseln wird durch die Rechnerumgebung beinflußt. Eine untere Schranke ist durch die Anzahl der Prozessoren vorgegeben. Die Topologie, in der die Populationen angeordnet sind, kann beliebige Formen annehmen. In der Regel findet man regelmäßige Anordnungen wie Torus, Gitter oder Ring vor. Der Diameter der Topologie entscheidet ebenso wie die Isolationszeit, wie eng die verschiedenen Populationen gekoppelt sind. Unter dem Migrationsschema versteht man die Art, in der eine Migration erfolgt. Es ist zu entscheiden, wann, wieviele und welche Individuen emigrieren und wie neue Individuen in die Population aufgenommen werden. Alle diese Parameter sind eng miteinander verknüpft und für den Erfolg der Parallelisierung verantwortlich.

Um einen frühzeitigen Varianzverlust in der Gesamtpopulation zu verhindern, sollte nicht immer das beste Individuum einer Population emigrieren, sondern eines der besten. Die Auswahl des Emigranten kann durch ähnliche Konzepte, wie sie in der Selektion verwendet werden, vorgenommen werden. In Analogie zur Schnittselektion wird der Emigrant zufällig aus den besten ϑ % Individuen der Population ausgewählt.

Da die beteiligten Populationen gleichzeitig simuliert werden können, eignet sich dieser Ansatz besonders für den Einsatz von Parallelrechnern. Die Generationenschleife jeder Population wird in diesem Fall von einem eigenen Prozessor ausgeführt. Durch die Verwendung von Puffern, in die neue Immigranten vor ihrer Integration abgelegt werden, kann auf eine Synchronisation der Populationen verzichtet werden, so daß in den Simulationen Wartezeiten zwischen den Prozessoren vermieden werden.

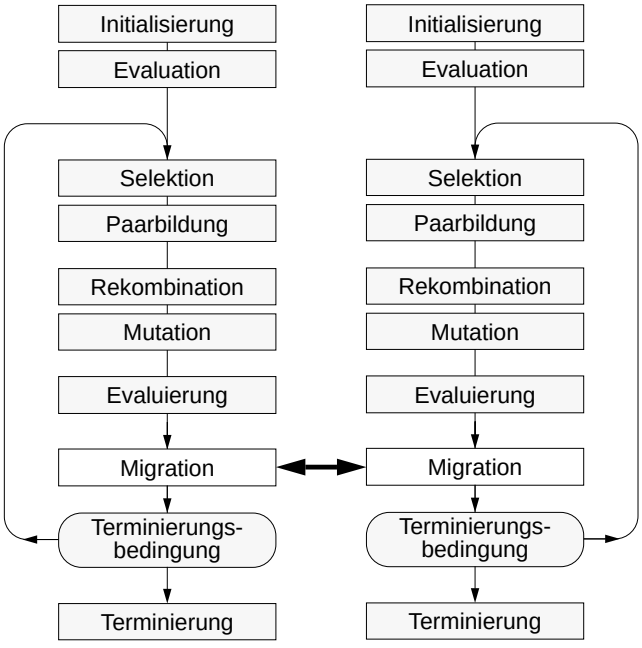

Abbildung 3.7: Struktur eines verteilten Evolutionären Algorithmus mit Migrationsblock

Diese parallele Variante von EA eignet sich besonders für die Optimierung separierbarer Probleme, die sich dadurch auszeichnen, daß sich ihr jeweiliges Optimum aus Teillösun-

gen suboptimaler Lösungen zusammensetzt. Die unterschiedlichen Teillösungen, die die verschiedenen Populationen während der Isolation finden, können so zu neuen Lösungen zusammengestellt werden.

Eine andere Art der Parallelisierung, die nicht wie das Island-Model auf der Populationsebene, sondern auf der Individuenebene aufsetzt, ist das sogenannte *Diffusionsmodell* oder *Nachbarschaftsmodell*, das in dem Algorithmus ASPARAGOS zur Lösung des *Travelling Salesman Problems* verwendet wurde ([GS90]). Hier sucht sich jedes Individuum seine Paarungspartner nach einem Verfahren, das eine festgelegte Nachbarschaftsstruktur berücksichtigt. Dieser Ansatz erfordert neben einer Parallelisierung des EA auch neue Selektionsmechanismen, die auf der lokalen Information der Nachbarschaft basieren. Rudolph wandte das Nachbarschaftsmodell erstmals im Rahmen der kontinuierlichen Parameteroptimierung an ([Rud92]). In [Rud93] stellt Rudolph ein auf der Evolutionsstrategie basierendes Nachbarschaftsmodell mit Simulated Annealing-Komponente vor und wendet es auf eine Menge bekannter Testfunktionen der kontinuierlichen Parameteroptimierung an. Dazu setzt er 16384 Individuen in einem 64×256-Torus ein. Sprave untersucht in [Spr94] für ein auf der Evolutionsstrategie basierendes Nachbarschaftsmodell mit Ringtopologie die Auswirkung der Nachbarschaftsgröße.

3.6.2 Hybride Evolutionäre Algorithmen

Unter einem *hybriden* Evolutionären Algorithmus versteht man einen EA, der neben genetischen Operatoren auch lokale Suchverfahren (engl.: *local hillclimber*) verwendet. In vielen Anwendungen aus der kombinatorischen Optimierung, die die Building-Block-Eigenschaft aufweisen, hat es sich als zweckmäßig erwiesen, die Individuen durch ein lokales Suchverfahren in das nächstgelegene lokale Optimum zu bringen, und dann den Raum der lokalen Optima durch den EA abzusuchen.

Für die kontinuierliche Parameteroptimierung können die in Kapitel 2 vorgestellten Optimierungsverfahren als lokale Suchverfahren eingesetzt werden. Das lokale Suchverfahren wird dazu, wie in Abbildung 3.8 dargestellt, in der Generationenschleife nach der Evaluierung aufgerufen.

Die Einbindung lokaler Suchverfahren in einen Evolutionären Algorithmus ist nicht unproblematisch. Neben den eigenen Parametern des Hillclimbers sind weitere externe Parameter notwendig, die dessen Aufruf steuern. So muß festgelegt werden, wann der Hillclimber aufgerufen wird und mit welchen Individuen er initialisiert wird. Der Aufruf kann beispielsweise in festgelegten Intervallen oder unter bestimmten Bedingungen erfolgen. Oft wird der Hillclimber nur auf das beste Individuum aus der Population angewendet. Denkbar ist jedoch auch, daß alle Individuen durch eine lokale Suche verbessert werden.

Im Extremfall, in dem die Population aus nur einem Individuum besteht, entspricht der hybride Evolutionäre Algorithmus einem sogenannten *iterated Hillclimber*. Iterated Hillclimber wenden ein lokales Optimierungsverfahren wiederholt auf das Resultat der vorhergehenden Iteration an. In [PTVF92] wird empfohlen, lokale Suchverfahren wiederholt zu starten, wobei das Resultat einer Iteration als Startpunkt der nächsten Iteration verwendet wird. Der Neustart eines Hillclimbers initialisiert die internen Parameter (internes Modell) neu und gibt dem Verfahren dadurch neues Potential. Der Nutzen der Einbeziehung lokaler Suchverfahren hängt stark von dem jeweiligen Problem ab.

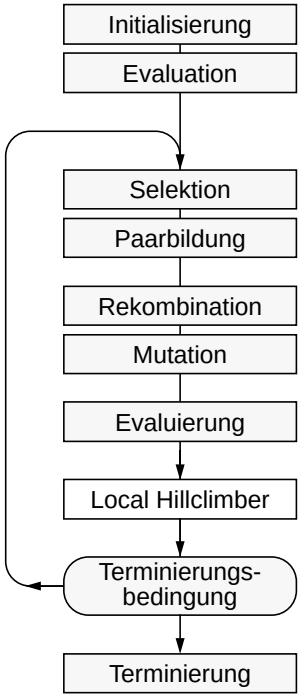

Abbildung 3.8: Struktur eines hybriden Evolutionären Algorithmus mit lokalem Optimierer

Kapitel 4

Untersuchung der BGA-Mutationsverteilung

4.1 Definitionen und Notationen

In diesem Abschnitt werden Definitionen und Notationen aus der Wahrscheinlichkeitstheorie, die für die theoretischen Betrachtungen und die Auswertung von Simulationsergebnissen benötigt werden, eingeführt.

Zufallsvariablen, Verteilungen und Dichtefunktionen

Eine Zufallsgröße $X : \Omega \to \mathbb{R}$, die über einem Wahrscheinlichkeitsraum $(\Omega, \mathcal{A}, P)$, mit Ω als der Menge der Elementarereignisse, $\mathcal{A}$ als der Sigma-Algebra der Ereignisse und P als Wahrscheinlichkeitsmaß definiert ist, wird durch ihre *Verteilungsfunktion* charakterisiert:

$$\begin{aligned} F_X \quad &: \quad \mathbb{R} \to [0,1] \\ &\quad\; x \mapsto F_X(x) = P(X \leq x). \end{aligned} \tag{4.1}$$

Sei $f_X : \mathbb{R} \to \mathbb{R}$ die *Dichtefunktion* einer stetigen Zufallsvariablen X, so ist

$$F_X(x) = \int_{-\infty}^{x} f_X(z)dz \qquad \forall x \in \mathbb{R}. \tag{4.2}$$

Wichtige Parameter einer Zufallsvariablen sind ihre Momente wie Erwartungswert, Varianz, Schiefe und Wölbung.

Definition 10 (Erwartungswert) *Der* Erwartungswert *einer eindimensionalen stetigen Zufallsvariablen X mit Dichtefunktion $f_X(x)$ ist definiert als*

$$E(X) = \int_{-\infty}^{\infty} x \cdot f_X(x)\, dx \tag{4.3}$$

Der Erwartungswert wird auch durch μ symbolisiert. Als Schätzwert für μ dient in der Folge der *arithmetische Mittelwert* $\bar{x}$, der aus einer Folge von Beobachtungen $X_1, \ldots, X_n$ in einer Zufallsstichprobe mit Stichprobenumfang n ermittelt wird.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{4.4}$$

Definition 11 (Momente einer Zufallsgröße) *Sei X eine Zufallsvariable mit Erwartungswert $E(X)$, dann heißt*

$$\mu_k = E((X \Leftrightarrow E(X))^k) \qquad (k \in \mathbb{N}) \tag{4.5}$$

das k-te zentrale Moment der Verteilung von X, *unter der Voraussetzung der Existenz des Moments.*

Das zweite zentrale Moment wird als *Varianz* bezeichnet. Die Varianz, die als $Var(X)$ oder σ^2 geschrieben wird, beschreibt die Streuung der Zufallsgröße X.

$$Var(X) = E((X \Leftrightarrow E(X))^2) \tag{4.6}$$

σ wird *Standardabweichung* der Zufallsgröße X genannt. Als Schätzwert für σ wird hier die *empirische Standardabweichung* s verwendet.

$$s = \sqrt{\frac{1}{n \Leftrightarrow 1} \sum_{i=1}^{n} (X_i \Leftrightarrow \bar{x})^2} \tag{4.7}$$

Das vierte zentrale Moment μ_4 der Zufallsgröße X dient zur Berechnung der Wölbung.

$$Kurt(X) = \frac{E((X \Leftrightarrow E(x))^4)}{(Var(X))^2} \tag{4.8}$$

Gütemaße für das arithmetische Mittel

Schätzwerte zu unbekannten Verteilungsparametern entsprechen Schätzfunktionen, die selbst zufällige Variablen mit geeigneten Wahrscheinlichkeitsverteilungen sind. Um den Fehler zwischen Schätzwert und tatsächlichem Wert beurteilen zu können, werden zwei Gütemaße für die Schätzwerte eingeführt.

Definition 12 (Standardfehler des arithmetischen Mittels) *Der* Standardfehler des arithmetischen Mittels $\sigma_{\bar{x}}$ *gibt an, wie stark der Stichprobenmittelwert* $\bar{X} = \frac{1}{n}(X_1 + \ldots + X_n)$ *von n unabhängigen Zufallsvariablen* $X_1, \ldots, X_n$ *derselben Verteilung um den Mittelwert* μ *der Zufallsvariablen X schwankt.*

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}} \tag{4.9}$$

wobei σ^2 *die Varianz der X-Grundgesamtheit ist.*

Als Schätzwert für $\sigma_{\bar{x}}$ dient

$$s_{\bar{x}} = \frac{s}{\sqrt{n}} = \sqrt{\frac{\sum (X_i \Leftrightarrow \bar{x})^2}{n(n \Leftrightarrow 1)}} = \sqrt{\frac{\sum X_i^2 \Leftrightarrow (\sum X_i)^2 / n}{n(n \Leftrightarrow 1)}} \tag{4.10}$$

mit s als empirischer Standardabweichung.

Tabelle 4.1 zeigt, welchen Anteil der Werte bzw. der Stichprobenmittelwerte einer Verteilung ein bestimmter Bereich enthält. Da die Abweichung mit steigendem Stichprobenumfang abnimmt, kann aufgrund des Standardfehlers der Stichprobenumfang inkrementell erhöht werden, um eine gewünschte Güte des Mittelwertes zu erzielen. Diese Technik eignet sich besonders für die statistische Auswertung von Experimenten, bei der die Anzahl Stichproben so vom Standardfehler abhängig gemacht werden kann.

Bereich	bel. Verteilung	Normalverteilung
$\bar{x} \pm 2s$	$> 89\%$	$\approx 95\%$
$\bar{x} \pm 2s_{\bar{x}}$	$< 95\%$	$\approx 95\%$

Tabelle 4.1: Anteil der Werte bzw. der Stichprobenmittelwerte im angegebenen Bereich ([Sac92], S. 160).

Notation

Die Notation und Symbole der oben vorgestellten statistischen Größen, wie sie im folgenden verwendet werden, sind in Tabelle 4.2 zusammengefaßt.

Bezeichnung	Statistiksymbol	Schätzwertsymbol
Mittelwert	$E(X)$, μ	$\bar{x}$
Varianz	$Var(X)$, σ^2	s^2
Standardabweichung	$\sqrt{Var(X)}$, σ	s
Wölbung	$Kurt(X)$, ω	a_4
Standardfehler	$\sigma_{\bar{x}}$	$s_{\bar{x}}$

Tabelle 4.2: Symbole für statistische Größen und ihre Schätzwerte.

4.2 BGA-Mutationsverteilung

Die Mutationsoperatoren des BGA basieren auf der in Abschnitt 3.5.5 eingeführten BGA-Mutationsverteilung $\mathbf{B}(k, \rho)$. In diesem Abschnitt werden die Eigenschaften dieser Mutationsverteilung,die sich wesentlich von Mutationsverteilungen anderer Evolutionärer Algorithmen unterscheidet, herausgearbeitet.

Die einfache BGA-Mutationsverteilung $\mathbf{B}(k, \rho)$ mit Präzisionsparameter $k > 0$ und Grundintervallparameter $\rho > 0$ generiert ausschließlich positive Werte im Intervall $[\rho 2^{-k}, \rho]$. Da der Grundintervallparameter ρ die Schrittweite der Mutation beeinflußt, wird er im folgenden auch als Schrittweitenparameter bezeichnet. Durch die Kombination der Größen k und ρ wird der Bereich und die Verteilung der Größe der Mutationsschritte festgelegt.

Die symmetrische BGA-Mutationsverteilung $\hat{\mathbf{B}}(k, \rho)$ setzt sich aus der BGA-Mutationsverteilung $\mathbf{B}(k, \rho)$ und einer Booleschen Zufallsvariablen, die das Vorzeichen bestimmt, zusammen. Soll eine Mutationsverteilung mit gleicher Wahrscheinlichkeit positive und negative Schritte erzeugen, so wird durch die Verwendung der symmetrischen Form die Hälfte der zu generierenden Zufallszahlen eingespart. Die Transformationsfunktionen, die aus einer gleichverteilten Zufallsvariablen eine Zufallsvariable der jeweiligen BGA-Verteilung erzeugen, sind in den Gleichungen 3.92 und 3.93 auf Seite 48 spezifiziert.

BGA-Mutationsverteilung $\mathbf{B}(k, \rho)$

Die BGA-Mutationsverteilung $\mathbf{B}(k, \rho)$, die ausschließlich positive Werte generiert, wird durch eine entsprechende Dichte- wie auch Verteilungsfunktion beschrieben. Eine Zufallsvariable der BGA-Mutationsverteilung wird im folgenden mit $X \sim \mathbf{B}(k, \rho)$ bezeichnet.

Satz 1 *Die* Dichtefunktion *der BGA-Mutationsverteilung* $\mathbf{B}(k,\rho)$ *ist*

$$\boxed{f_X(x) = \begin{cases} \frac{1}{k \ln 2} \cdot \frac{1}{x} & : \quad \rho 2^{-k} \leq x \leq \rho \\ 0 & : \quad sonst \end{cases}} \tag{4.11}$$

Beweis 1 *Sei $g(z)$ eine stetig differenzierbare Funktion, die eine eineindeutige Transformation einer Zufallsvariablen Z in eine Zufallsvariable X definiert. Dann ist die Dichtefunktion der Zufallsvariablen X, die aus der Zufallsvariablen Z mit der Dichtefunktion $f_Z(z)$ durch die Funktion $g(z)$ erzeugt wird ([Fis73], S. 60):*

$$f_X(x) = f_Z\left(h(x)\right) \cdot |h'(x)| \tag{4.12}$$

Laut Definition 7 der BGA-Mutationsverteilung wird eine Zufallszahl $X \sim \mathbf{B}(k,\rho)$ aus einer gleichverteilten Zufallszahl $z \sim \mathbf{U}([0,1])$ durch die Transformation $x = g(z) = \rho 2^{-kz}$ generiert. Die Umkehrfunktion von $g(z)$ und deren Ableitung ergeben sich zu

$$h(x) \quad = \quad \Leftrightarrow \frac{1}{k \ln 2} \ln\left(\frac{x}{\rho}\right) \tag{4.13}$$

$$h'(x) \quad = \quad \Leftrightarrow \frac{1}{k \ln 2} \cdot \frac{1}{x} \tag{4.14}$$

Da $g(z)$ im Intervall $[0,1)$ stetig differenzierbar ist, ergibt sich durch Einsetzen der Dichtefunktion der Gleichverteilung auf dem Intervall $[0,1]$ $f_Z(z) = 1$ und der Funktion $h'(x)$ aus Gleichung 4.14 in Gleichung (4.12) die Dichte der BGA-Mutationsverteilung. ∎

Die Verteilung der Zufallsvariablen $X \sim \mathbf{B}(k,\rho)$ liegt damit im Intervall $[\rho 2^{-k}, \rho]$ unter einem beidseitig gestutzten Hyperbelast.

Satz 2 *Die* Verteilungsfunktion *der BGA-Mutationsverteilung* $\mathbf{B}(k,\rho)$ *ist*

$$\boxed{F_X(x) = \begin{cases} 0 & : \quad x < \rho 2^{-k} \\ \frac{1}{k \ln 2} \ln\left(\frac{x}{\rho 2^{-k}}\right) & : \quad \rho 2^{-k} \leq x \leq \rho \\ 1 & : \quad x > \rho \end{cases}} \tag{4.15}$$

Beweis 2 *Die Verteilungsfunktion der Zufallvariablen $X \sim \mathbf{B}(k,\rho)$ ist definiert durch*

$$F_X(x) = \int_{-\infty}^{x} f_X(t)\, dt \tag{4.16}$$

mit der Dichtefunktion $f_X(t)$ der BGA-Mutationsverteilung $\mathbf{B}(k,\rho)$.

Setzt man die Dichtefunktion aus Gleichung 4.11 in die obige Gleichung ein, so ergibt sich die Verteilungsfunktion

Für $\rho 2^{-k} \leq x \leq \rho$*:*

$$\begin{aligned} F_X(x) &= \int_{\rho 2^{-k}}^{x} \frac{1}{k \ln 2} \cdot \frac{1}{t}\, dt \\ &= \frac{1}{k \ln 2} \cdot \int_{\rho 2^{-k}}^{x} \frac{1}{t}\, dt \\ &= \frac{1}{k \ln 2} \cdot [\ln t]_{\rho 2^{-k}}^{x} \\ &= \frac{1}{k \ln 2} \cdot \ln\left(\frac{x}{\rho 2^{-k}}\right) \end{aligned}$$

Für $x < \rho 2^{-k}$*: Da die Dichtefunktion für* $x < \rho 2^{-k}$ *Null ist, ist die Verteilungsfunktion in diesem Bereich*

$$F_X(x) = 0 \tag{4.17}$$

Für $x > \rho$*: Da die Dichtefunktion für* $x > \rho$ *Null ist, hat die Verteilungsfunktion hier den konstanten Wert*

$$F_X(x) = F_X(\rho) = 1 \tag{4.18}$$

■

Die Dichte- und Verteilungsfunktion der BGA-Mutationsverteilung sind in Abbildung 4.1 für verschiedene Werte des Präzisionsparameters k dargestellt.

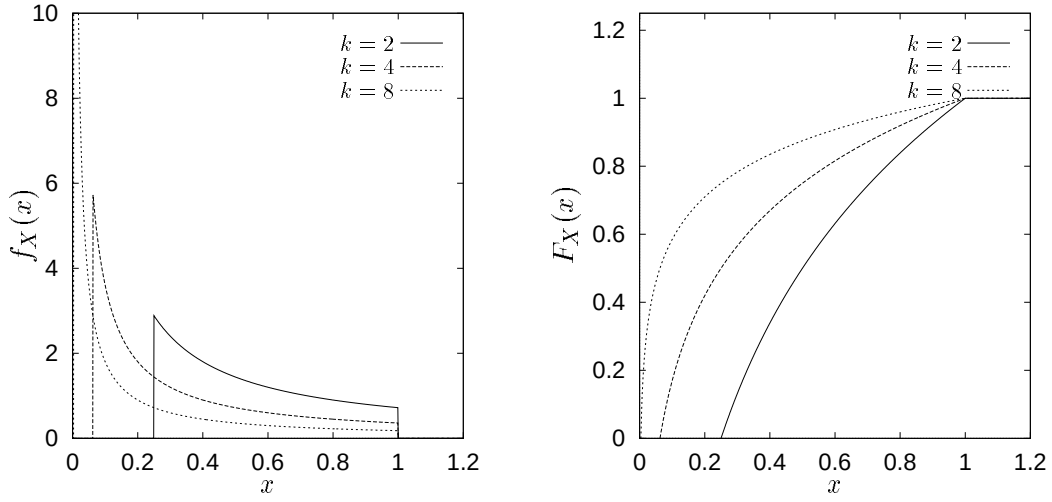

Abbildung 4.1: Dichte- und Verteilungsfunktion der BGA-Mutationsverteilung $\mathbf{B}(k, \rho)$ für verschiedene Werte des Präzisionsparameters ($k = 2, 4$ und 8; $\rho = 1$).

In Abbildung 4.2 ist der Einfluß des Schrittweitenparameters ρ wiedergegeben. Für den Präzisionsparameter $k = 2$ sind die Dichtefunktionen für verschiedene Schrittweitenparameter als gefüllte Flächen eingezeichnet. Die eingezeichneten Funktionen zeigen, daß

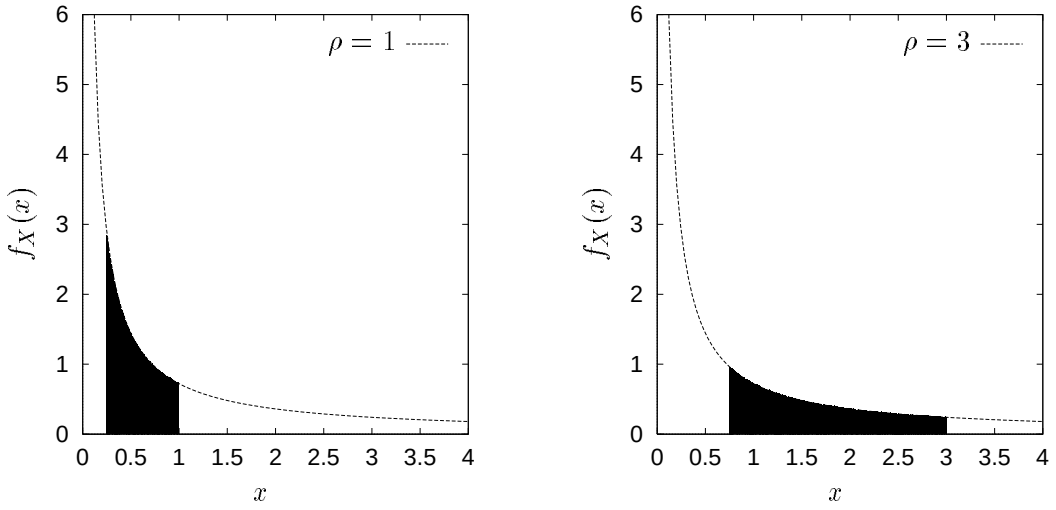

Abbildung 4.2: Dichtefunktion der BGA-Mutationsverteilung $\mathbf{B}(k, \rho)$ für verschiedene Werte des Schrittweitenparameters ($k = 2$; $\rho = 1$ und 3).

der Schrittweitenparameter ρ lediglich die Grenzen der Dichtefunktion, nicht aber deren Form, beeinflußt.

Satz 3 *Der* Erwartungswert *der Zufallsvariablen* $X \sim \mathbf{B}(k, \rho)$ *ist*

$$\boxed{E(X) = \frac{\rho}{k \ln 2} \cdot \left(1 \Leftrightarrow 2^{-k}\right)} \tag{4.19}$$

Beweis 3 *Durch Einsetzen der Dichtefunktion der BGA-Mutationsverteilung* $\mathbf{B}(k, \rho)$ *aus Satz 4.11 in Gleichung 4.3 ergibt sich für den Erwartungswert der BGA-Mutationsverteilung:*

$$\begin{aligned} E(X) &= \int_{\rho 2^{-k}}^{\rho} \frac{1}{k \ln 2} \cdot \frac{1}{x} \cdot x \, dx \\ &= \frac{1}{k \ln 2} \cdot \int_{\rho \cdot 2^{-k}}^{\rho} 1 \, dx \\ &= \frac{1}{k \ln 2} \cdot [x]_{\rho \cdot 2^{-k}}^{\rho} \\ &= \frac{\rho}{k \ln 2} \cdot \left(1 \Leftrightarrow 2^{-k}\right) \end{aligned}$$

■

Satz 4 *Die* Varianz *der Zufallsvariablen* $X \sim \mathbf{B}(k, \rho)$ *ist*

$$\boxed{Var(X) = \frac{\rho^2}{2k \ln 2} \left(1 \Leftrightarrow 2^{-2k}\right) \Leftrightarrow \left(\frac{\rho}{k \ln 2} \left(1 \Leftrightarrow 2^{-k}\right)\right)^2} \tag{4.20}$$

Beweis 4 *Allgemein ist die Varianz einer Zufallsvariablen X definiert als*

$$Var(X) = E((X \Leftrightarrow E(X))^2) \tag{4.21}$$
$$= E(X^2) \Leftrightarrow E^2(X) \tag{4.22}$$

Mit der Dichtefunktion von $\mathbf{B}(k,\rho)$ aus Gleichung 4.11 erhält man zunächst

$$E(X^2) = \int_{\rho 2^{-k}}^{\rho} \frac{1}{k \ln 2} \cdot \frac{1}{x} \cdot x^2 \, dx$$
$$= \frac{1}{k \ln 2} \cdot \left[\frac{x^2}{2} \right]_{\rho 2^{-k}}^{\rho}$$
$$= \frac{\rho^2}{2k \ln 2} \left(1 \Leftrightarrow 2^{-2k} \right)$$

Setzt man nun $E(X^2)$ und $E(X)$ in Gleichung 4.22 ein, so ergibt sich die Varianz der Zufallsvariablen $X \sim \mathbf{B}(k,\rho)$ zu

$$Var(X) = \frac{\rho^2}{2k \ln 2} \left(1 \Leftrightarrow 2^{-2k} \right) \Leftrightarrow \left(\frac{\rho}{k \ln 2} \left(1 \Leftrightarrow 2^{-k} \right) \right)^2 \tag{4.23}$$

■

Bemerkung 1 *Für große k gilt*

$$E(X) \approx \frac{\rho}{k \ln 2} \tag{4.24}$$

$$Var(X) \approx \frac{\rho^2}{2k \ln 2} \tag{4.25}$$

Die Erwartungswerte und Standardabweichungen der Zufallsvariablen $X \sim \mathbf{B}(k,\rho)$ mit $\rho = 1$ für verschiedene Werte des Präzisionsparameters k sind in Tabelle 4.3 auf Seite 69 zu finden. Interessant ist die Entwicklung der Standardabweichungen, $Var(X)$ wird maximal für $k = 4$.

Symmetrische BGA-Mutationsverteilung $\hat{\mathbf{B}}(k,\rho)$

Die symmetrische BGA-Mutationsverteilung wird durch ihre Dichte- sowie ihre Verteilungsfunktion beschrieben. Die Zufallsvariable der symmetrischen BGA-Mutationsverteilung wird im folgenden mit $\hat{X} \sim \hat{\mathbf{B}}(k,\rho)$ bezeichnet.

Satz 5 *Die* Dichtefunktion *der symmetrischen BGA-Mutationsverteilung $\hat{\mathbf{B}}(k,\rho)$ ist*

$$f_{\hat{X}}(x) = \begin{cases} \frac{1}{2k \ln 2} \cdot \frac{1}{|x|} & : \quad \rho 2^{-k} \leq |x| \leq \rho \\ 0 & : \quad sonst \end{cases} \tag{4.26}$$

Beweis 5 *Die Dichtefunktion der symmetrischen BGA-Mutationsverteilung ergibt sich durch Spiegelung und anschließende Normierung der Dichtefunktion der einfachen BGA-Mutationsverteilung* $\mathbf{B}(k,\rho)$ *aus Gleichung 4.11.*
∎

Satz 6 *Die* Verteilungsfunktion *der symmetrischen BGA-Mutationsverteilung* $\hat{\mathbf{B}}(k,\rho)$ *ist*

$$F_{\hat{X}}(x) = \begin{cases} 0.0 & : \quad x < \Leftrightarrow\rho \\ \Leftrightarrow\frac{1}{2k\ln 2} \cdot \ln(\Leftrightarrow\frac{x}{\rho}) & : \quad \Leftrightarrow\rho \leq x \leq \Leftrightarrow\rho 2^{-k} \\ 0.5 & : \quad \Leftrightarrow\rho 2^{-k} \leq x \leq \rho 2^{-k} \\ 0.5 + \frac{1}{2k\ln 2} \cdot \ln\left(\frac{x}{\rho 2^{-k}}\right) & : \quad \rho 2^{-k} \leq x \leq \rho \\ 1.0 & : \quad x > \rho \end{cases} \tag{4.27}$$

Beweis 6 *Setzt man die Dichtefunktion der BGA-Mutationsverteilung* $\hat{\mathbf{B}}(k,\rho)$ *aus Gleichung 4.26 in die Gleichung 4.2 ein, ergibt sich die Verteilungsfunktion. Die Verteilungsfunktion hat ebenso wie die Dichtefunktion zwei Trägerbereiche.*
Für $\Leftrightarrow\rho \leq x \leq \Leftrightarrow\rho 2^{-k}$*:*

$$\begin{aligned} F_{\hat{X}}(x) &= \int_{-\rho}^{x} f_{\hat{X}}(t)\,dt \\ &= \Leftrightarrow\frac{1}{2k\ln 2} \cdot \int_{-\rho}^{x} \frac{1}{t}\,dt \\ &= \Leftrightarrow\frac{1}{2k\ln 2} \cdot [\ln(\Leftrightarrow t)]_{-\rho}^{x} \\ &= \Leftrightarrow\frac{1}{2k\ln 2} \cdot \ln\left(\Leftrightarrow\frac{x}{\rho}\right) \end{aligned}$$

Für $\rho 2^{-k} \leq x \leq \rho$*:*

$$\begin{aligned} F_X(x) &= 0.5 + \int_{\rho 2^{-k}}^{x} f_{\hat{X}}(t)\,dt \\ &= 0.5 + \frac{1}{2k\ln 2} \cdot \int_{\rho 2^{-k}}^{x} \frac{1}{t}\,dt \\ &= 0.5 + \frac{1}{2k\ln 2} \cdot [\ln(t)]_{\rho 2^{-k}}^{x} \\ &= 0.5 + \frac{1}{2k\ln 2} \cdot \ln\left(\frac{x}{\rho 2^{-k}}\right) \end{aligned}$$

∎

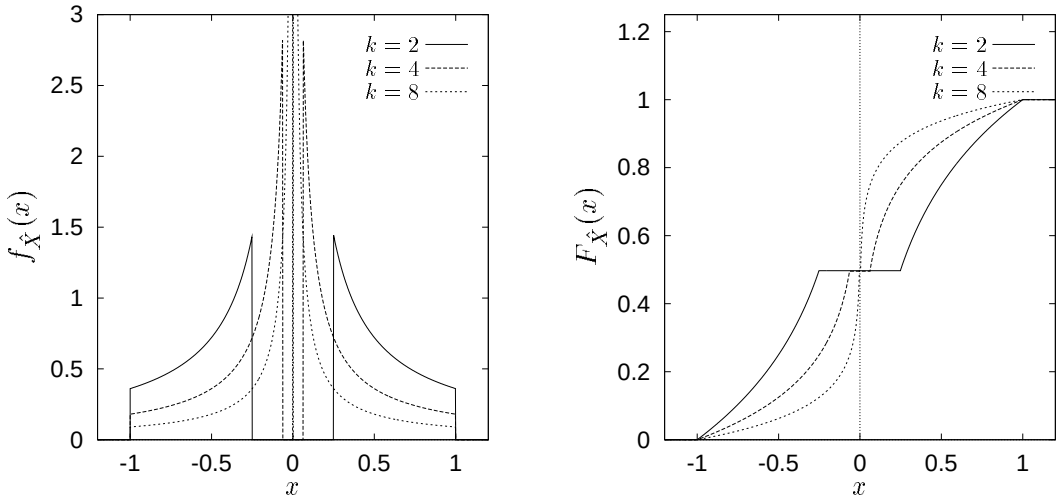

Abbildung 4.3: Dichte- und Verteilungsfunktion der Zufallsvariablen $\hat{X} \sim \hat{\mathbf{B}}(\mathbf{k}, \mathbf{1})$ für verschiedene Präzisionsparameter ($k = 2, 4$ und 8).

Die Dichte- und Verteilungsfunktion der symmetrischen BGA-Mutationsverteilung sind in Abbildung 4.3 für verschiedene Präzisionsparameter k dargestellt.

Satz 7 *Die* Varianz *der Zufallsvariablen* $\hat{X} \sim \hat{\mathbf{B}}(k, \rho)$ *ist*

$$\boxed{Var(\hat{X}) = \frac{1}{2k \ln 2} \cdot \rho^2 \cdot (1 \Leftrightarrow 2^{-2k})} \tag{4.28}$$

Beweis 7 *Da für den Erwartungswert der symmetrischen BGA-Mutationsverteilung* $E(\hat{X}) = 0$ *gilt, folgt aus der Formel für die Varianz (Gleichung 4.6):*

$$\begin{aligned}
Var(\hat{X}) &= E(\hat{X}^2) \\
&= \int_{-\rho}^{\rho} f_{\hat{X}}(x) \cdot x^2 \, dx \\
&= \frac{1}{2k \ln 2} \cdot \left(\int_{-\rho 2^{-k}}^{-\rho} x \, dx + \int_{\rho 2^{-k}}^{\rho} x \, dx \right) \\
&= \frac{1}{2k \ln 2} \cdot 2 \cdot \int_{\rho 2^{-k}}^{\rho} x \, dx \\
&= \frac{1}{2k \ln 2} \cdot 2 \cdot \left[\frac{1}{2} \cdot x^2 \right]_{\rho 2^{-k}}^{\rho} \\
&= \frac{1}{2k \ln 2} \cdot 2 \cdot \left(\frac{1}{2} \cdot \left(\rho^2 \Leftrightarrow (\rho 2^{-k})^2 \right) \right) \\
&= \frac{1}{2k \ln 2} \cdot \rho^2 \cdot (1 \Leftrightarrow 2^{-2k})
\end{aligned}$$

∎

Satz 8 *Die* Wölbung *der Zufallsvariablen* $\hat{X} \sim \hat{\mathbf{B}}(k,\rho)$ *ist*

$$\boxed{Kurt(\hat{X}) = \frac{1}{Var^2(\hat{X})} \cdot \frac{1}{4k \ln 2} \cdot \rho(1 \Leftrightarrow 2^{-4k})} \tag{4.29}$$

Beweis 8 *Setzt man die Dichtefunktion* $f_{\hat{X}}(x)$ *aus Gleichung 4.26 in die Formel für die Wölbung (Gleichung 4.8) ein, so ergibt sich unter Ausnutzung der Symmetrie von* $f_{\hat{X}}(x)$ $(E(\hat{X}) = 0)$*:*

$$\begin{aligned}
Kurt(\hat{X}) &= \frac{E(\hat{X} \Leftrightarrow E(\hat{X}))^4}{Var(X)^2} \\
&= \frac{1}{Var^2(\hat{X})} \cdot E(\hat{X}^4) \\
&= \frac{1}{Var^2(\hat{X})} \cdot \int_{-\rho}^{\rho} f_{\hat{X}}(x) \cdot x^4 \, dx \\
&= \frac{1}{Var^2(\hat{X})} \cdot \frac{1}{2k \ln 2} \cdot \left(\int_{-\rho 2^{-k}}^{-\rho} x^3 \, dx + \int_{\rho 2^{-k}}^{\rho} x^3 \, dx \right) \\
&= \frac{1}{Var^2(\hat{X})} \cdot \frac{1}{2k \ln 2} \cdot 2 \cdot \int_{\rho 2^{-k}}^{\rho} x^3 \, dx \\
&= \frac{1}{Var^2(\hat{X})} \cdot \frac{1}{2k \ln 2} \cdot \left(2 \cdot \left[\frac{1}{4} \cdot y^4 \right]_{\rho 2^{-k}}^{\rho} \right) \\
&= \frac{1}{Var^2(\hat{X})} \cdot \frac{1}{4k \ln 2} \cdot \rho(1 \Leftrightarrow 2^{-4k})
\end{aligned}$$

■

In Tabelle 4.3 sind Standardabweichung und Wölbung der Zufallsvariablen $\hat{X} \sim \hat{\mathbf{B}}(k,\rho)$ für verschiedene Werte des Präzisionsparameters k aufgelistet.

4.3 Weitere Mutationsverteilungen

Die Mutationsverteilungen von Evolutionären Algorithmen orientieren sich an der Art des Suchraums. Typischerweise findet man in Genetischen Algorithmen, deren Genotypen Punkte eines binären Suchraums repräsentieren, die Binominalverteilung und in Evolutionsstrategien, die meist direkt im Raum der reellen Zahlen suchen, die Normalverteilung vor. Mutationsverteilungen im reellwertigen Suchraum sind jedoch keineswegs auf die Normalverteilung beschränkt. In den Abschnitten 3.2.4 und 3.2.4, in denen die allgemeinen genetischen Operatoren für reelle Suchräume beschrieben wurden, findet sich bereits eine Auswahl verschiedener stetiger Verteilungen. Zu diesen zählen neben Normalverteilung auch Gleichverteilung, Exponentialverteilung, gleichschenklige Dreiecksverteilung, Cauchyverteilung und BGA-Mutationsverteilung. Die Dichte- und Transformationsfunktionen dieser Verteilungen sind in Tabelle 4.4 aufgeführt. Die BGA-Mutationsverteilung hat, wie die

k	$\mathbf{B}(k,1)$		$\hat{\mathbf{B}}(k,1)$	
	$E(X)$	$\sqrt{Var(X)}$	$\sqrt{Var(\hat{X})}$	$Kurt(\hat{X})$
1	0.721	0.144	0.736	1.155
2	0.541	0.213	0.582	1.571
4	0.338	0.256	0.424	2.794
8	0.180	0.241	0.300	5.545
16	0.090	0.192	0.212	11.090
32	0.045	0.143	0.150	22.180
64	0.023	0.104	0.106	44.360

Tabelle 4.3: Spezielle Verteilungsmomente der beiden BGA-Mutationsverteilungen $\mathbf{B}(k,1)$ und $\hat{\mathbf{B}}(k,1)$ für verschiedene Werte des Präzisionsparameters k.

Gleichverteilung und die Dreiecksverteilung, eine Obergrenze für die Schrittweite. Sie weist als einzige eine Untergrenze der Schrittweite auf, wodurch die Genauigkeit der Suche eingeschränkt wird.

Abbildung 4.4 zeigt die Dichtefunktionen der verschiedenen Mutationsverteilungen. Die Werte wurden durch die in Tabelle 4.4 aufgeführten Transformationfunktionen aus einer gleichverteilten Zufallszahl generiert.

Die Eignung einer Mutationsverteilung hängt letztlich von der Problemstellung und der Art des Evolutionären Algorithmus ab. Ein EA ohne Schrittweitenregelung benötigt in Bezug auf den Schrittweitenparameter eine robustere Mutationsverteilung als ein EA, dessen Schrittweitenparameter im Laufe der Suche angepaßt wird.

Kappler machte kürzlich erste Untersuchungen mit einer Evolutionsstrategie, deren Objektvariablen nicht normalverteilt, sondern nach einer Cauchyverteilung mutiert werden ([Kap96]). Da die Cauchyverteilung mit höherer Wahrscheinlichkeit große Schritte erzeugt als die Normalverteilung, wird erwartet, daß sie eher in der Lage ist, lokale Optima wieder zu verlassen. Für eine zweidimensionale einfache Zielfunktion zeigte Kappler, daß eine $(1+\lambda)$-ES mit fester Schrittweite und Cauchyverteilung ein robusteres Verhalten bezüglich der Schrittweite aufweist als mit der Normalverteilung.

4.4 Vergleich der Mutationsverteilungen

Zur Bewertung der Mutationsverteilungen sind Maße notwendig, die deren Effizienz und Robustheit bezüglich ihrer Parameter charakterisieren. Für diesen Zweck werden im folgenden Effizienz- und Robustheitsmaße eingeführt. Zunächst wird jedoch das Mutations-Selektions-Verfahren und das Abstandsmaß, auf dem die Untersuchung basiert, spezifiziert.

Verteilung	$f_X(x)$	$g(u)$
$\mathbf{B}(k,\rho)$	$\frac{\rho}{k\cdot\ln(2)}\cdot\frac{1}{x}$: $\rho\cdot 2^{-k}\le x\le\rho$ 0 : $sonst$	$\rho\cdot 2^{-k\cdot u}$
$\mathbf{C}(\sigma)$	$\frac{\sigma}{\pi(x^2+\sigma^2)}$	$\sigma\tan(\pi u)$
$\mathbf{E}(\lambda)$	$\lambda\cdot e^{-\lambda x}\quad x>0$	$-\frac{1}{\lambda}\log(u)$
$\mathbf{N}(\mu,\sigma)$	$\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$\mu+\sigma\sqrt{-2\ln u_1}\sin(2\pi u_2)$ $\mu+\sigma\sqrt{-2\ln u_1}\cos(2\pi u_2)$
$\mathbf{T}(a)$	$\frac{2}{a}\left(1-\frac{x}{a}\right)\quad 0<x\le a$	$a(1-\sqrt{u})$
$\mathbf{U}(a,b)$	$\frac{1}{b-a}$: $a\le x\le b$ 0 : $sonst$	$a+u(b-a)$

Tabelle 4.4: Dichtefunktionen $f_X(x)$ und Transformationsfunktionen $g(u)$, die aus der gleichverteilten Zufallszahl $u\sim\mathbf{U}([0,1])$ eine Zufallszahl der entsprechenden Verteilung erzeugt. Die Transformation der Normalverteilung erzeugt aus zwei unabhängigen gleichverteilten Zufallszahlen $u_1,u_2\sim\mathbf{U}([0,1])$ zwei normalverteilte Zufallszahlen. Bei der Exponentialverteilung wird die gleichverteilte Zufallsvariable $u\sim\mathbf{U}((0,1])$ verwendet. (**B**: BGA-V., **C**: Cauchyv., **E**: Exponentialv., **N**: Normalv., **T**: Dreiecksv.,**U**: Gleichv.)

4.4.1 Mutations-Selektions-Verfahren

Im nachstehenden schematischen Algorithmus 4.1 ist ein Mutations-Selektions-Verfahren beschrieben, das mit verschiedenen Mutationsverteilungen parametrisiert werden kann.

Algorithmus 4.1 *(1+1)-EA*

(1+1)-EA$(f,\vec{x}^0,\epsilon,,,\rho,\nu)$
1 $\vec{x}\leftarrow\vec{x}^0$
2 **while** $\delta(\vec{x})>\epsilon$
3 **do** $\vec{z}\leftarrow\mathbb{M}^{\mathrm{EA}}_{\Gamma,\rho,\nu}(\vec{x})$ Mutation
4 **if** $f(\vec{z})<f(\vec{x})$
5 **then** $\vec{x}\leftarrow\vec{z}$ Selektion
6 **return** $\vec{x}$

Die Parameter , , ρ und ν entsprechen den Parametern des in Abschnitt 3.2.4 eingeführten allgemeinen Mutationsoperators $\mathbb{M}^{\mathrm{EA}}_{\Gamma,\rho,\nu}(\vec{x})$. , $\in\{B,C,E,N,T,U\}$ spezifiziert die Art der eindimensionalen Mutationsverteilung, ρ den Schrittweitenparameter und ν die Anzahl der gleichzeitig zu mutierenden Objektvariablen.

4.4.2 Zielfunktion und Abstandsmaß

Der Untersuchung der Mutationsverteilungen liegt eine symmetrische, unimodale Zielfunktion $f(\vec{x})$, deren Optimum $\vec{x}^*$ sich im Ursprung befindet, zugrunde. Eine einfache Funktion,

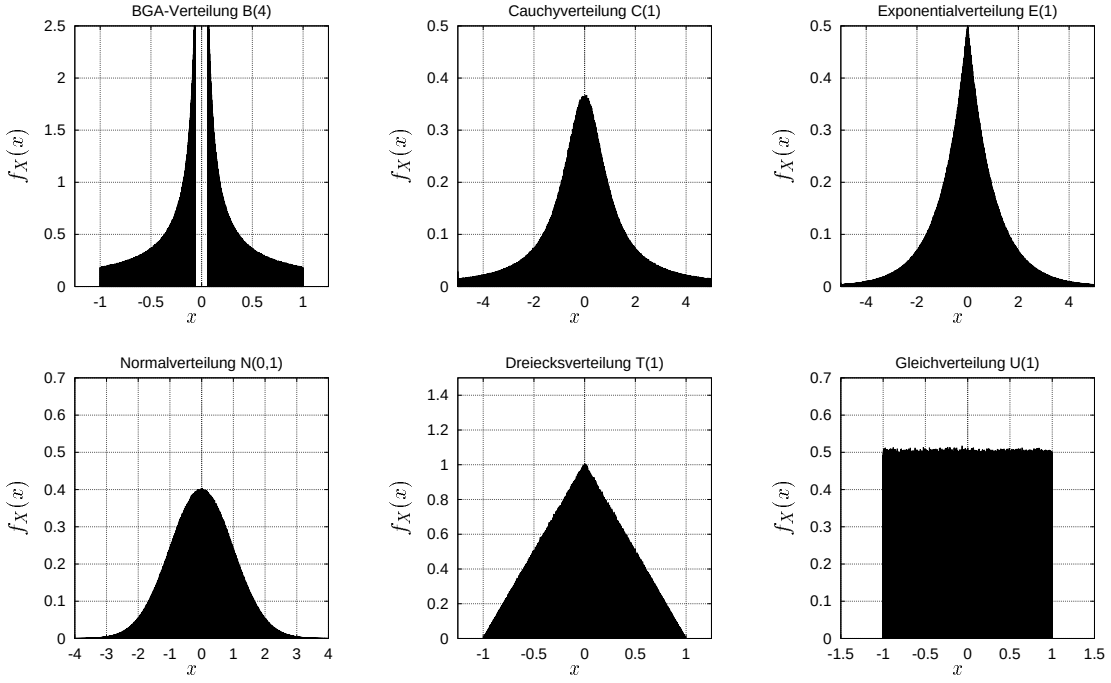

Abbildung 4.4: Dichtefunktionen der verschiedenen Mutationsverteilungen in ihrer symmetrischen Form.

die diese Eigenschaft hat, ist das Kugelmodell:

$$f(\vec{x}) = \sum_{i=1}^{n} x_i^2 \tag{4.30}$$

Als Fehlermaß $\delta(\vec{x})$ wird der euklidische Abstand des Punktes $\vec{x}$ zum Optimum $\vec{x}^*$ im Raum der Objektvariablen verwendet:

$$\delta(\vec{x}) = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{f(\vec{x})} \tag{4.31}$$

Für den eindimensionalen Fall ($n = 1$) vereinfacht sich die Zielfunktion und der Fehler zu:

$$f(x) = x^2 \tag{4.32}$$

$$\delta(x) = |x| \tag{4.33}$$

Die Notation für die Zielfunktion $f(\vec{x})$ ist nicht mit der Notation für die Dichtefunktion $f_X(x)$ zu verwechseln.

4.4.3 Effizienz

Anzahl der Funktionsauswertungen

Ein Maß, das eine quantitative Aussage über die Konvergenzgeschwindigkeit eines Mutations-Selektions-Verfahrens erlaubt, ist der Erwartungswert der Anzahl der Funktionsauswertungen, die benötigt werden, um das Optimum einer Zielfunktion $f(\vec{x})$ von einem

gegebenen Startpunkt $\vec{x}^0$ bis auf eine bestimmte Genauigkeit ϵ an das Optimum $\vec{x}^*$ anzunähern.

Dieses Konvergenzmaß wird, außer von dem Mutations-Selektions-Verfahren selbst, von drei weiteren Faktoren beeinflußt:

- Zielfunktion $f(\vec{x})$
- Startpunkt $\vec{x}^0$
- gewünschte Zielannäherung ϵ bzgl. $\delta(\vec{x})$

Dieses Konvergenzmaß bietet den Vorteil, daß sich sein Schätzwert, die mittlere Anzahl der Funktionsauswertungen, leicht empirisch bestimmen läßt. Da das Maß nur Aussagen bezüglich einer Kombination dieser drei Faktoren erlaubt, ist der Informationsgehalt allerdings recht speziell. Um eine allgemeinere Aussage über die Konvergenzgeschwindigkeit eines Mutations-Selektions-Verfahrens zu erhalten, müssen repräsentative Kombinationen ausgewählt werden.

Im folgenden wird das Verhalten der Mutationsverteilungen zunächst an der oben eingeführten eindimensionalen Zielfunktion $f(x)$ aus Gleichung 4.32 untersucht. Abbildung 4.5 zeigt die mittlere Anzahl der Funktionsauswertungen, die in Abhängigkeit der Schrittweite ρ benötigt werden, um von dem Startpunkt $x^0 = 1.0$ die Zielannäherung von $\epsilon = 0.1$ zu erreichen.

Die Funktionsverläufe zeigen die Bedeutung des Präzisionsparameters k der BGA-Mutationsverteilung $\mathbf{B}(k, \rho)$ für Effizienz und Robustheit in Bezug auf den Schrittweitenparameter ρ. Ein kleinerer Präzisionsparameter führt bei geeignetem Schrittweitenparameter ρ zu höherer Effizienz. Gleichzeitig sinkt jedoch die Robustheit bezüglich des Schrittweitenparameters ρ. Zu gegebenem k führt ein ρ zu einer minimalen Schrittweite $\rho 2^{-k}$. Ein zu großes ρ führt dazu, daß die vorgegebene Zielannäherung ϵ aufgrund einer zu großen minimalen Schrittweite nicht erreicht werden kann. In Abbildung 4.5 wird dieser Fall an den abgebrochenen Funktionsverläufen der BGA-Mutationsverteilungen $\mathbf{B}(4, \rho)$ und $\mathbf{B}(8, \rho)$ sichtbar.

Im Vergleich zur Normal- und Gleichverteilung erweist sich die BGA-Mutationsverteilung $\mathbf{B}(16, \rho)$ zwar als weniger effizient, dafür aber als äußerst robust bezüglich des Schrittweitenparameters ρ. Die minimale Anzahl von Funktionsauswertungen ist fast dreimal so hoch wie die der beiden anderen Mutationsverteilungen. Ab einem bestimmten Wert von ρ bleibt die Effizienz jedoch über einen weiten Bereich annähernd konstant. Der Schrittweitenparameter muß also nicht so genau eingestellt werden wie bei der Normal- und Gleichverteilung, um eine gute Effizienz zu erzielen. Grundsätzlich gilt bei den BGA-Mutationsverteilungen, daß die Effizenz des Suchverfahrens durch einen zu großen Schrittweitenparameter weniger stark beeinträchtigt wird als durch einen zu kleinen Schrittweitenparameter.

Das Verhalten der verschiedenen Mutationsverteilungen bei der mehrdimensionalen Zielfunktion $f(\vec{x})$ aus Gleichung 4.30 ist in Abbildung 4.6 für 10 Dimensionen ($n = 10$) dargestellt. Der Startpunkt $\vec{x}^0$ ist so gewählt, daß sein Fehler $\delta(\vec{x}^0) = 1.0$ ist und alle $x_i = \sqrt{1/n}$ denselben Wert haben. Die vorgegebene Zielannäherung beträgt $\epsilon = 0.1$. Bei einer mehrdimensionalen Funktion muß auf algorithmischer Seite die Anzahl der gleichzeitig zu mutierenden Variablen festgelegt werden. In den dargestellten Experimenten werden entweder genau eine ($\nu = 1$) oder alle zehn Objektvariablen ($\nu = 10$) mutiert.

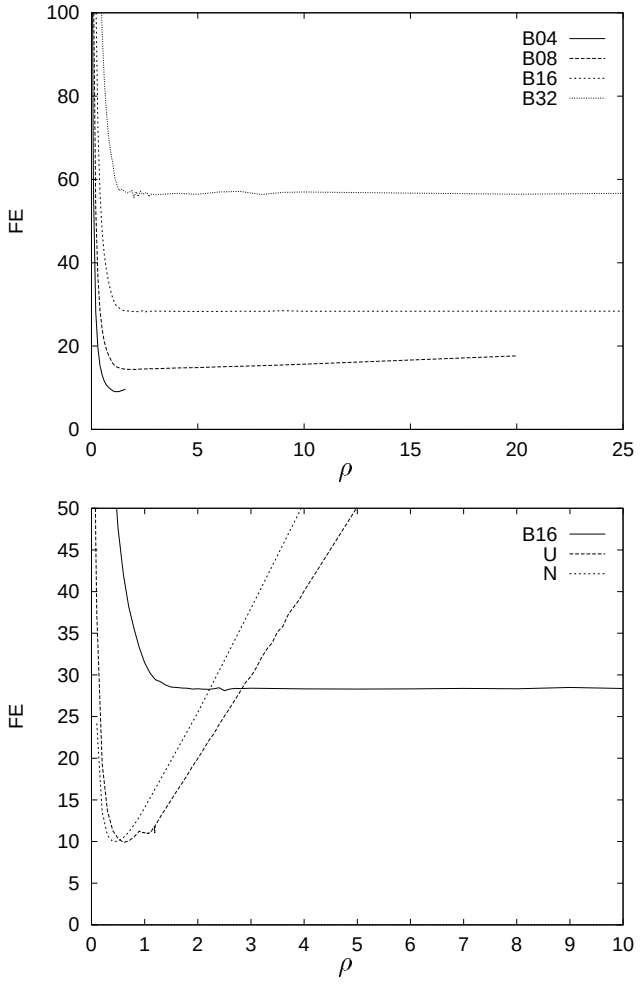

Abbildung 4.5: Einfluß von Mutationsverteilung und Schrittweitenparameter ρ auf die mittlere Anzahl von Funktionsauswertungen (FE) für das eindimensionale Kugelmodell, Startpunkt $x^0 = 1.0$ und Zielannäherung $\epsilon = 0.1$. Für BGA-Mutationsverteilungen B'k'=$\mathbf{B}(k,1)$, Gleichverteilung U=$\mathbf{U}(0,1)$ und Normalverteilung N=$\mathbf{N}(0,1)$ sind die Funktionsauswertungen in Abhängigkeit des Schrittweitenparameters ρ eingetragen. Aufgrund der unterschiedlichen Skalierungen ist das Resultat der BGA-Mutationsverteilung B16 zum besseren Vergleich in beiden Abbildungen eingezeichnet.

Qualitativ entsprechen die Resultate den Ergebnissen des eindimensionalen Falls. Die BGA-Mutationsverteilungen zeichnen sich in Bezug auf den Schrittweitenparameter ρ auch hier durch eine hohe Robustheit aus. Die Effizienzunterschiede zwischen den verschiedenen Mutationsverteilungen sind jedoch geringer als im eindimensionalen Fall. Die Mutations-Selektions-Verfahren, die alle Objektparameter gleichzeitig mutieren ($\nu = 10$), sind aufgrund der groben Zielannäherung dieser speziellen Problemstellung wesentlich effizienter als die, die nur einen einzigen mutieren. In Abbildung 4.7 ist die Zielannäherung von Mutations-Selektions-Verfahren dargestellt, die unterschiedlich viele Objektvariablen gleichzeitig mutieren. Die Mutations-Selektions-Verfahren basieren auf der BGA-Mutation mit Präzisionsparameter $k = 16$ und konstantem Schrittweitenparameter $\rho = 1$.

Die Entwicklung des Fehlers $\delta(\vec{x})$ über die Anzahl der Funktionsauswertungen zeigt, daß die Mutation mehrerer Objektvariablen bei festem Schrittweitenparameter nur für einen bestimmten Bereich effizienter ist. Bei dieser Problemstellung führt die gleichzeitige

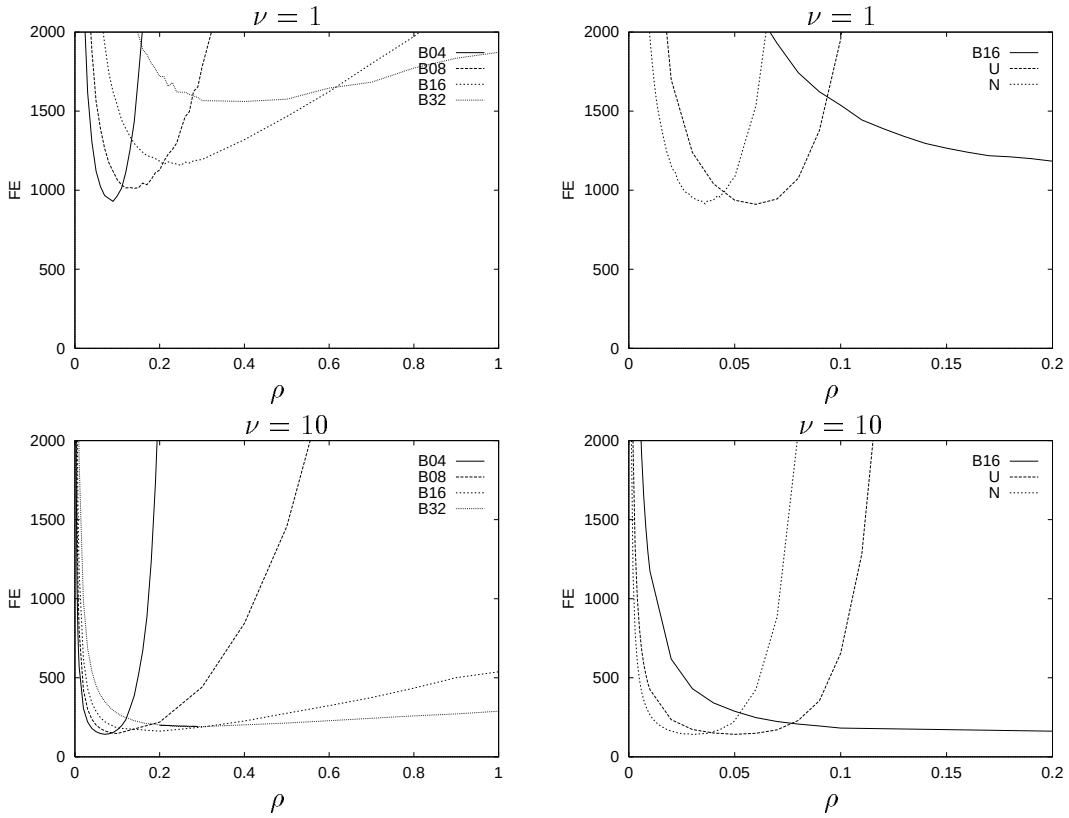

Abbildung 4.6: Einfluß von Mutationsverteilung und Schrittweitenparameter ρ auf die mittlere Anzahl von Funktionsauswertungen (FE) bei der zehndimensionalen Zielfunktion $f(\vec{x})$. Der Startpunkt ist $\vec{x}^0 = (\sqrt{0.1}, \ldots, \sqrt{0.1})$, die vorgegebene Zielannäherung $\epsilon = 0.1$. Auf der Abszisse ist der Schrittweitenparameter und auf der Ordinate die Anzahl der Funktionsauswertungen eingetragen. Der Parameter ν in den Bildüberschriften gibt die Anzahl der gleichzeitig zu mutierenden Variablen an. Zur besseren Lesbarkeit sind die Abszissen der linken und rechten Abbildungen unterschiedlich skaliert.

Mutation aller Variablen aufgrund der minimalen Schrittweite der BGA-Mutation schon früh zu Stagnation. Die gleichzeitige Mutation von zehn Variablen ist bis zum Fehler $\delta(\vec{x}) = 3.0$ effizienter als die Mutation einer Variablen. Selbst die Mutation von zwei Variablen liefert nur bis zum Fehler $\delta(\vec{x}) = 0.01$ ein besseres Verhalten. Der optimale Wert der Anzahl der gleichzeitig zu mutierenden Variablen ist für eine bestimmte Problemstellung von der gewünschten Zielannäherung abhängig. Der Stagnation, die durch die Mutation mehrerer Objektvariablen auftritt, kann durch die Anpassung des Schrittweitenparameters ρ begegnet werden.

Die Entwicklung der BGA-Mutation zielte auf ein einfach zu implementierendes Mutationsverfahren ab, das in der Lage ist, sich dem Optimum auch ohne Steuerung des Schrittweitenparameters über mehrere Zehnerpotenzen effizient anzunähern. Um dieses Ziel zu erreichen, verändert die BGA-Mutation nur eine Objektvariable. Wie in Abbildung 4.7 zu sehen ist, erfolgt mit dieser Mutation eine Zielannäherung über fünf Zehnerpotenzen.

Mutationsverfahren, die alle Objektvariablen gleichzeitig und unabhängig voneinander durch eine beliebige Mutationsverteilung ändern, nähern sich für $n \gg 1$ aufgrund des zentralen Grenzwertsatzes der Normalverteilung. Die charakteristischen Eigenschaften spezieller Verteilungen gehen in diesem Fall also mit zunehmender Dimension verloren.

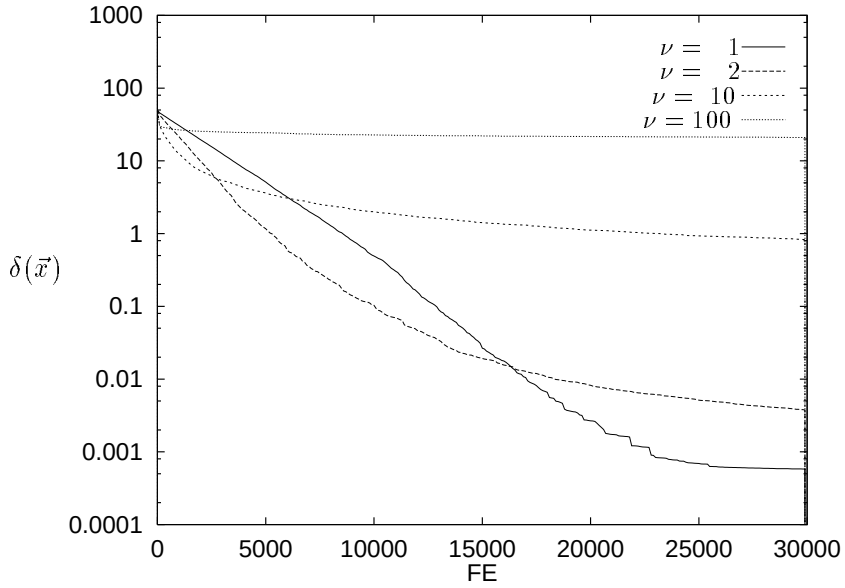

Abbildung 4.7: Zielannäherung von Mutations-Selektions-Verfahren, die sich in der Anzahl ν der gleichzeitig mutierten Objektvariablen unterscheiden. Die Mutations-Selektions-Verfahren basieren auf der BGA-Mutationsverteilung $\mathbf{B}(16{,}1)$. Dargestellt ist der Fehler $\delta(\vec{x})$ in Abhängigkeit der Funktionsauswertungen (FE). Die Werte von $\delta(\vec{x})$ sind Durchschnittswerte aus 100 Simulationsläufen. (Kugelmodell, $n = 100$, $\vec{x}^0 = (5, \ldots, 5)^{\mathrm{T}}$)

Fortschrittsgeschwindigkeit im eindimensionalen Fall (n=1)

Ein Leistungsmaß, das zur Analyse von Evolutionsstrategien, aber auch von reinen Zufallssuchverfahren verwendet wird, ist der *Erwartungswert der Verbesserung im Raum der Objektvariablen* ([Rec73], [Sch81], [BS91], [TŽ88], [SW81]). Dieses Maß, das die *lokale* Zielannäherung des Suchverfahrens beschreibt, wird die *Fortschrittsgeschwindigkeit* eines Suchverfahrens genannt.

Im Gegensatz zu dem oben vorgestellten Konvergenzmaß *Erwartungswert der Funktionsauswertungen* wird die Zielannäherung des Verfahrens nicht mehr auf den gesamten Suchraum, sondern nur auf die *nähere* Umgebung eines ausgewählten Bezugspunkts bezogen. Damit hängt dieses Maß nicht mehr von der Zielannäherung ϵ, sondern nur von den beiden folgenden Faktoren ab:

- Zielfunktion $f(x)$
- Startpunkt x^0

Mit $r = \delta(x^0)$ wird im folgenden der Fehler des Startpunkts x^0 bezeichnet.

Definition 13 (Fortschrittsgeschwindigkeit) *Die* Fortschrittsgeschwindigkeit φ *eines Mutations-Selektions-Verfahrens zu einer gegebenen Zielfunktion $f(x)$ und einem gegebenen Punkt x^0 ist definiert als der Erwartungswert der Verbesserung, die durch eine Mutation erzielt wird:*

$$\varphi = E\left(\delta(x^0) \Leftrightarrow \delta(x')\right) \tag{4.34}$$

wobei x' der durch Mutation aus x^0 entstandene Punkt und $\delta(x)$ das in Gleichung 4.31 eingeführte Fehlermaß ist.

Zur einfacheren Handhabung basiert die Berechnung der Fortschrittsgeschwindigkeit eines Mutations-Selektions-Verfahrens auf ihrer *asymmetrischen* Mutationsverteilung, die ausschließlich Schritte in Richtung des Minimums x^* macht. Dazu wird zunächst die Fortschrittsgeschwindigkeit φ^{asym} eines Mutations-Selektions-Verfahrens, das nur Schritte in Richtung des Minimums macht, eingeführt. Aus der Fortschrittsgeschwindigkeit eines Mutations-Selektions-Verfahrens mit asymmetrischer Mutationsverteilung kann leicht die Fortschrittsgeschwindigkeit eines Mutations-Selektions-Verfahrens mit entsprechender symmetrischer Mutationsverteilung berechnet werden. Die Fortschrittsgeschwindigkeit φ^{asym} eines Mutations-Selektions-Verfahrens, dessen asymmetrische Mutationsverteilung die Dichte $f_X(x)$ hat, ist:

$$\varphi^{\text{asym}} = \int_0^\infty f_X(x) \cdot h(x)\, dx \tag{4.35}$$

wobei $h(s)$ die Verbesserung eines Schritts s zu einem gegebenen Startpunkt x^0 liefert:

$$h(s) = \begin{cases} r \Leftrightarrow |s \Leftrightarrow r| & : \quad 0 \leq s \leq 2r \\ 0 & : \quad sonst \end{cases} \tag{4.36}$$

Die Funktion $h(s)$ ist in Abbildung 4.8 dargestellt.

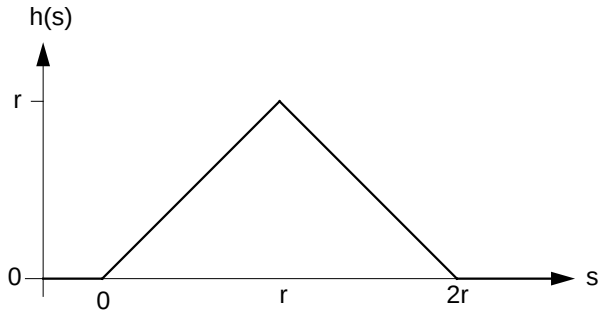

Abbildung 4.8: Die Funktion $h(s)$ gibt die Verbesserung eines Schritts der Länge s in Richtung des Optimums an. Der Abstand des Startpunkts zu Optimum beträgt r.

Da Mutations-Selektions-Verfahren nicht ausschließlich Schritte in die Richtung des Minimums machen, sondern mit derselben Wahrscheinlichkeit in die andere Richtung mutieren, ist die wirkliche Mutationsverteilung symmetrisch. Die Fortschrittsgeschwindigkeit φ^{sym} dieser *symmetrischen* Mutationsverteilung, die sich aus der asymmetrischen Mutationsverteilung und einer Booleschen Variablen zusammensetzt, kann aus der Fortschrittsgeschwindigkeit φ^{asym} der asymmetrischen Mutationsverteilung berechnet werden:

$$\varphi^{\text{sym}} = \frac{1}{2}\, \varphi^{\text{asym}} \tag{4.37}$$

Im folgenden wird durch φ die Fortschrittsgeschwindigkeit einer symmetrischen Mutationsverteilung bezeichnet.

$$\varphi^{\text{sym}} = \frac{1}{2} \cdot \int_0^\infty f_X(x) \cdot h(x)\, dx \tag{4.38}$$

Fallunterscheidung. Zur Berechnung der Fortschrittsgeschwindigkeit werden in Abhängigkeit der Größen r, $s^{\min}$ und $s^{\max}$ vier Fälle unterschieden. $s^{\min}$ und $s^{\max}$ bezeichnen den kleinsten beziehungsweise größten Wert, den eine Zufallsvariable der verwendeten Mutationsverteilung annehmen kann. In den Abbildungen 4.9 und 4.10 sind die vier Fälle schematisch dargestellt.

$$\begin{array}{ll} \text{Fall 1:} & s^{\max} \leq r \\ \text{Fall 2:} & s^{\min} < r < s^{\max} \\ \text{Fall 3:} & r \leq s^{\min} \\ \text{Fall 4:} & 2r < s^{\min} \end{array} \tag{4.39}$$

Da Fall 4 zu keiner Verbesserung führt, wird er von der weiteren Betrachtung ausgeschlossen. Die Fälle 2 und 3 werden aufgrund des Verhältnisses zwischen $s^{\max}$ und $2r$ noch einmal jeweils in die Unterfälle (a) und (b) aufgeteilt:

$$\begin{array}{ll} \text{(a):} & s^{\max} \leq 2r \\ \text{(b):} & 2r < s^{\max} \end{array} \tag{4.40}$$

Durch die Fallunterscheidung ergibt sich aus Gleichung 4.38 für die Fortschrittsgeschwindigkeit φ eines Mutations-Selektions-Verfahrens, dessen Mutationsverteilung die Dichte $f_X(x)$ hat und Schrittweiten im Intervall $[s^{\min}, s^{\max}]$ erzeugt:

$$\varphi = \begin{cases} \frac{1}{2} \cdot \int_{s^{\min}}^{s^{\max}} f_X(x) \cdot x \, dx & : \quad s^{\max} \leq r \\ \frac{1}{2} \cdot \left(\int_{s^{\min}}^{r} f_X(x) \cdot x \, dx + \int_{r}^{y} f_X(x) \cdot (2r \Leftrightarrow x) \, dx \right) & : \quad s^{\min} < r < s^{\max} \\ \frac{1}{2} \cdot \int_{s^{\min}}^{y} f_X(x) \cdot (2r \Leftrightarrow x) \, dx & : \quad r \leq s^{\min} \end{cases} \tag{4.41}$$

mit $y = \min(s^{\max}, 2r)$.

Diese Formel für die Fortschrittsgeschwindigkeit stellt die Basis für die Berechnung der Fortschrittsgeschwindigkeiten der verschiedenen Mutationsverteilungen dar. Im folgenden wird die Fortschrittsgeschwindigkeit für die BGA-Mutationsverteilung, die Gleichverteilung und die Normalverteilung untersucht.

Satz 9 *Die Fortschrittsgeschwindigkeit eines Mutations-Selektions-Verfahrens, das die BGA-Mutationsverteilung* $\mathbf{B}(k, \rho)$ *verwendet, ist:*

$$\varphi = \begin{cases} \frac{1}{2k \ln(2)} \cdot \rho (1 \Leftrightarrow 2^{-k}) & : \quad s^{\max} \leq r \\ \frac{1}{2k \ln(2)} \left(2r \left(1 + \ln\left(\frac{y}{r}\right) \right) \Leftrightarrow \rho 2^{-k} \Leftrightarrow y \right) & : \quad s^{\min} < r < s^{\max} \\ \frac{1}{2k \ln(2)} \left(2r \ln\left(\frac{y}{\rho 2^{-k}} \right) \Leftrightarrow y + 2^{-k} \right) & : \quad r \leq s^{\min} \end{cases} \tag{4.42}$$

mit $y = \min(\rho, 2r)$.

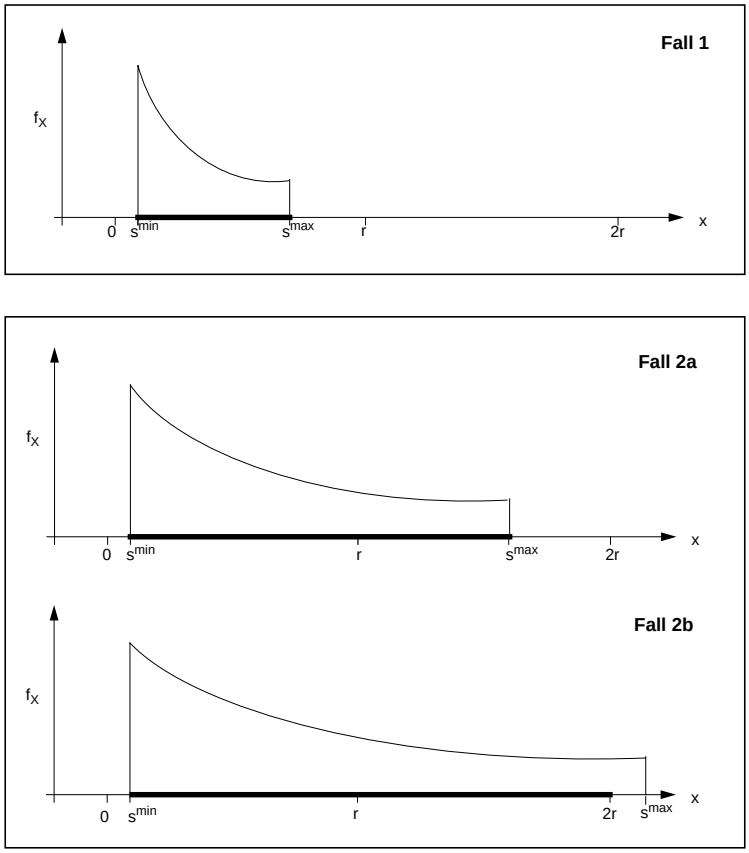

Abbildung 4.9: Skizze zur Berechnung der Fortschrittsgeschwindigkeit (Fall 1 und 2 aus Gleichungen 4.39): In den Abbildungen ist die Dichte einer beidseitig beschränkten asymmetrischen Mutationsverteilung dargestellt. $s^{\min}$ und $s^{\max}$ bezeichnen die minimale beziehungsweise maximale Schrittweite. r ist der Abstand des Startpunkts zum Optimum. Der Erfolgsbereich ist fett unterlegt.

Beweis 9 *Die Schrittweiten der BGA-Mutationsverteilung* $\mathbf{B}(k,\rho)$ *liegen im Intervall* $[\rho 2^{-k},\rho]$. *Für die extremen Schrittweiten ergibt sich damit:*

$$s^{\min} = \rho 2^{-k} \tag{4.43}$$

$$s^{\max} = \rho \tag{4.44}$$

Fall 1: $s^{\max} \leq r$

Setzt man die Dichtefunktion der asymmetrischen BGA-Mutationsverteilung $f_X(x)$ *aus Gleichung 4.15 in Gleichung 4.41 ein, erhält man:*

$$\begin{aligned}\varphi &= \frac{1}{2}\left(\frac{1}{k\ln(2)}\left(\int_{\rho 2^{-k}}^{\rho}\frac{1}{x}\cdot x\,dx\right)\right)\\ &= \frac{1}{2k\ln(2)}\cdot(\rho \Leftrightarrow \rho 2^{-k})\\ &= \frac{1}{2k\ln(2)}\rho(1 \Leftrightarrow 2^{-k})\end{aligned}$$

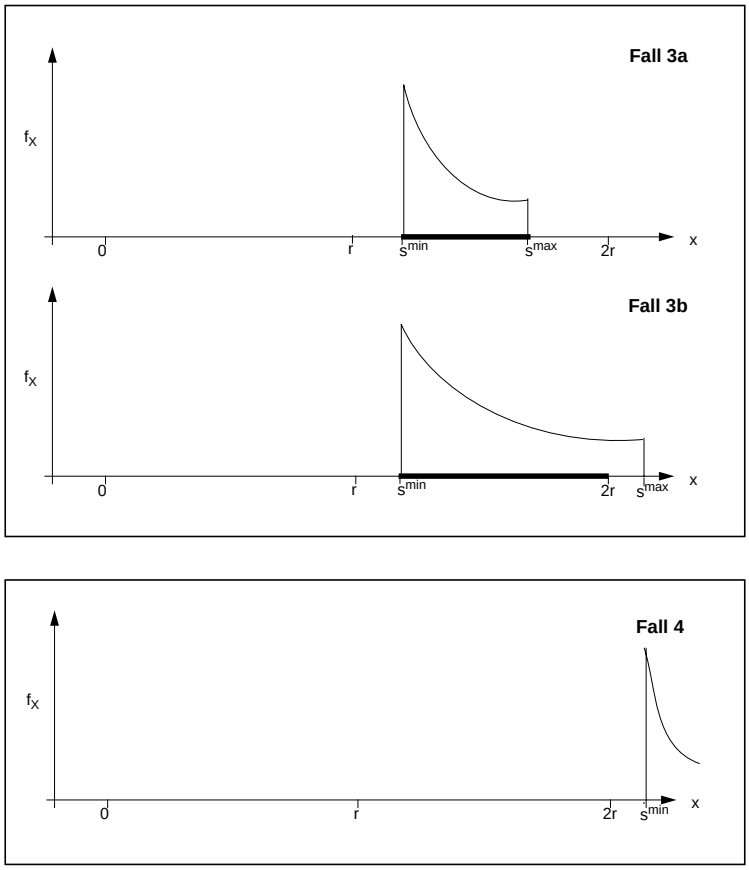

Abbildung 4.10: Skizze zur Berechnung der Fortschrittsgeschwindigkeit (Fall 3 und 4): Die Beschreibung der Abbildung entspricht der von Abbildung 4.9.

Fall 2: $s^{\min} < r < s^{\max}$

Setzt man $f_X(x)$ aus Gleichung 4.15 in Gleichung 4.41 ein, erhält man:

$$\begin{aligned}
\varphi &= \frac{1}{2k\ln(2)}\left(\int_{\rho 2^{-k}}^{r}\frac{1}{x}\cdot x\,dx + \int_{r}^{y}\frac{1}{x}\cdot(2r \Leftrightarrow x)\,dx\right) \\
&= \frac{1}{2k\ln(2)}\left([1]_{\rho 2^{-k}}^{r} + [2r\ln(x) \Leftrightarrow x]_{r}^{y}\right) \\
&= \frac{1}{2k\ln(2)}\left(2r\left(1+\ln\left(\frac{y}{r}\right)\right) \Leftrightarrow \rho 2^{-k} \Leftrightarrow y\right)
\end{aligned}$$

mit $y = \min(\rho, 2r)$.

Fall 3: $r \leq s^{\min}$

Setzt man $f_X(x)$ aus Gleichung 4.15 in Gleichung 4.41 ein, ergibt sich:

$$\varphi = \frac{1}{2k\ln(2)}\left(\int_{\rho 2^{-k}}^{y}\frac{1}{x}\cdot(2r \Leftrightarrow x)\,dx\right)$$

$$= \frac{1}{2k\ln(2)} \left[2 \cdot x^0 \cdot \ln(x) \Leftrightarrow x\right]_{\rho 2^{-k}}^{y}$$

$$= \frac{1}{2k\ln(2)} \left(2r \ln\left(\frac{y}{\rho 2^{-k}}\right) \Leftrightarrow y + \rho 2^{-k}\right)$$

mit $y = \min(\rho, 2r)$.
∎

In Abbildung 4.11 ist die Fortschrittsgeschwindigkeit φ der BGA-Mutationsverteilung in Abhängigkeit des Schrittweitenparameters ρ dargestellt.

Satz 10 *Die Fortschrittsgeschwindigkeit eines Mutations-Selektions-Verfahrens, dessen Mutationsschritte gleichverteilt im Intervall* $[\Leftrightarrow\rho, \rho]$ *generiert werden, ist:*

$$\varphi = \begin{cases} \frac{\rho}{4} & : \quad s^{\max} \leq r \\ \frac{1}{2\rho}\left(\Leftrightarrow r^2 + 2ry \Leftrightarrow \frac{1}{2}y^2\right) & : \quad s^{\min} < r < s^{\max} \end{cases} \tag{4.45}$$

mit $y = \min(\rho, 2r)$.

Beweis 10 *Die Dichtefunktion der gleichverteilten Zufallszahl* $X \sim \mathbf{U}([0, \rho])$ *ist:*

$$f_X(x) = \frac{1}{\rho} \tag{4.46}$$

Ihre minimale und maximale Schrittweite ist:

$$s^{\min} = 0 \tag{4.47}$$

$$s^{\max} = \rho \tag{4.48}$$

Setzt man obige Dichtefunktion $f_X(x)$ *in die Gleichung 4.41 ein, erhält man die gesuchte Fortschrittsgeschwindigkeit. Fall 3 entfällt, da die minimale Schrittweite Null ist.*

Fall 1: $s^{\max} \leq r$

$$\varphi = \frac{1}{2} \cdot \int_0^{\rho} \frac{1}{\rho} x \, dx$$

$$= \frac{1}{2\rho}\left(\frac{1}{2}\rho^2\right)$$

$$= \frac{\rho}{4}$$

Fall 2: $s^{\min} < r < s^{\max}$

$$\begin{aligned}
\varphi &= \frac{1}{2}\left(\int_0^r \frac{1}{\rho}x\,dx + \int_r^y \frac{1}{\rho}(2r \Leftrightarrow x)\,dx\right) \\
&= \frac{1}{2\rho}\left(\left[\frac{1}{2}x^2\right]_0^r + \left[2rx \Leftrightarrow \frac{1}{2}x^2\right]_r^y\right) \\
&= \frac{1}{2\rho}\left(\frac{1}{2}r^2 + 2ry \Leftrightarrow \frac{1}{2}y^2 \Leftrightarrow 2r^2 + \frac{1}{2}r^2)\right) \\
&= \frac{1}{2\rho}\left(\Leftrightarrow r^2 + 2ry \Leftrightarrow \frac{1}{2}y^2\right)
\end{aligned}$$

mit $y = \min(\rho, 2r)$.
∎

Satz 11 *Die Fortschrittsgeschwindigkeit eines Mutations-Selektions-Verfahrens, dessen Mutationsschritte normalverteilt mit Erwartungswert* $\mu = 0$ *und Standardabweichung* σ *generiert werden, ist*

$$\boxed{\varphi = \frac{1}{\sqrt{2\pi}\sigma} \cdot \left(\int_0^r \exp\left(\Leftrightarrow\frac{x^2}{2\sigma^2}\right) \cdot x\,dx + \int_r^{2r} \exp\left(\Leftrightarrow\frac{x^2}{2\sigma^2}\right) \cdot (2r \Leftrightarrow x)\,dx\right)} \tag{4.49}$$

Beweis 11 *Da die Normalverteilung nicht beschränkt ist, muß nur der Fall 2 aus Gleichung 4.41 betrachtet werden. Da die Dichtefunktion der Mutationsverteilung in dieser Gleichung nur über den positiven Bereich läuft, muß zunächst die Dichtefunktion der linkseitig zu 50 % gestutzten Normalverteilung* $\tilde{\mathbf{N}}(0, \sigma)$ *gebildet werden:*

$$f_X(x) = \frac{2}{\sqrt{2\pi}\sigma} \exp\left(\Leftrightarrow\frac{x^2}{2\sigma^2}\right) \qquad 0 \le x < \infty \tag{4.50}$$

Setzt man diese asymmetrische Dichtefunktion in Gleichung 4.41 ein, erhält man gesuchte Gleichung. ∎

Dieser Ausdruck kann nun ausintegriert und durch eine Taylorreihenentwicklung approximiert oder durch numerische Integration ausgewertet werden. In Abbildung 4.11 sind die durch numerische Integration bestimmten Werte der Fortschrittsgeschwindigkeit dargestellt. Rechenberg berechnet die Fortschrittsgeschwindigkeit für das Kugelmodell für hohe Dimensionen ($n >> 1$) ([Rec73], S.115–120):

$$\varphi = \frac{\sigma}{\sqrt{2\pi}}\left(e^{-(\frac{n\sigma}{\sqrt{8}r})^2} \Leftrightarrow \sqrt{\pi}\frac{n\sigma}{\sqrt{8}r}\left(1 \Leftrightarrow \Phi\left(\frac{n\sigma}{\sqrt{8}r}\right)\right)\right) \tag{4.51}$$

wobei $r = \delta(\vec{x}^0)$ den Fehler des Startpunkts $\vec{x}^0$ bezeichnet.

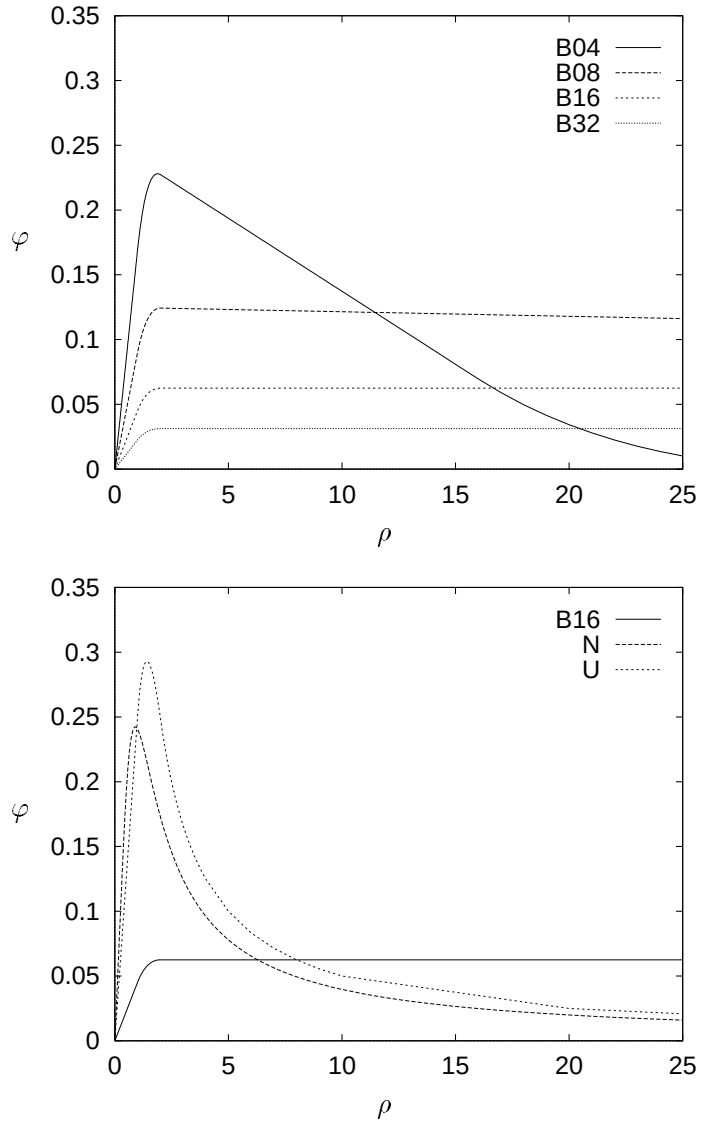

Abbildung 4.11: Fortschrittsgeschwindigkeit in Abhängigkeit vom Schrittweitenparameter ρ für die verschiedenen Mutationsverteilungen bei der eindimensionalen Zielfunktion $f(x) = |x|$ und Startpunkt $x^0 = 1.0$.

Für die eindimensionale Zielfunktion $f(x) = |x|$ und den Startpunkt $x^0 = 1.0$ sind die optimalen Fortschrittsgeschwindigkeiten der verschiedenen Mutationsverteilungen in Tabelle 4.5 aufgeführt. Abbildung 4.11 zeigt die Abhängigkeit der Fortschrittsgeschwindigkeit von dem Schrittweitenparameter ρ für verschiedene Mutationsverteilungen.

Die BGA-Mutationsverteilungen erzielen die maximale Fortschrittsgeschwindigkeit bei einem Schrittweitenparameter $\rho = 2.0$. Eine Verkleinerung von ρ führt zu einer stärkeren Verschlechterung als eine gleichgroße Vergrößerung. Eine Vergrößerung von k reduziert zwar den optimalen Wert der Fortschrittsgeschwindigkeit, trägt auf der anderen Seite aber zu einem robusteren Verhalten im Hinblick auf zu große Werte des Schrittweitenparameters ρ bei. Die anderen Mutationsverteilungen, die gegenüber den BGA-Mutationsverteilungen eine höhere maximale Fortschrittsgeschwindigkeit aufweisen, unterscheiden sich nur wenig

voneinander. Sie reagieren jedoch äußerst empfindlich auf Abweichungen vom optimalen Wert des Schrittweitenparameters ρ.

4.4.4 Robustheit

Einen ersten Eindruck zur Abhängigkeit der Fortschrittsgeschwindigkeit φ von Mutationsverteilung und Schrittweitenparameter ρ vermittelt Abbildung 4.11. Um diese Abhängigkeit und damit die Robustheit der Mutationsverteilungen gegenüber dem Schrittweitenparameter zu quantifizieren, werden, basierend auf der Fortschrittsgeschwindigkeit, spezielle Robustheitsmaße eingeführt.

Während Effizienzbetrachtungen von Mutationsverteilungen, wie sie in Abschnitt 4.4.3 durchgeführt wurden, nach der maximalen Fortschrittsgeschwindigkeit φ^* fragen, steht hier der funktionale Zusammenhang zwischen Fortschrittsgeschwindigkeit und Schrittweitenparameter im Vordergrund. Die schematische Abbildung 4.12, die ebenso wie die Abbildung 4.11 auf Seite 82 die Entwicklung der Fortschrittsgeschwindigkeit über dem Schrittweitenparameter zeigt, dient zur Beschreibung der Robustheitsmaße.

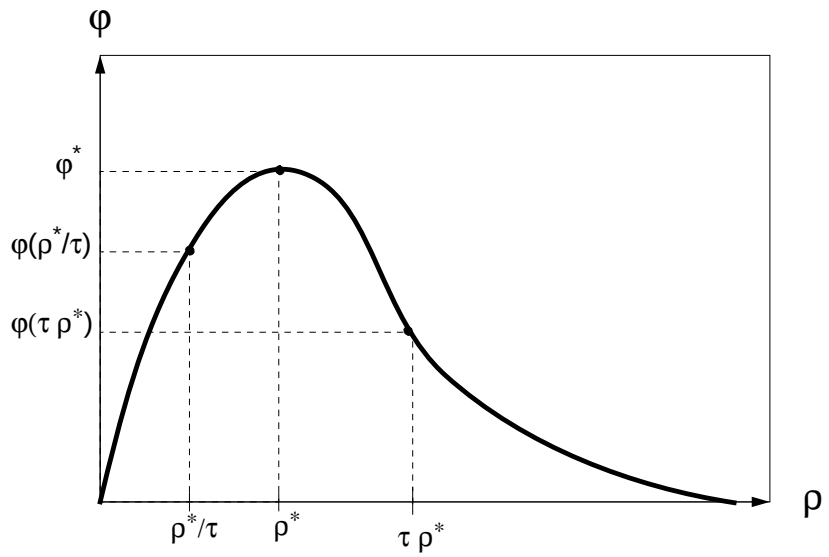

Abbildung 4.12: Schema zur Beschreibung der Robustheitsmaße. Aus dem funktionalen Zusammenhang zwischen Fortschrittsgeschwindigkeit φ und Schrittweitenparameter ρ werden in Gleichung 4.52 die Robustheitsmaße $\beta_{1/\tau}$ und β_τ definiert.

Die beiden folgenden Maße beschreiben die Robustheit des Mutationsoperators gegenüber Abweichungen des Schrittweitenparameters ρ vom optimalen Schrittweitenparameter ρ^*. Die Maße $\beta_{1/\tau}$ und β_τ geben die prozentuale Veränderung der Fortschrittsgeschwindigkeit durch eine τ-fach zu kleine beziehungsweise zu große Schrittweite an.

$$\beta_{1/\tau} = \frac{\varphi\left(\frac{1}{\tau}\cdot\rho^*\right)}{\varphi(\rho^*)}\cdot 100 \qquad \beta_\tau = \frac{\varphi\left(\tau\cdot\rho^*\right)}{\varphi(\rho^*)}\cdot 100 \tag{4.52}$$

MV	Effizienz		Robustheit			
	φ^*	ρ^*	$\beta_{1/10}$	$\beta_{1/2}$	β_2	β_{10}
B01	0.419	1.3	11.3	58.7	23.7	-
B02	0.341	1.65	12.0	63.8	62.4	-
B04	0.229	1.875	13.0	70.4	91.1	19.0
B08	0.125	2.0	14.5	72.3	99.4	94.9
B16	0.063	2.0	14.4	72.1	100.0	100.0
B32	0.032	2.0	14.4	72.1	100.0	100.0
N	0.242	0.906	14.9	72.8	76.7	18.0
U	0.293	1.414	12.1	60.4	60.4	12.1

Tabelle 4.5: Effizienz- und Robustheitsmaße für verschiedene Mutationsverteilungen (MV).

In Abbildung 4.13 sind die in Tabelle 4.5 aufgeführten Effizienz- und Robustheitsmaße dargestellt. Die beiden oberen Graphiken zeigen die optimale Fortschrittsgeschwindigkeit φ^* mit dazugehörigem Schrittweitenparameter ρ^* der verschiedenen Mutationsverteilungen. Die anderen Graphiken enthalten die Robustheitsmaße $\beta_{1/\tau}$ und β_τ aus Gleichung 4.52 für $\tau = 2$ und $\tau = 10$.

Die in Abbildung 4.13 und Tabelle 4.5 dargestellten Maße machen den Tradeoff zwischen maximaler Effizienz und Robustheit bezüglich des Schrittweitenparameters ρ deutlich. Je höher die maximale Fortschrittsgeschwindigkeit φ^* ist, umso empfindlicher reagiert das Mutations-Selektions-Verfahren auf Abweichungen vom optimalen Schrittweitenparameter ρ^*.

Der Einfluß des Präzisionsparameters k der BGA-Mutationsverteilung $\mathbf{B}(k, \rho)$ wird durch die Leistungsmaße verdeutlicht. Die größte maximale Fortschrittsgeschwindigkeit φ^* wird mit dem Präzisionsparameter $k = 1$ erzielt. Mit wachsendem k sinkt die maximale Fortschrittsgeschwindigkeit bei steigender Robustheit. Wie die β-Robustheitsmaße in Tabelle 4.5 zeigen, reagiert die BGA-Mutationsverteilung mit einem Präzisionsparameter $k \geq 8$ äußerst robust auf Abweichung des Schrittweitenparameters ρ nach oben.

Selbst ein 10-fach zu großer Schrittweitenparameter führt bei einem Präzisionsparameter $k = 8$ zu einer Fortschrittsgeschwindigkeit, die nur 5 % unter der maximal zu erreichenden liegt. Ab $k = 16$ bleibt die Fortschrittsgeschwindigkeit fast unverändert. Auf Abweichungen vom optimalen Schrittweitenparameter nach unten reagiert die BGA-Mutationsverteilung dagegen empfindlicher.

Die Untersuchungen der BGA-Mutationsverteilung haben die Unterschiede zu anderen in Evolutionären Algorithmen häufig verwendeten Mutationsverteilungen deutlich gemacht. Der wesentlichste Aspekt ist die Robustheit gegenüber dem Schrittweitenparameter. Aufgrund dieser Robustheit kann auf eine präzise Steuerung des Schrittweitenparameters, wie sie die Normalverteilung und Gleichverteilung benötigen, verzichtet werden.

Da eine Abweichung des Schrittweitenparameters vom optimalen Schrittweitenparameter nach oben kaum zu wesentlich geringeren Effizienzeinbußen führt als eine Abweichung nach unten, sollte der Schrittweitenparameter in Abhängigkeit von k eher groß gewählt werden. Die optimale Fortschrittsgeschwindigkeit φ^* und der Präzisionsparame-

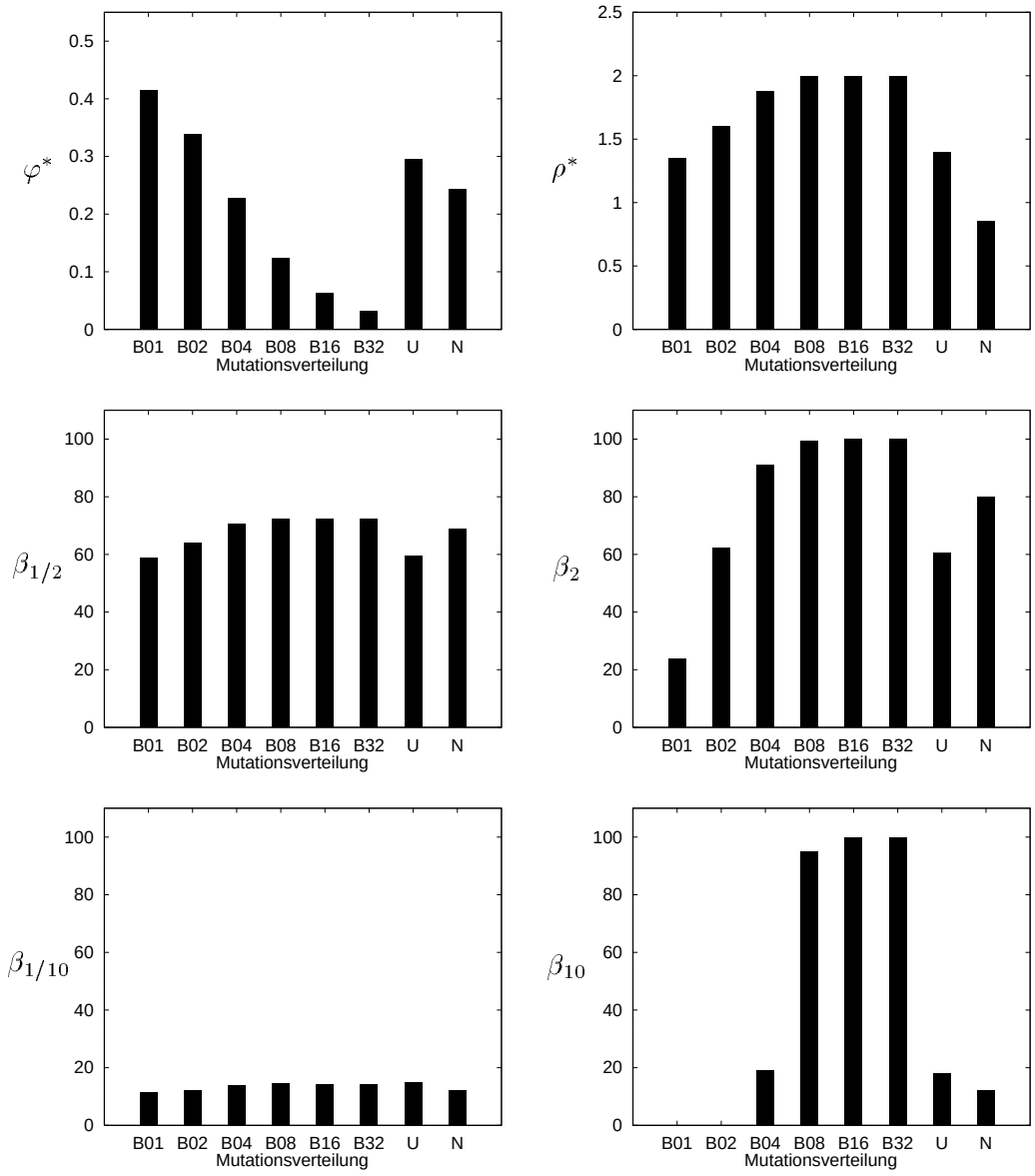

Abbildung 4.13: Leistungsmaße für die verschiedenen Mutationsverteilungen. Dargestellt sind die optimale Fortschrittsgeschwindigkeit φ^* und der dazugehörige Schrittweitenparameter ρ^* der Mutationsverteilungen, sowie die Robustheitsmaße $\beta_{1/2}$, β_2, $\beta_{1/10}$ und β_{10}.

ter k verhalten sich für $k \geq 8$ umgekehrt proportional zueinander:

$$\varphi^* \sim \frac{1}{k} \tag{4.53}$$

Der Präzisionsparameter k ist ein wichtiger Regelparameter der BGA-Mutationsverteilung, der die gegenläufigen Größen Effizienz und Robustheit festlegt. Zur Steigerung der Effizienz der BGA-Mutationsverteilung bietet sich die Wahl eines kleinen Präzisionsparameters k an. Der damit verbundene Robustheitsverlust kann durch eine grobe Steuerung des Schrittweitenparameters ρ kompensiert werden. Zu einer solchen Steuerung können die im nächsten Kapitel eingeführten Wettbewerbsmodelle verwendet werden.

Kapitel 5

Wettbewerbsmodelle zur Strategieanpassung

5.1 Begriff der Suchstrategie

Die Methode, mit der ein Suchverfahren neue Punkte im Suchraum generiert, wird als dessen Suchstrategie bezeichnet. Viele Suchverfahren modifizieren im Laufe der Suche bestimmte Faktoren ihrer Suchstrategie und passen sie so an die ständig wechselnden Gegebenheiten an.

Die Suchstrategie eines Evolutionären Algorithmus besteht aus der Kombination aller *variablen* Faktoren des Evolutionären Algorithmus aus Definition 1. Verschiedene EA unterscheiden sich lediglich in der Repräsentation ihrer Individuen sowie in der Ausprägung der einzelnen Phasen, also in der Art, wie Eltern zusammengestellt und aus ihnen neue Individuen gebildet werden.

Definition 14 (Suchstrategie) *Die* Suchstrategie ($\mathcal{S}$) *eines Evolutionären Algorithmus setzt sich aus der Kodierung und der Menge aller Operatoren und ihrer Parameter zusammen.*

$$\mathcal{S} = \{\mathbb{C}_{\Theta_{\mathbb{C}}}, \mathbb{M}_{\Theta_{\mathbb{M}}}, \mathbb{R}_{\Theta_{\mathbb{R}}}, \mathbb{S}_{\Theta_{\mathbb{S}}}, \mathbb{P}_{\Theta_{\mathbb{P}}}, \mathbb{K}_{\Theta_{\mathbb{K}}}\} \tag{5.1}$$

Beim Entwurf eines EA stellt sich die Frage nach der *optimalen* Suchstrategie, also der optimalen Kombination aller Operatoren und ihrer Parameter. Eine allgemeine Beantwortung dieser Frage ist jedoch nicht möglich, da die Güte eines Suchverfahrens sowohl von den Eigenschaften des zu lösenden Problems als auch von der Zielsetzung des Optimierers abhängt. Bei der Festlegung der Zielsetzung eines Optimierungsverfahrens müssen die gegenläufigen Kriterien Effizienz und Robustheit nach den individuellen Bedürfnissen gewichtet werden.

Da die Eigenschaften realer Probleme in der Regel nicht bekannt sind, können selbst bei statischen Problemen a priori kaum Angaben über die Güte einer bestimmten Suchstrategie gemacht werden. Hinzu kommt, daß sich die Suchsituation während der Suche infolge der neu generierten Punkte ständig verändert. Eine Strategie, die am Anfang der Suche gute Beiträge liefert, kann zu einem späteren Zeitpunkt einbrechen. Ein Beispiel ist die Schrittweite von Mutationsoperatoren. Sind die Suchpunkte weit vom Optimum entfernt, führen große Schrittweiten anfangs zu einem besseren Fortschritt als am Ende der Suche, wenn sich die Suchpunkte dem Optimum angenähert haben.

Weitaus schwieriger gestaltet sich die Beantwortung der Frage nach der optimalen Strategie bei sogenannten dynamischen Problemstellungen. Da sich hier die Lage des Optimums während der Suche ständig ändert, können über die Güte einer Suchstrategie kaum Aussagen getroffen werden.

Wegen der einem Suchprozeß anhaftenden Dynamik und des Informationsdefizits über die Struktur des Problems erscheint die Verwendung einer einzigen statischen Suchstrategie als nicht ausreichend. So findet man in praktisch allen Optimierungsverfahren dynamische Komponenten, die die Schrittweite und Richtung während der Suche an die aktuelle Situation anpassen. Aus den vergangenen Suchschritten werden Informationen zur Schätzung guter Schrittweiten und Suchrichtungen, also guter Suchstrategien, gewonnen.

Die Anpassung der Suchstrategie ist folglich ein grundlegender Bestandteil eines Optimierungsverfahrens, das flexibel auf die jeweilige Problemstellung und auf die Zielsetzung des Anwenders eingehen soll.

5.2 Ebenen der Strategieanpassung

Die Anpassung der Suchstrategie eines Evolutionären Algorithmus ist nicht auf die Anpassung der externen Parameter der genetischen Operatoren beschränkt. Vielmehr können alle in Definition 14 enthaltenen Faktoren Gegenstand der Anpassung sein. Ein EA kann im Rahmen der Strategieanpassung die genetischen Operatoren ebenso wie deren Parameter variieren.

Analog zu den Mechanismen der natürlichen Evolution kann die Anpassung der Suchstrategie in Evolutionären Algorithmen auf verschiedenen Ebenen vorgenommen werden. Die Strategieanpassung kann auf Individuen- oder Populationsebene erfolgen. Die Wahl der Ebene legt die Möglichkeiten der Anpassung fest. Auf Individuenebene können alle Komponenten, die ein einzelnes Individuum betreffen, Bestandteil der Anpassung sein. Zu diesen Komponenten zählen die Kodierung sowie die genetischen Operatoren und ihre Parameter. Auf der Populationsebene können alle Komponenten der Suchstrategie angepaßt werden.

Beide Ebenen der Strategieanpassung wurden bereits erfolgreich in EA eingesetzt. Das bekannteste Verfahren auf Individuenebene ist die in Abschnitt 3.3 vorgestellte Evolutionsstrategie von Schwefel, deren Individuen mittlere Schrittweite und Orientierung der Mutation enthalten.

5.2.1 Strategieanpassung auf Individuenebene

Strategieparameter in der Evolutionsstrategie

Die Evolutionsstrategie wurde bereits in Abschnitt 3.3 ausführlich vorgestellt. An dieser Stelle soll die Grundidee der von Schwefel eingeführten Strategieparameter in Bezug auf die Strategieanpassung noch einmal erläutert werden ([Sch77]). Jedes Individuum der Evolutionsstrategie enthält zusätzlich zu den zu optimierenden Objektparametern sogenannte Strategieparameter, die Schrittweiten und ggf. Korrelationen des Mutationsoperators beeinflußen. Diese Strategieparameter werden ebenfalls durch genetische Operatoren modifiziert und implizit der Selektion unterworfen. Da die Selektion von Individuen aufgrund der Fitnesswerte, die durch ihre Objektparameter erzielt werden, erfolgt, ist die Selektion von Strategieparametern nur mittelbar. Die Grundidee dieses Verfahrens besteht darin, daß

gute Strategieparameter zu guten Objektparametern führen und sich so in der Population durchsetzen.

Indem die Evolution Schrittweiten und Richtungen an die aktuelle Situation anpaßt, baut sie ein internes Modell der lokalen Fitnesslandschaft auf. Wie Schwefel gezeigt hat, handelt es sich um einen kollektiven Lernprozeß, d.h. es entsteht kein Superindividuum, sondern die Population als Ganzes agiert 'intelligent' ([Sch87]).

Kontrollparameter in Genetischen Algorithmen

Bäck überträgt das Konzept der Strategieanpassung in ES durch Strategieparameter auf Genetische Algorithmen ([Bäc92]). In diesem Ansatz wird die Mutationswahrscheinlichkeit des GA, die im traditionellen GA durch einen externen Parameter spezifiziert wird, während der Suche durch den Algorithmus selbst angepaßt.

Dabei wird eine Mutationsrate für alle Bits des Individuums oder eine Mutationsrate pro Bitposition als Kontrollparameter in das Individuum aufgenommen. Dieser Kontrollparameter wird ebenso wie die Strategieparameter in ES durch die genetischen Operatoren modifiziert und anschließend für die Mutation der Objektparameter verwendet.

Bäck erzielt mit dieser Art der Selbstanpassung für Probleme der kontinuierlichen Parameteroptimierung erhebliche Verbesserungen gegenüber einem GA mit konstanter Mutationsrate. Es zeigt sich, daß die Verwendung einer globalen Mutationsrate besonders für unimodale Fitnessfunktionen vorteilhaft ist, während die Verwendung einer Mutationsrate pro Objektparameter die Exploration betont und so für die Optimierung multimodaler Fitnessfunktionen günstiger ist.

5.2.2 Strategieanpassung auf Populationsebene

Die Idee der Strategieanpassung auf Populationsebene besteht darin, den eigentlichen EA als Individuum eines übergeordneten EA zu betrachten. Die Aufgabe des äußeren EAs besteht in der Optimierung der externen Parameter des inneren EA. Der äußere EA paßt damit die Suchstrategie des inneren EA an die momentane Situation an. Dieses Konzept, das die Evolution rekursiv auf sich selbst anwendet, wird auch als *Meta-Evolution* bezeichnet.

Die Schachtelungstiefe kann konzeptionell beliebig sein. Grenzen sind dieser Art der Strategieanpassung nur durch den Berechnungsaufwand gesetzt, der mit jeder Ebene sehr stark anwächst. So entspricht die Fitnessfunktion der zweiten Hierarchieebene bereits der Ausführung des EA auf der Ebene darunter.

Geschachtelte Evolutionsstrategien

Das Konzept der geschachtelten Evolutionsstrategie wurde in seiner allgemeinen Form von Rechenberg eingeführt ([Rec94]). Rechenberg wurde durch in der natürlichen Evolution beobachtbare Phänomene, die nicht den Nutzen des einzelnen Individuums, sondern den der gesamten Population berücksichtigen, dazu inspiriert, das Individuum als alleinige Selektionseinheit in Frage zu stellen. So kann beispielsweise die Herausbildung altruistischer Verhaltensweisen nicht durch die Selektion einzelner Individuen erklärt werden.

Die Konsequenz aus dieser Überlegung führt zu der geschachtelten Evolutionsstrategie, in der nicht nur Individuen, sondern auch Populationen Bestandteil der Selektion sind. Dieses Prinzip läßt sich, wie oben bereits geschildert, beliebig fortsetzen.

Die Festlegung der Fitnessfunktion und der genetischen Operatoren der höheren Hierarchieebenen stellt den Anwender vor keine leichte Aufgabe. Rechenberg schlägt vor, die Individuenqualität und Populationsqualität unabhängig voneinander zu wählen, um die Dominierung der einen Zielfunktion durch die andere zu verhindern. Ein Beispiel für den Einsatz der geschachtelten Evolutionsstrategie ist die Anpassung der Strategieparameter auf Populationsebene. In diesem Fall wird auf Populationsebene nach der Fortschrittsgeschwindigkeit und auf der Individuenebene, wie üblich, nach dem Zielfunktionswert selektiert. Die Mutation wird auf der Populationsebene auf die Strategieparameter angewandt.

Herdy wendet das Konzept der geschachtelten Evolutionsstrategie mit drei Hierarchieebenen zur Adaptation der Isolationsdauer einer Art an ([Her92]). Auf der zweiten und dritten Hierarchieebene wird die Schrittweite einer Population bzw. die Isolationsdauer zwischen den Populationen angepaßt.

5.3 Exkurs: Populationsdynamik

In dieser Arbeit werden Wettbewerbsmodelle zur Strategie- und Populationsgrößenanpassung von Evolutionären Algorithmen eingeführt. Diese Wettbewerbsmodelle orientieren sich an den in der natürlichen Evolution zu beobachtenden Interaktionen von verschiedenen Arten, die um eine beschränkte Ressource konkurrieren. Da die Untersuchung dieser Interaktionen Gegenstand des biologischen Forschungzweiges *Populationsdynamik* ist, werden in diesem Abschnitt die wichtigsten Modelle und Begriffe vorgestellt. Aus den biologischen Modellen lassen sich Anregungen und Ideen zum Entwurf neuer Konzepte für den Bereich der Evolutionären Algorithmen gewinnen.

Die Populationsdynamik, die sich aus den Forschungsbereichen Populationsgenetik und evolutionäre Ökologie entwickelte, untersucht Interaktionen zwischen verschiedenen Arten und Populationen. Die zentrale Fragestellung der Populationsdynamik ist die Entwicklung von Populationsgrößen unter bestimmten Randbedingungen, wie Umwelteinflüsse und Koexistenz mehrerer Populationen.

Die zur Beantwortung dieser Frage entwickelten Modelle sind zwar mathematisch handhabbar, aufgrund starker Vereinfachung aber oft ungenau. Ein klassisches Modell, das die Entwicklung einer einzelner Population ohne Altersstruktur beschreibt, ist die *logistische Gleichung*. Interaktionen zwischen verschieden Arten, wie Wettbewerb und Räuber-Beute-Beziehungen, lassen sich durch die berühmten *Lotka-Volterra-Gleichungen* modellieren.

Die einfachen Modelle der Populationsdynamik setzen, wie auch das Generationenmodell in Evolutionären Algorithmen, Populationen ohne Altersstruktur voraus.

5.3.1 Basismodelle

Die Basismodelle der Populationsdynamik beschreiben die Entwicklung der Populationsgröße unter bestimmten Randbedingungen. Die zukünftige Populationsgröße, die auch als *Dichte* der Population aufgefaßt werden kann, ist von der derzeitigen Populationsgröße abhängig.

Setzt man diskrete Zeitintervalle voraus, wird die Entwicklung der Populationsgröße $N(t)$ durch folgende Gleichung beschrieben:

$$N(t+1) = (1 + R(t)) \cdot N(t) \tag{5.2}$$

Durch die Verwendung von Zeitintervallen infinitesimaler Länge, kann obige Gleichung kontinuisiert werden:

$$\frac{dN(t)}{dt} = r(t) \cdot N(t) \tag{5.3}$$

$R(t)$ und $r(t)$ werden als die *Wachstumsrate* der Population bezeichnet. Die Entscheidung, ob ein zeitdiskretes oder kontinuierliches Modell zu verwenden ist, hängt von dem zu beschreibenden Phänomen ab. Das kontinuierliche Modell eignet sich zur Beschreibung von Populationen, die mehr oder weniger gleichmäßig über die Zeit Nachkommen erzeugen, während das diskrete Modell verwendet wird, falls Populationen in großen Zeitintervallen innerhalb kurzer Perioden Nachkommen erzeugen. Evolutionäre Algorithmen, die auf dem Generationenmodell basieren, lassen sich am geeignetsten durch zeitdiskrete Modelle beschreiben ([May73]).

Die logistische Gleichung

Die logistische Gleichung ist ein einfaches Modell, das das Wachstum einer Population in Abhängigkeit von Umwelteinflüssen beschreibt. Die Wachstumsrate, im folgenden als $r(t)$ bezeichnet, ist dadurch charakterisiert, daß sie mit steigender Populationsgröße immer schwächer ansteigt und ab einer bestimmten Populationsgröße sogar abfällt. Folgende allgemeine Gleichung genügt diesem Sachverhalt:

$$r(t) = r_0 \left(1 \Leftrightarrow \frac{N(t)}{K}\right) \tag{5.4}$$

Die Größe r_0, die die maximale Wachstumsrate bezeichnet, hängt von Umwelteinflüssen und Eigenschaften der Art ab. Die Größe K kann als die maximale Anzahl von Individuen betrachtet werden, die die Umwelt zuläßt. Der Umweltfaktor, der die Populationsgröße limitiert, wird als *begrenzender Faktor* (engl.: *limiting factor*) oder als *Kapazität* (engl.: *carrying capacity*) bezeichnet. Beispiele für begrenzende Faktoren sind Nahrung und Raum. Setzt man Gleichung 5.4 in Gleichung 5.3 ein, ergibt sich die *logistische Gleichung*:

$$\frac{dN(t)}{dt} = r_0 N(t) \left(1 \Leftrightarrow \frac{N(t)}{K}\right) \tag{5.5}$$

Aus der logistischen Gleichung kann eine Formel für die Entwicklung der Populationsgröße aufgrund der initialen Populationsgröße $N(0)$ und der maximalen Wachstumsrate r_0 abgeleitet werden ([Eml84]):

$$N(t) = N(0) \frac{K}{(K \Leftrightarrow N(0)) \cdot \exp(\Leftrightarrow r_0 t + N(0))} \tag{5.6}$$

Abbildung 5.1 zeigt die durch die logistische Gleichung berechnete Populationsentwicklung für unterschiedliche initiale Populationsgrößen und Wachstumsraten. Alle drei Kombinationen konvergieren gegen die Kapazität $K = 100$.

Die Dichterückkopplung

Die logistische Gleichung enthält einen sogenannten *Dichterückkopplungsterm* (engl.: *density feedback*), der die Wachstumrate $r(t)$ in Abhängigkeit der Populationsgröße $N(t)$ regelt. Der Dichterückkopplungsterm der logistischen Gleichung

$$r(t) = \frac{1}{N(t)} \frac{dN(t)}{dt} = r_0 \left(1 \Leftrightarrow \frac{N(t)}{K}\right) = r_0 \Leftrightarrow \frac{r_0}{K} N(t) \tag{5.7}$$

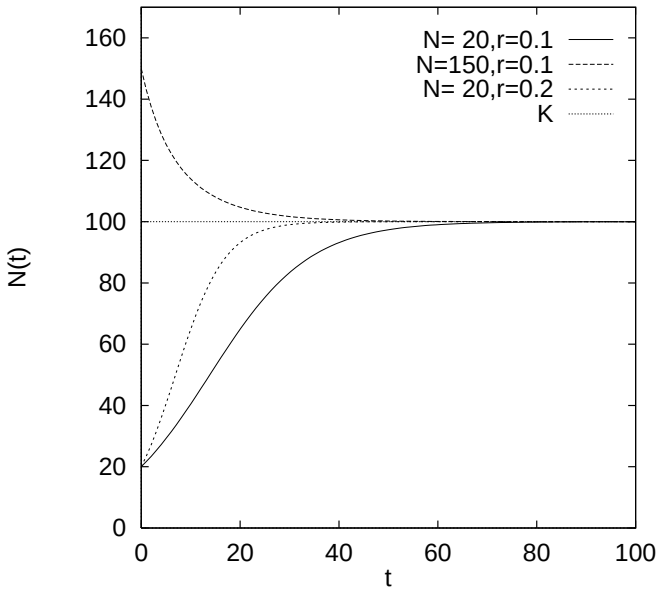

Abbildung 5.1: Populationsentwicklung aufgrund der logistischen Gleichung für unterschiedliche initiale Populationsgrößen N und Wachstumsraten r.

ist r_0/K. Die Hinzunahme eines Individuums reduziert die Wachstumsrate $r(t)$ um den Betrag r_0/K. Daraus folgt, daß eine Population unter der Annahme einer beschränkten Ressource nicht beliebig wachsen kann. Sobald eine Population eine bestimmte Größe erreicht hat, sinkt die Geburtenrate bzw. steigt die Todesrate.

Die bekanntesten Ressourcen, deren Mangel zu einem Wachstumsrückgang der Population führt, sind *Nahrung* und *Lebensraum*. Aber auch Streß oder eine zu große Räuberpopulation, deren Wachstum wiederum von der Größe der Beutepopulation anhängt, führt zu einem Wachstumsrückgang der Beutepopulation. Die Populationen werden in Abhängigkeit von dem begrenzenden Faktor als *nahrungs-*, *lebensraum-*, *streß-* oder *räuberlimitiert* bezeichnet. In der Regel ist eine Population aber nicht durch einen einzigen Faktor, sondern durch verschiedene Faktoren limitiert.

Die Aussagekraft der logistischen Gleichung wird durch die strengen Annahmen stark eingeschränkt. Die logistische Gleichung geht von einer konstanten Ressource und deren linearem Einfluß auf die Populationsgröße aus. Beide Voraussetzungen hängen von der jeweiligen Ressource ab. Im Gegensatz zur konstanten Ressource Lebensraum wird das Vorkommen der Ressource Nahrung wiederum durch die Populationsgröße beeinflußt. Daraus ist ersichtlich, daß die Modellierung der Ressourcenlimitierung und deren Auswirkung auf das Populationswachstum wesentlich komplexer sein müßte als die einfache logistische Gleichung suggeriert.

Umweltveränderungen

Die Umwelt, in der sich Populationen befinden, ist ständigen Veränderungen unterworfen. Das Nahrungsangebot verändert sich im Laufe der Zeit ebenso wie die Räuberpopulation und das Klima. Umweltveränderungen lassen sich in *dichteabhängige* (z.B. Nahrung) und *dichteunabhängige* (z.B. Klima) klassifizieren.

Da nicht nur die Umwelt, sondern auch die darin lebenden Populationen einem ständi-

gen Veränderungsprozeß unterworfen sind, stellt sich die Frage, ob sich eine Population überhaupt in einem Gleichgewicht befinden kann. Die Beantwortung dieser Frage hängt von der Stärke der Umweltveränderung und der Sensitivität der Population ab. Große Populationen, die über eine große Mobilität und physiologische Elastizität verfügen, können sich an Umweltveränderungen anpassen und ihre Größe mehr oder weniger konstant halten. So weisen z.B. Wirbeltiere gegenüber Insekten wesentlich stabilere Populationsdichten auf.

Stabilität von Populationen

Die Populationsdynamik unterscheidet verschiedene Arten von Stabilität. Eine Population wird als *nachbarschaftsstabil* (engl.: *neigborhood stable*) bezeichnet, falls es eine Gleichgewichtspopulationsgröße gibt, die die Population nach kleinen Störungen wieder annimmt. Eine Population heißt *global stabil*, falls sie nach jeder beliebigen Störung wieder die Gleichgewichtspopulationsgröße annimmt. Sie heißt *geschützt* (engl.: *protected*), falls ihre Wachstumsrate für eine kleine Populationsgröße stets positiv ist und so ihr Aussterben verhindert.

Räumlich strukturierte Populationen

Individuen einer Population sind nicht, wie in den einfachen Modellen angenommen, zufällig in ihrem Lebensraum angeordnet, sondern bilden Konzentrationen. Populationen sind im allgemeinen also keine kontinuierlichen Einheiten mit frei interagierenden Mitgliedern, sondern bestehen aus räumlich strukturierten, teilweise verbundenen Subpopulationen. Die Populationsdynamik untersucht die Auswirkungen dieser räumlichen Strukturierung auf die Entwicklung der Population. Dazu zählen Fragen nach ihrem Einfluß auf die Aussterberate, die Adaptation an sich verändernde Umweltbedingungen und die Stabilität einer Population.

In der Natur zeigt sich, daß Populationen, die sich aus räumlich verteilten Subpopulationen zusammensetzen, durch den damit verbundenen Puffereffekt besser gegen Aussterben geschützt sind. Von diesem Puffereffekt profitieren im besonderen Arten wie Insekten, die eine hohe Reproduktionsrate und Migrationsfähigkeit besitzen. Eine lokale Population, die ausstirbt, kann durch die Neugründung einer anderen ersetzt werden ([Eml84]).

5.3.2 Räuber-Beute-Interaktionen

Räuber-Beute-Modelle beschreiben die Entwicklung zweier Populationen, von denen die eine die Beute, und damit das Nahrungsangebot der anderen ist. Beide Populationen sind eng miteinander verknüpft. Das Wachsen der Beute-Population ermöglicht aufgrund des größeren Nahrungsangebots das Wachstum der Räuber-Population. Das Wachsen der Räuber-Population reduziert wiederum die Beute-Population, was das Abnehmen der Räuber-Population zur Folge hat.

Das bekannteste Modell, das diese Beziehung beschreibt, ist das *Lotka-Volterra-Modell*. Dieses Modell geht davon aus, daß die Rate, mit der die Beute vom Räuber gefressen wird, proportional zur Begegnungsrate der beiden Arten ist. Im einfachsten Fall ist die Begegnungsrate durch die Dichten der beiden Populationen gegeben.

Das Lotka-Volterra-Räuber-Beute-Modell

Unter der Annahme, daß ein Beuteindividuum nur durch einen Räuber zu Tode kommt, gilt für die Entwicklung der Beute-Population folgende Gleichung.

$$\frac{dN_1}{dt} = b_1 N_1 \Leftrightarrow d_1 N_1 = b_1 N_1 \Leftrightarrow \delta N_1 N_2 = N_1(b_1 \Leftrightarrow \delta N_2) \tag{5.8}$$

wobei b_1 die Geburtenrate und δ eine Proportionalitätskonstante ist. Falls weiterhin die Geburtenrate der Räuberpopulation direkt (und linear) von der Nahrungsaufnahme pro Individuum abhängt, gilt:

$$\frac{dN_2}{dt} = b_2 N_2 \Leftrightarrow d_2 N_2 = \beta N_1 N_2 \Leftrightarrow d_2 N_2 = N_2(\beta N_1 \Leftrightarrow d_2) \tag{5.9}$$

wobei β ein Maß für die Umsetzung von Nahrung in Räubernachkommen ist. Die Gleichungen 5.8 und 5.9 werden nach ihren Entdeckern *Lotka-Volterra-Räuber-Beute-Gleichungen* genannt ([Lot25], [Vol25]). Das aus den Lotka-Volterra-Gleichungen resultierende Räuber-Beute-System führt unter bestimmten Bedingungen zu einem sich ständig wiederholenden Zyklus. Da die Lotka-Volterra-Gleichungen ebenso wie die logistische Gleichung von stark vereinfachten Annahmen ausgeht, die von der Realität abweichen, wurden für unterschiedliche Problemstellungen Korrekturen eingefügt. Ob in der Natur die durch die Lotka-Volterra-Gleichungen beschriebenen endogenen Zyklen auftreten, ist eine Kernfrage der Räuber-Beute-Problematik.

5.3.3 Wettbewerbs-Interaktionen

Eine wichtige Fragestellung der Populationsdynamik betrifft die Auswirkungen von konkurrierenden Interaktionen zwischen verschiedenen Populationen. Hier unterscheidet die Populationsdynamik verschiedene Wettbewerbsformen (engl.: *competition*).

Werden die Gleichgewichtsdichten zweier Arten von derselben Ressource beeinflußt, stehen beide Arten in einem *Ressourcen-Wettbewerb* (engl.: *resource competition*) oder *Ausnutzungswettbewerb* (engl.: *exploitation competition*). Die Ressourcenmenge, die die eine Art konsumiert, steht der anderen Art nicht mehr zur Verfügung. Das Wachsen der einen Population führt nicht nur zu einer negativen Rückkopplung der eigenen, sondern auch der anderen Wachstumsrate.

$$\frac{\delta r_i}{\delta N_i} < 0 \qquad \frac{\delta r_j}{\delta N_i} < 0 \tag{5.10}$$

Die Intensität des Wettbewerbs $\delta r_j / \delta N_i$ (und $\delta r_i / \delta N_j$) drückt aus, wie stark die Überlappung der Ressourcen zwischen den Arten i und j ist. Falls m_k die Verfügbarkeit der k-ten Ressource bezeichnet, kann die Intensität des Wettbewerbs von Art i auf Art j durch folgende Formel beschrieben werden:

$$\sum_k \frac{\delta r_i}{\delta m_k} \frac{\delta m_k}{\delta N_j} \tag{5.11}$$

Eine andere Form des Wettbewerbs ist der *Einmischungswettbewerb* (engl.: *interference competition*). Hierbei erfolgt der Wettbewerb nicht über eine von beiden konsumierte geteilte Ressource, sondern über einen dritten Organismus. Ein Beispiel sind zwei

räuberlimitierte Arten, die von derselben Räuberart gejagt werden. Das Anwachsen der einen Beuteart führt zum Wachsen der Räuberpopulation und somit zu einem negativen Wachstumsfaktor beider Beutearten.

Beide Wettbewerbsformen findet man in allen Bereichen der Ökologie. Besonders deutlich ist der Ressourcen-Wettbewerb in der Pflanzenwelt zu beobachten. Pflanzen konkurrieren um Sonnenlicht, Nährstoffe und Wasser. Eine Pflanzenart, die die vorhandenen Mineralien besser bzw. schneller umsetzt als eine andere, ist wettbewerbsfähiger und verdrängt andere.

Das Lotka-Volterra-Wettbewerbs-Modell

Die ersten Wettbewerbsmodelle wurden, wie auch die Räuber-Beute-Modelle, von Lotka und Volterra aufgestellt. Die logistische Gleichung für zwei konkurrierende Arten basiert auf der logistischen Gleichung einer einzelnen ressourcenlimitierten Art (Gleichung 5.7). Danach reduziert ein zusätzliches Individuum der Art i die Wachstumsrate r_i um den Faktor $1/K_i$.

Bei dem Wettbewerbsmodell wird die Wachstumsrate r_i nicht nur durch ein zusätzliches Individuum seiner eigenen Art, sondern ebenso durch ein zusätzliches Individuum der konkurrierenden Art j reduziert. Um die Verringerung der Wachstumsrate r_i durch ein zusätzliches Individuum einer anderen Art von der Hinzunahme eines Individuums der eigenen Art zu unterscheiden, wird ein Faktor α_{ij} eingeführt, der die relative Auswirkung von Art i und j auf r_i darstellt. Jedes zusätzliche Individuum der Art j reduziert die Wachstumsrate der Art i um den Faktor $(1/K_i)\alpha_{ij}$, so daß für r_i gilt:

$$r_i = \left(r_{i0} \Leftrightarrow \frac{r_{i0}}{K_i} N_i \Leftrightarrow \frac{r_{i0}}{K_i} \alpha_{ij} N_j\right) = r_{i0}\left(1 \Leftrightarrow \frac{N_i + \alpha_{ij} N_j}{K_i}\right) \tag{5.12}$$

Analog gilt für r_j:

$$r_j = r_{j0}\left(1 \Leftrightarrow \frac{N_j + \alpha_{ji} N_i}{K_j}\right) \tag{5.13}$$

Die Populationsgrößenentwicklung der Art i liefert uns folgende Gleichung:

$$\frac{dN_i}{dt} = r_{i0} \cdot N_i \left(1 \Leftrightarrow \frac{N_i + \alpha_{ij} N_j}{K_i}\right) \tag{5.14}$$

In Abbildung 5.2 ist eine Anwendung des Lotka-Volterra-Modells auf zwei konkurrierende Populationen dargestellt. Beide Populationen starten mit einer Populationsgröße von $N = 20$. Sie unterscheiden sich geringfügig in ihren Kapazitäten $K_1 = 95$ und $K_2 = 100$. Die Wachstumsrate der ersten ist erheblich größer als die der zweiten Population ($r_1 = 0.25, r_2 = 0.1$). Die Wettbewerbskoeffizienten sind $\alpha_{12} = 1.1$ und $\alpha_{21} = 0.9$. Aufgrund ihrer größeren Wachstumsrate gewinnt die erste Population anfangs schneller an Größe als ihr Konkurrent. Später wird sie aufgrund ungünstigerer Wettbewerbskoeffizienten von der zweiten verdrängt. Die dominierende Population strebt, wie bei der logistischen Gleichung für eine einzelne Population, gegen ihre Kapazität.

Das Lotka-Volterra-Wettbewerbsmodell ist durch das zugrundeliegende logistische Modell auch nur für dessen Voraussetzungen gültig. Zusätzlich wurde angenommen, daß α_{ij} und α_{ji} invariant gegenüber den Populationsgrößen sind. Für verschiedene K_i/α_{ij}- bzw. K_j/α_{ji}-Kombinationen kann gezeigt werden, ob eine Art durch die andere verdrängt oder

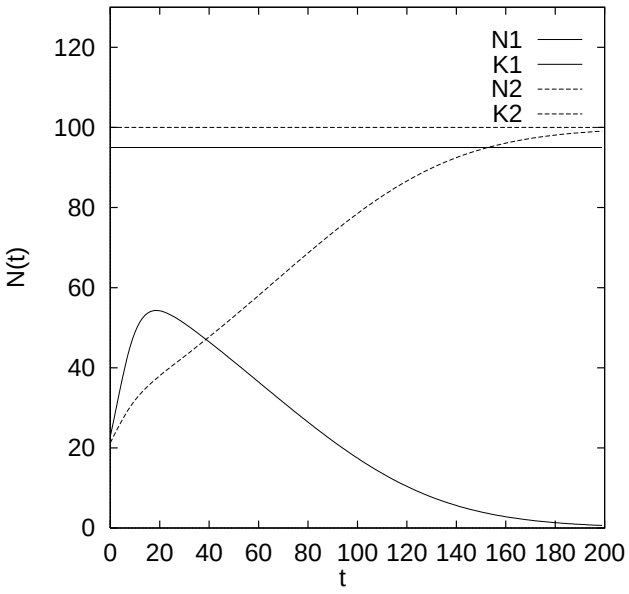

Abbildung 5.2: Lotka-Volterra-Modell für zwei konkurrierende Populationen

ob ein Gleichgewichtszustand erreicht werden kann, in dem beide Arten existieren. Gause kommt zu dem Ergebnis, daß konkurrierende Koexistenz nur möglich ist, falls beide Arten größere Dichte-Feedbacks auf ihre eigene Art als auf ihre jeweiligen Konkurrenten aufweisen ([Gau32]). Unter allen anderen Voraussetzungen stirbt eine der Arten aus.

Zwei Arten, die identische Nischen besetzen, wirken in gleicher Weise auf ihre eigene wie auf die konkurrierende Art, wodurch obige Voraussetzung für eine Koexistenz nicht gegeben ist. Dieser Sachverhalt ist als *Gause's Prinzip* bekannt.

Die Frage nach der Voraussetzung für die Koexistenz zweier Arten hat die Populationsdynamik lange Zeit beschäftigt und kann aufgrund ihrer Komplexität nicht eindeutig beantwortet werden. Zur Beschreibung des Wettbewerbs zwischen zwei Arten ist die alleinige Analyse ihrer Ressourcenanforderung, also ihrer Nischen, unzureichend. Zwei Arten, die zwar dieselbe Ressource konsumieren, diese aber unterschiedlich effizient in Biomasse umwandeln, weisen unterschiedliche Wettbewerbskoeffizienten α_{ij} und α_{ji} auf.

Eine weitere Fragestellung der Populationsdynamik betrifft den Effekt des Wettbewerbs zwischen Arten, die in räumlich strukturierten Populationen leben. So kann in der Natur beobachtet werden, daß Wettbewerb zwischen Arten, deren Konsumverhalten sich nur leicht unterscheidet, in heterogenen Umgebungen zu einer räumlichen Trennung führt ([Eml84]). Die räumliche Strukturierung ihrerseits fördert aufgrund ihres Puffereffekt die Koexistenz von konkurrierenden Arten, von denen ansonsten nur eine überleben würde.

5.4 Wettbewerb in Evolutionären Algorithmen

Die in der Natur zu beobachtenden Wettbewerbsformen, in denen verschiedene Arten um eine begrenzte Ressource konkurrieren, werden in dieser Arbeit auf die Evolutionären Algorithmen übertragen. In der Natur erhalten Populationen, die besser an ihre Umgebung angepaßt sind als andere, mehr Einheiten der begrenzten Ressource. Da die Ressource die

Grundlage der Produktion von Nachkommen darstellt, erhöht sich die Populationsgröße gut angepaßter Arten auf Kosten der anderen. Da jede Art durch die Hervorbringung neuer Individuen einem stetigen Wandel unterworfen ist, führt der Wettbewerb dazu, daß sich immer besser angepaßte Arten herausbilden.

Dieser Mechanismus läßt sich direkt auf Evolutionäre Algorithmen übertragen. Populationen, die mit unterschiedlichen Suchstrategien ausgestattet sind, entsprechen den verschiedenen Arten. Jede Population ist aufgrund ihrer Suchstrategie mehr oder weniger gut an die momentane Fitnesslandschaft angepaßt. Die Güte einer Population ist durch die in ihr enthaltenen Lösungen des Optimierungsproblems definiert. Da eine gut angepaßte Population die besten Chancen für die größte Verbesserung hat, soll sie in der nahen Zukunft auch mehr Nachkommen und damit mehr Versuche erhalten. Durch dieses Wettbewerbskonzept wird die zur Verfügung stehende Rechenzeit in die Strategien investiert, die den höchsten Fortschritt versprechen. Durch die gezielte oder zufällige Modifikation von Suchstrategien erfolgt auf diese Weise eine Anpassung von Suchstrategien an die aktuelle Situation.

Zur Modellierung des Wettbewerbsmechanismus wird der EA statt mit einer Population fester Größe nun mit mehreren Populationen variabler Größe ausgestattet. Die verschiedenen Populationen bearbeiten dieselbe Fitnessfunktion mit unterschiedlichen Suchstrategien. Sie entsprechen damit den verschiedenen Arten, die sich in derselben Umgebung befinden und die begrenzte Ressource konsumieren. Die begrenzte Ressource ist hier die Rechenzeit. Die Größen der Populationen entscheiden, wieviel Rechenzeit jede Population konsumiert. Eine gut angepaßte Population erhält aufgrund ihrer höheren Populationsgröße mehr Versuche als eine schlechter angepaßte.

Im Gegensatz zu den Modellen der Populationsdynamik, die beobachtete Wechselwirkungen zwischen verschiedenen Populationen erklären und modellieren, basiert das hier eingeführte Wettbewerbsmodell auf der Simulation dieser Wechselwirkungen. So ist hier nicht in erster Linie die Größe einer Population, sondern wie in der Natur die Güte einer Population für ihre weitere Entwicklung ausschlaggebend.

5.4.1 Konkurrierende Populationen

Das einfachste Modell, das einen Wettbewerb zwischen verschiedenen Populationen darstellt, arbeitet mit statischen Suchstrategien ([SVM94]). Die Suchstrategien der einzelnen Populationen werden initial festgelegt und bleiben während der Evolution unverändert. Zudem wird festgelegt, daß die Individuen aller Populationen denselben Verbrauch der begrenzten Ressource aufweisen. Weiterhin wird angenommen, daß die Ressource konstant ist und von den Populationen vollständig konsumiert wird, so daß die Gesamtzahl der Individuen konstant bleibt. Während der Evolution werden lediglich die Größen der einzelnen Populationen variiert. Die Modellierung des Wettbewerbs erfordert die Einführung folgender Maße und Operatoren:

- Qualität einer Population
- Ressourcenaufteilung
- Strategiemodifikation
- Migration

Um einen Wettbewerb zwischen den verschiedenen Populationen des EA zu ermöglichen, muß zunächst eine Bewertungsfunktion, die die Qualität der einzelnen Populationen widerspiegelt, aufgestellt werden.

Aufgrund ihrer Qualitäten erfolgt die Aufteilung der beschränkten Ressource. Eine Population mit hoher Qualität erhält mehr Anteile der Ressource und kann infolgedessen in der nächsten Generation mehr Nachkommen erzeugen. Auf diese Weise werden in der nächsten Generation mehr Funktionsauswertungen mit einer Strategie gemacht, die sich in der jüngsten Vergangenheit als die bessere erwiesen hat. Die Ressourcenaufteilung und die damit verbundene Veränderung der Populationsgrößen erfolgt in festgelegten Zeitintervallen. Ein solches Zeitintervall wird *Wettbewerbsintervall* genannt. In Abbildung 5.3 sind zwei mögliche Zusammensetzungen von drei konkurrierenden Populationen schematisch dargestellt.

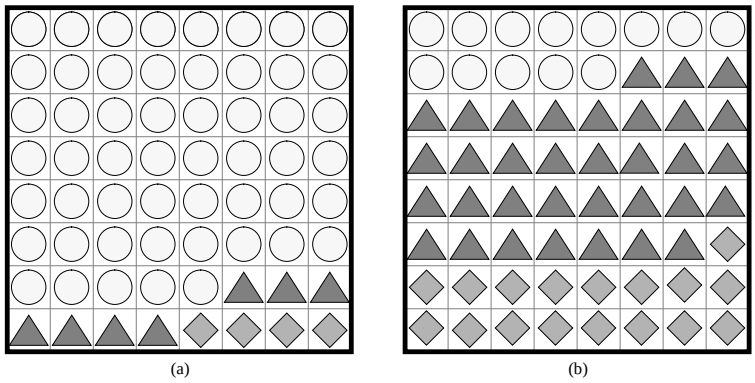

Abbildung 5.3: Schematische Darstellung von drei konkurrierenden Populationen: Die Anzahl der Felder entspricht der beschränkten Ressource. Jedes Symbol repräsentiert ein Individuum. In Abbildung a) dominiert die durch Kreise symbolisierte Population. In Abbildung b) sind die Populationsgrößen ausgeglichener.

Damit im Laufe der Evolution keine der Suchstrategien durch Aussterben einer Population verloren geht, wird eine untere Schranke für die Größe jeder Population festgelegt. Im Vokabular der Populationsdynamik werden die Populationen als geschützt bezeichnet. In bestimmten Zeitintervallen erfolgt eine Migration von Individuen zwischen den konkurrierenden Populationen. Die Migration ermöglicht eine Anpassung der Populationen aneinander, so daß die Startbedingungen für ein neues Wettbewerbsintervall angeglichen sind. Im Gegensatz zur geschachtelten Evolutionsstrategie von Rechenberg sieht dieses Wettbewerbsmodell Populationen als atomar an. Eine Population, deren Strategie zur Zeit keinen Fortschritt liefert, stirbt nicht aus, sondern reduziert nur ihre Größe. Auf diese Weise kann das Wettbewerbsmodell auch ohne eine explizite Modifikation der Suchstrategien eingesetzt werden.

Formal läßt sich ein Evolutionärer Algorithmus mit konkurrierenden Populationen durch die EA-Syntax folgendermaßen definieren.

Definition 15 (Evolutionärer Algorithmus mit konkurrierenden Populationen) *Ein* Evolutionärer Algorithmus mit konkurrierenden Populationen (cEA) *besteht aus mehreren EA-Populationen, die mit unterschiedlichen Suchstrategien ausgestattet sind. Die Größen der Populationen werden während der Suche durch ein Wettbewerbsverfahren* (𝕎)

gesteuert. Nach festgelegten Intervallen erfolgt durch das Kommunikationsverfahren ($\mathbb{K}$) *ein Informationsaustausch zwischen allen Populationen.*

$$\langle \mathsf{cea} \rangle ::= \mathbf{EA}\left(\langle \mathsf{pop}_1, \ldots, \mathsf{pop}_\kappa \rangle, \mathbb{T}, \mathbb{K}, \mathbb{W}\right) \tag{5.15}$$

In Algorithmus 5.1 ist ein Evolutionärer Algorithmus mit konkurrierenden Populationen skizziert.

Algorithmus 5.1 *Skizze eines EA mit konkurrierenden Populationen*

cEA()

1	$t \leftarrow 0$	
2	**for** $i \leftarrow 1$ **to** κ	
3	**do**	
4	$P_i(0) = \mathbb{I}(D)$	Initialisierung
5	$\hat{P}_i(t) \leftarrow \{\vec{a}_j(t), \mathbb{F}(\vec{a}_j(t))\}_{j=1}^{\lambda_i}$	Evaluierung
6	**while** $\mathbb{T}\left(\{\hat{P}_i(t)\}_{i=1}^{\kappa}\right) \neq \mathsf{true}$	
7	**do** $t \leftarrow t + 1$;	
8	**for** $i \leftarrow 1$ **to** κ	
9	**do**	
10	$\hat{P^{\mathrm{s}}}_i(t) \leftarrow \mathbb{S}(\hat{P}_i(t \Leftrightarrow 1) \cup \hat{P^{\mathrm{s}}}_i(t \Leftrightarrow 1))$	Selektion
11	$Q_i(t) \leftarrow \mathbb{P}(\hat{P^{\mathrm{s}}}_i(t))$	Paarbildung
12	$P_i(t) \leftarrow \mathbb{R}(Q_i(t))$	Rekombination
13	$P_i(t) \leftarrow \mathbb{M}(P_i(t))$	Mutation
14	$\hat{P}_i(t) \leftarrow \{\vec{a}_j(t), \mathbb{F}(\vec{a}_j(t))\}_{j=1}^{\lambda_i}$	Evaluierung
15		
16	$\mathbb{W}\left(\left\{\hat{P}_i(t)\right\}_{i=1}^{\kappa}\right)$	Wettbewerb
17		
18	$\mathbb{K}\left(\left\{\hat{P}_i(t)\right\}_{i=1}^{\kappa}\right)$	Migration
19		
20	**return** $\vec{a_k}(t) \quad mit \quad \mathbb{F}(\vec{a}_k(t)) < \mathbb{F}(\vec{a}_j(t)) \qquad \forall j \neq k$	

Der Aufbau eines cEAs ist in Abbildung 5.4 dargestellt. Exemplarisch sind hier zwei konkurrierende Population eingezeichnet. Jede Population hat, wie in einem Standard-EA, ihre eigene Generationenschleife. Der Wettbewerbsblock führt zu einer Kopplung der beteiligten Populationen.

Definition 16 legt das Wettbewerbsverfahren fest, welches das Wachstum der konkurrierenden Populationen steuert.

Definition 16 (Wettbewerbsverfahren)

Das Wettbewerbsverfahren *ermittelt in bestimmten Zeitintervallen die Qualitäten aller konkurrierenden Populationen und legt daraufhin die Größen der Populationen in der nächsten Generation fest.*

$$\mathbb{W} = \left(\mathbb{Q}, \mathbb{A}, \vec{\lambda}_{\min}, \eta, [\vec{\alpha}, \mathbb{V}]\right) \tag{5.16}$$

mit

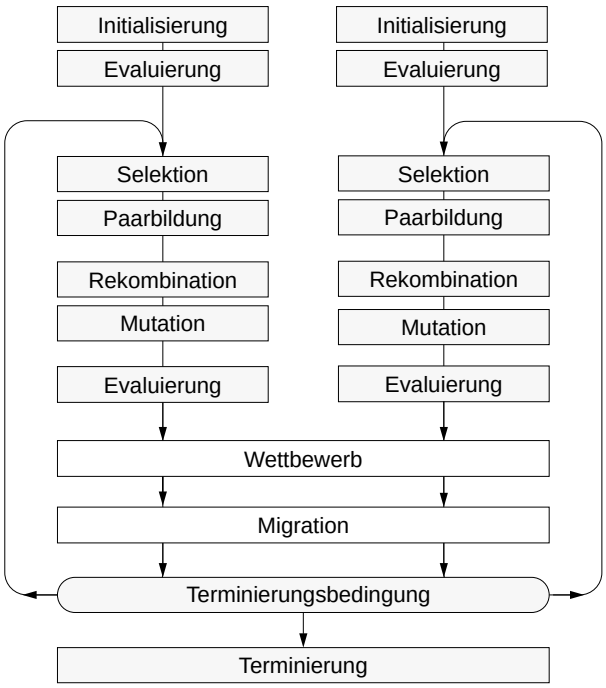

Abbildung 5.4: Aufbau eines Evolutionären Algorithmus mit konkurrierenden Populationen (cEA).

$$
\begin{array}{ll}
\mathbb{Q} : \left\{\hat{P}(i)\right\}_u^v \to \mathbb{R} & := \textit{Qualitätskriterium einer Population} \\
\mathbb{A} : (\mathbb{N} \times \mathbb{R})^\kappa \to \mathbb{N}^\kappa & := \textit{Ressourcenzuweisung} \\
\vec{\lambda}_{min} \in \mathbb{N}^\kappa & := \textit{minimale Populationsgrößen} \\
\eta & := \textit{Wettbewerbsintervall} \\
\vec{\alpha} \in \mathbb{N}^\kappa & := \textit{Verbrauchsfaktoren} \\
\mathbb{V} & := \textit{Strategienmodifikation}
\end{array}
$$

Das Qualitätskriterium *legt die Qualität einer Population fest. Die* Ressourcenzuweisung *verteilt auf Grundlage der Qualität die beschränkte Ressource auf die Populationen und regelt damit indirekt deren Größen. Der Vektor* $\vec{\lambda}_{\min}$ *spezifiziert die minimale Größe einer jeden Population. Optional stehen die Verbrauchsfaktoren* $\vec{\alpha}$ *und ein Verfahren zur Modifikation von Suchstrategien zur Verfügung. Die* Verbrauchsfaktoren *legen fest, wieviele Einheiten der beschränkten Ressource von einem Individuum einer Population verbraucht werden. Die Voreinstellung ist* $(1, \ldots, 1)^T$*. Das Verfahren zur Strategiemodifikation verändert Suchstrategien von Populationen.*

Der strukturelle Aufbau eines Wettbewerbmodells ist in Abbildung 5.5 dargestellt. Die erste Stufe eines Wettbewerbs besteht in der Bewertung der konkurrierenden Populationen. Anschließend erfolgt aufgrund der ermittelten Qualitäten die Aufteilung der beschränkten Ressource auf die Populationen. Aus den Ressourcen der Populationen werden im nächsten Schritt die Populationsgrößen der nächsten Generation berechnet. Die letzte Stufe, die Modifikation von Suchstrategien, ist optional.

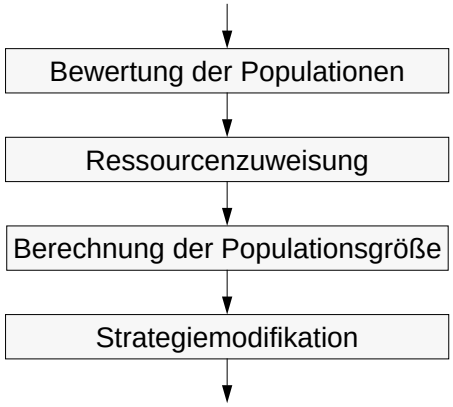

Abbildung 5.5: Struktureller Aufbau des Wettbewerbverfahrens.

5.4.2 Bewertung der Populationen

Der Bewertung einer Population kommt im Wettbewerbsverfahren eine zentrale Bedeutung zu. Die *Qualität* einer Population drückt den erwarteten Nutzen ihrer Suchstrategie in den nächsten Generationen aus. Auf ihrer Grundlage erfolgt die Zuweisung der Ressource und damit das Wachstum der Populationen. Indem Populationen, die eine hohe Qualität aufweisen, in der nächsten Generation mehr Nachkommen produzieren, erhält die zugehörige Suchstrategie mehr Versuche als die Suchstrategien von Populationen geringerer Qualität.

Die Berechnung der Qualität erfolgt auf Grundlage von Kennzahlen, die sich während der Evolution aus der Population gewinnen lassen. Größen, die als Kennzahlen in Frage kommen, sind:

- beste Fitness der Population (BF)
- durchschnittliche Fitness der Population (MF)
- Selektionserfolg (R)

Der Selektionserfolg R, der als Differenz der mittleren Fitness zweier aufeinanderfolgender Generationen definiert ist, ist ein Maß aus der Populationsgenetik, das von uns zur Analyse von rekombinationsbasierten Evolutionären Algorithmen verwendet wurde ([MSV93b], [MSV93a], [MSV94], [MSV95], [VMSV96]). Zur Berechnung der Qualität können neben den Kennzahlen der aktuellen Generation auch Kennzahlen vergangener Generationen verwendet werden. Dadurch lassen sich mögliche zufallsbedingte Schwankungen kompensieren.

Im folgenden wird ein rangbasiertes Qualitätskriterium eingeführt, das auf Kennzahlen vergangener Generationen basiert.

Qualitätskriterium $\mathbb{Q}^{\mathrm{TOP}}_{\gamma,\omega}$

Dieses Qualitätskriterium verwendet die durch den Parameter γ spezifizierten Kennzahlen aus den vergangenen ω Generationen. Als Kennzahlen können beispielsweise die oben vorgestellten ($\gamma \in \{BF, MF, R\}$) verwendet werden.

Dazu werden die Populationen, deren Kennzahlen in den vergangenen ω Generationen den jeweils besten Wert aufwiesen, in einem Vektor $\vec{b} = (b_0, \ldots, b_{\omega-1})^T$ vermerkt. Der k-te

Eintrag von $\vec{b}$ enthält den Index der Population, die k Generationen zuvor das Individuum mit der besten Kennzahl enthielt. Die Qualität der i-ten Population berechnet sich aus dem Vektor $\vec{b}$ wie folgt:

$$\boxed{Q_i(\vec{b}) = \sum_{k=0}^{\omega-1} \begin{cases} \frac{\omega-k}{\omega} & : \quad i = b_k \\ 0 & : \quad i \neq b_k \end{cases}} \tag{5.17}$$

Der Besitz der besten Kennzahl wird auf diese Weise mit deren Auftreten gewichtet. Je weiter der Besitz der besten Kennzahl zurück liegt, umso weniger trägt er zur Qualität bei.

Das Qualitätskriterium $\mathbb{Q}^{\mathrm{TOP}}_{\mathrm{BF},10}$, in dem die Qualität aus den besten Fitnesswerten der letzten 10 Generationen ermittelt wird, hat sich in vielen Experimenten als sehr zweckmäßig erwiesen und wird in den Experimenten in Kapitel 6 eingesetzt.

5.4.3 Ressourcenzuweisung

Die Aufgabe der Ressourcenzuweisung besteht in der Verteilung der zur Verfügung stehenden Ressourcen auf die konkurrienden Populationen. K bezeichnet im folgenden die Gesamtmenge der zur Verfügung stehenden Ressourcen. Wie in der Populationsdynamik wird K im folgenden auch als *Kapazität* bezeichnet. Mit K_i^t wird die Ressourcenmenge bezeichnet, die der $i\Leftrightarrow$ten Population in Generation t zur Verfügung steht. Die initialen Kapazitäten K_i^0 berechnen sich aus den initialen Populationsgrößen λ_i^0 und den Verbrauchsfaktoren α_i der Populationen:

$$K_i^0 = \alpha_i \cdot \lambda_i^0 \tag{5.18}$$

Die Bedeutung der Verbrauchsfaktoren wird in Abschnitt 5.4.4 behandelt. Die Gesamtkapazität K, die während der Evolution unverändert bleibt, ist die Summe der Einzelkapazitäten:

$$K = \sum_{i=0}^{\kappa} K_i^t \qquad t = 0, 1, \ldots \tag{5.19}$$

Die Verteilung der Ressource, die aufgrund der Qualitäten der Populationen erfolgt, kann auf verschiedene Weise realisiert werden:

- abhängig oder unabhängig von der momentanen Aufteilung der Kapazität
- proportional oder rangbasiert zur Qualität

Das Resultat der Ressourcenzuweisung ist eine Umverteilung der vorhandenen Ressourcen. Da die Gesamtkapazität konstant ist und zwischen Kapazität und Populationsgröße ein linearer Zusammenhang herrscht, ist ein unkontrolliertes Wachstum der Gesamtpopulation ausgeschlossen.

Im folgenden wird ein rangbasiertes Verfahren zur Ressourcenzuweisung vorgestellt, das die Kapazitäten in Abhängigkeit der vorhandenen Kapazitäten umverteilt.

Ressourcenzuweisung $\mathbb{A}_{\gamma}^{\mathrm{WTA}}$

Dieses Verfahren hat die Bezeichnung *WTA* (winner-take-all), da die Population mit der höchsten Qualität die Verluste aller anderen Populationen erhält. Das Verfahren ähnelt einem Spiel, bei dem alle Spieler einen bestimmten Einsatz setzen und um den gesamten Einsatz spielen. Der Gewinner des Spiels ist die Population, die die beste Qualität aufweist.

Das hier verwendete Verfahren unterscheidet die Population, die die höchste Qualität aufweist, von allen anderen. Diese Population wird im folgenden als *Gewinner*, alle anderen als *Verlierer* bezeichnet. Der Zuwachs der Gewinner-Population entspricht dem akkumulierten Verlust aller Verlierer-Populationen. Der Verlust einer Population ist proportional zu ihrer Größe und wird durch den *Verlustfaktor* $\gamma \in [0,1]$ festgelegt.

Der Index der Population mit der höchsten Qualität wird im folgenden mit w^* bezeichnet. Falls mehrere Populationen die beste Qualität aufweisen, wird eine von ihnen zufällig gezogen und als Gewinner deklariert. Zur Berechnung der neuen Kapazitäten werden zunächst die Verluste der Verlierer durch die folgende Gleichung bestimmt:

$$\boxed{L_i^t = \begin{cases} 0 & : \; i = w^* \\ \min\left(\gamma K_i, K_i^t \Leftrightarrow K_i^{\min}\right) & : \; i \neq w^* \end{cases}} \tag{5.20}$$

für $i = 1, \ldots, \kappa$. $K_i^{\min} = \lambda^{\min} \cdot \alpha$ bezeichnet die minimale Ressourcenmenge. Der Verlustfaktor γ wird normalerweise auf 0.125 gesetzt. Das bedeutet, daß eine Population ein Achtel ihrer bisherigen Ressource einbüßt. Aus den Verlusten der Verlierer lassen sich anschließend die neuen Kapazitäten berechnen. Der Gewinner erhält die akkumulierten Verluste aller Verlierer.

$$\boxed{\Delta K_i^t = \begin{cases} \sum\limits_{j=1}^{\kappa} L_j^t & : \quad i = w^* \\ \Leftrightarrow L_i^t & : \quad i \neq w^* \end{cases}} \tag{5.21}$$

Die Kapazitäten der nächsten Generation sind damit:

$$\boxed{K_i^{t+1} = K_i^t + \Delta K_i^t} \tag{5.22}$$

5.4.4 Berechnung der Populationsgrößen

Die Berechnung der Populationsgrößen der nächsten Generation erfolgt aufgrund der den Populationen zugeteilten Kapazitäten. Zwischen der Kapazität K_i einer Population und deren zukünftiger Populationsgröße λ_i herrscht ein linearer Zusammenhang:

$$\boxed{\lambda_i = \alpha_i \cdot K_i} \tag{5.23}$$

wobei α_i der Verbrauchsfaktor der i-ten Population ist.

Das hier eingeführte Wettbewerbsmodell sieht für die Berechnung der Populationsgrößen zwei Alternativen vor. In der ersten Variante werden die Populationsgrößen im gleichen Verhältnis wie die Kapazitäten bestimmt. Die zweite Variante modelliert einen artenspezifischen Pro-Kopf-Verbrauch der Ressourcen, indem die Ressourcen in den verschiedenen Populationen unterschiedlich gewichtet werden. Die Wahl des Modells erfolgt durch die Belegung der Verbrauchsfaktoren.

Identische Verbrauchsfaktoren

Falls die Verbrauchsfaktoren der konkurrierenden Populationen identisch sind, entspricht das Verhältnis der verschiedenen Populationsgrößen genau dem Verhältnis der verschiedenen Kapazitäten. Die Verbrauchsfaktoren, die optionale Parameter des Wettbewerbsverfahrens sind, werden, falls sie nicht explizit spezifiziert sind, auf den Wert 1.0 gesetzt. In diesem Fall entspricht die Populationsgröße genau der Kapazität, wodurch sich Gleichung 5.23 vereinfacht zu:

$$\lambda_i = K_i, \qquad i = 1, \ldots, \kappa \tag{5.24}$$

Unterschiedliche Verbrauchsfaktoren

In der Natur setzen verschiedene Arten die ihnen zur Verfügung stehende Ressource unterschiedlich in die Produktion von Nachkommen um. Ein anschauliches Beispiel ist die Ressource Lebensraum. Aufgrund ihrer geringeren Platzanforderungen können wesentlich mehr kleine als große Lebewesen in einem beschränkten Terrain existieren. Der unterschiedliche Pro-Kopf-Verbrauch der beschränkten Ressource reguliert ihre Populationsgrößen.

Analog zur Natur, in der eine Art eine bestimmte Populationsgröße benötigt, um in ihrer Umgebung existieren zu können, benötigen auch EA-Populationen in Abhängigkeit ihrer Suchstrategien eine bestimmte Größe, um ihre volle Effizienz zu entfalten.

Zur Modellierung des artenspezifischen Pro-Kopf-Verbrauchs werden in das Wettbewerbsmodell *Verbrauchsfaktoren* eingeführt ([SVM96]). Der Verbrauchsfaktor einer Population spezifiziert den Pro-Kopf-Verbrauch der beschränkten Ressource. Abbildung 5.6 zeigt die Anteile dreier konkurrierender Populationen mit (a) identischen und (b) unterschiedlichen Verbrauchsfaktoren. Die Größe der Symbole kennzeichnet den Verbrauchsfaktor eines Individuums. Ein Individuum mit hohem Verbrauchsfaktor verbraucht mehr Einheiten der begrenzten Ressource als Individuen mit niedrigeren Verbrauchsfaktoren.

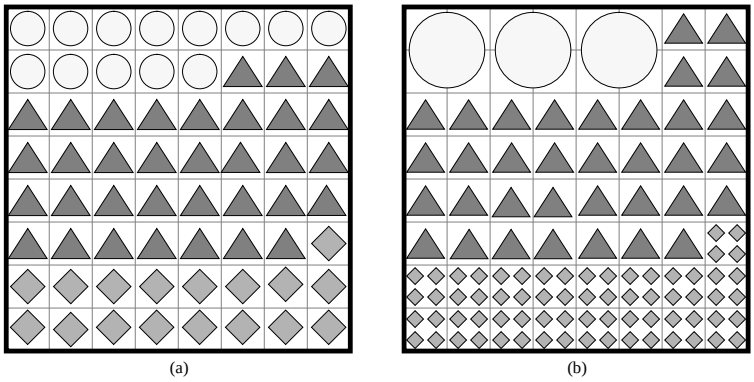

Abbildung 5.6: Wettbewerb mit (a) identischen und (b) unterschiedlichen Verbrauchsfaktoren.

Da die Kapazität einer Population Grundlage für deren Nachkommen ist, paßt dieses Verfahren indirekt die verschiedenen Populationsgrößen an. Gleichung 5.23 zeigt den funktionalen Zusammenhang zwischen Kapazität und Populationsgröße. Falls die Verbrauchsfaktoren α aller Populationen 1.0 sind, entspricht die Populationsgröße der Kapazität. Da

die konstante Ressource lediglich umverteilt wird, bleibt auch die gesamte Populationsgröße konstant.

Falls der Verbrauchsfaktor einer Population kleiner als 1.0 ist, bedeutet dies, daß diese Population mit einer Einheit der beschränkten Ressource K mehr als ein Individuum unterhalten kann. Durch die Umverteilung der Ressource verändert sich in Abhängigkeit der Verbrauchsfaktoren also die gesamte Populationsgröße.

5.4.5 Migrationsverfahren

Die konkurrierenden Populationen werden in bestimmten Intervallen aneinander angeglichen. Der Grund für diese Angleichung liegt darin, daß für zukünftige Wettbewerbe ähnliche Voraussetzungen geschaffen werden. Populationen mit bislang schlechten Strategien erhalten so gleiche Startbedingungen für das folgende Wettbewerbsintervall. Sie profitieren damit von den Erfolgen der anderen Populationen.

Die Notwendigkeit einer Migration wird besonders bei einem Wettbewerb zwischen zwei Populationen, deren Suchstrategien verschiedene Schrittweiten aufweisen, deutlich. Durch die Migration eines Individuums der Population mit großer Schrittweite in die mit kleiner Schrittweite kann die Population mit kleiner Schrittweite auf der Lösung der anderen aufsetzen und muß die bereits gefundene Lösung nicht erst selbst generieren.

Best-To-All-Migrationsverfahren $\mathbb{K}_{\theta}^{\mathrm{B2A}}$

Das im Wettbewerbsmodell verwendete Migrationsverfahren ist ein Spezialfall des Migrationsverfahrens des in Abschnitt 3.6.1 beschriebenen Inselmodells. Da hier, im Gegensatz zum Inselmodell, nicht die Varianzerhaltung im Vordergrund steht, kann auf die Wahl einer speziellen Topologie verzichtet werden. Stattdessen wird eine Kopie des global besten Individuums von der entsprechenden Population an alle anderen Populationen geschickt. Das Migrationsintervall wird durch den Parameter θ spezifiziert.

Behandlung unterschiedlicher Kodierungen

Das Wettbewerbskonzept ist nicht auf Populationen mit gleicher Kodierung beschränkt. Populationen verschiedener EA-Klassen können ebenso in einen Wettbewerb eingebunden sein wie Populationen derselben Klasse. Falls sich zwei Populationen lediglich in der Kodierung der Objektparameter unterscheiden, werden die Individuen von dem Absender in Gleitkommavektoren dekodiert und beim Empfänger in dessen Kodierung umgewandelt.

Die Migration zwischen Populationen, die mit unterschiedlichen internen Modellen arbeiten, ist dagegen nicht so eindeutig und erfordert ein auf die Kodierungen abgestimmtes Verfahren. Interne Modelle enthalten Informationen über die Fitnesslandschaft, die ein EA während der Suche akkumuliert hat. Sie können, wie bei der BGA-Mutation mit Hauptachsentransformation, Bestandteil der Population oder, wie die Strategieparameter der Evolutionsstrategien, Bestandteil des einzelnen Individuums sein.

Da unterschiedliche interne Modelle in der Regel nicht dieselbe Information enthalten, ist eine Konversion zwischen den zugrundeliegenden Kodierungen nicht immer möglich. Beispielsweise kann ein aus einer GA-Population stammendes Individuum aufgrund der fehlenden Strategieparameter nicht ohne weiteres in eine ES-Population aufgenommen werden. Da der Immigrant lediglich die Objektparameter bereitstellt, müssen die Strategieparameter auf eine andere Weise bestimmt werden. Als Möglichkeit bietet sich die

übliche Initialisierung an. Da sich ein Immigrant, der zwar über gute Objektparameter, aber nur über ein schlechtes internes Modell verfügt, in einer gut angepaßten Population kaum behaupten kann, ist es sinnvoll, das interne Modell des besten Individuums der Population in den Immigranten zu übernehmen.

Wie die fehlende Information ergänzt wird, hängt von den Kodierungen der konkurrierenden Populationen ab und muß daher für das jeweilige Wettbewerbsmodell entworfen werden.

5.4.6 Strategiemodifikation

Durch die Einführung eines Verfahrens zur Strategiemodifikation ($\mathbb{V}$) können die Strategien der verschiedenen Populationen während einer Evolution modifiziert werden. Die Modifikation von Strategien führt zu einem *Meta*-EA, der auf der Populationsebene einen neuen Optimierungsprozeß durchführt. Mit der Einführung der Strategiemodifikation lassen sich als Grenzfall die in Abschnitt 5.2.2 beschriebenen geschachtelten Evolutionären Algorithmen modellieren.

Die Intention der Strategiemodifikation im Wettbewerbsmodell ist jedoch defensiver. Im Gegensatz zu den geschachtelten Evolutionsstrategien wird die Menge der zur Verfügung stehenden Strategien eingeschränkt. Nachdem die *Menge der selektierten Strategien* $\tilde{S} = \{\tilde{S}_1, \ldots, \tilde{S}_\kappa\} \subseteq S$ aus der Menge aller Strategien S zusammengestellt ist, wird auf $\tilde{S}$ eine Ordnung

$$\tilde{S}_{i_1} < \tilde{S}_{i_2} < \ldots < \tilde{S}_{i_\kappa} \qquad \text{mit } i_j \in [1, \kappa] \tag{5.25}$$

definiert.

Die Ordnung hängt von der Menge der selektierten Strategien ab. Unterscheiden sich die Strategien beispielsweise in den verwendeten Schrittweitenparametern, kann die Ordnung aufgrund dieser Schrittweitenparameter festgelegt werden.

Die konkurrierenden Populationen werden mit Strategien aufeinanderfolgender Ordnungen ausgestattet, so daß die Populationen einen Strategiebereich oder ein *Strategiefenster* aus der Menge aller Strategien S abdecken. Die im Strategiefenster enthaltenen Strategien werden als *aktive Strategien* bezeichnet.

$$\tilde{S}^a = \{\tilde{S}^a_{i_1}, \ldots, \tilde{S}^a_{i_\kappa}\} \subset \tilde{S} \tag{5.26}$$

Die beiden Populationen, deren Strategien die niedrigste oder die höchste Ordnung der aktiven Strategien haben, werden im folgenden als *Randpopulationen* bezeichnet. Die Randpopulation mit der Strategie kleinster Ordnung wird *linke Randpopulation*, die mit der Strategie höchster Ordnung *rechte Randpopulation* genannt. Falls sich eine dieser Randpopulationen als besonders erfolgreich erweist, werden die Strategien aller Populationen folgendermaßen modifiziert: Falls die linke Randpopulation die erfolgreichste ist, wird das Strategiefenster nach links verschoben oder ausgedehnt. Falls die rechte Randpopulation die erfolgreichste ist, wird es nach rechts verschoben. Das Kriterium, das den Erfolg einer Strategie quantifiziert, kann die Qualität der Population oder die Populationsgröße sein.

Strategiemodifikation $\mathbb{V}^{\mathrm{BGA,RHO}}_{\upsilon,\zeta}$

Dieses Verfahren zur Strategiemodifikation paßt den Schrittweitenparameter ρ aus der BGA-Mutation in jeder ζ-ten Generation an. Die Menge der selektierten Strategien unterscheidet sich ausschließlich im Schrittweitenparameter ρ. Im folgenden bezeichnet ρ_j den

Schrittweitenparameter der Strategie $\tilde{S}_{i_j}$. Sämtliche Schrittweitenparameter werden modifiziert, falls die linke oder rechte Randpopulation mindestens 90 Prozent ihrer maximalen Populationsgröße besitzt.

$$\rho_j = \begin{cases} \frac{1}{\upsilon} \cdot \rho_j & : \quad \lambda(\tilde{S}^a_{i_1}) \geq 0.9 \cdot \lambda^{\max}(\tilde{S}^a_{i_1}) \\ \upsilon \cdot \rho_j & : \quad \lambda(\tilde{S}^a_{i_\kappa}) \geq 0.9 \cdot \lambda^{\max}(\tilde{S}^a_{i_\kappa}) \\ \rho_j & : \quad sonst \end{cases} \tag{5.27}$$

$\lambda(S)$ bezeichnet die Populationsgröße der Population mit Strategie S. $\lambda^{\max}(S)$ bezeichnet die maximale Größe, die die Population mit Strategie S erreichen kann.

Schnecke und Vornberger wenden dieses Wettbewerbsmodell in einer modifizierten Form zur Optimierung von VLSI-Layouts an ([SV96]). Sie setzen mehrere Populationen mit festen Populationsgrößen ein, um externe Parameter der genetischen Operatoren an die aktuelle Suchsituation anzupassen. Da die Populationsgrößen fest sind, erfolgt die Anpassung der Strategie aufgrund des Qualitätskriteriums. Um den Einfluß migrierender Individuen zu begrenzen, wählten sie als Qualitätskriterium den Selektionserfolg R, der als Differenz der mittleren Fitness zweier aufeinanderfolgender Generationen definiert ist.

5.4.7 Selektion in Populationen variabler Größe

Populationsgröße und Selektion sind eng miteinander verknüpft. Bei der Schnittselektion führt eine niedrige Remontierungsrate in einer kleinen Population sehr viel schneller zur Stagnation als in einer großen Population. Aus diesem Grund wird die Remontierungsrate in einem Evolutionären Algorithmus mit konstanter Populationsgröße passend zur Populationsgröße gewählt. In einer kleinen Population wird in der Regel weicher selektiert als in einer großen. Da die Größen der Populationen in Wettbewerbsmodellen variabel sind, muß die Remontierungsrate der Schnittselektion als Funktion über die Populationsgröße definiert werden.

Schnittselektion mit populationsgrößenabhängiger Remontierungsrate $\mathbb{S}^{\mathrm{cEA,TR}}_{f,[c]}$

Die Remontierungsrate der Schnittselektion sei von der Größe der Population abhängig. Je größer die Population ist, umso niedriger wird die Remontierungsrate. Der funktionale Zusammenhang zwischen Populationsgröße und Remontierungsrate wird durch die Funktion $f : \mathbb{N} \to \mathbb{N}$ und den Faktor c spezifiziert. Der optionale Parameter c hat standardmäßig den Wert 1.0. Die Anzahl der Eltern wird aus der Populationsgröße folgendermaßen bestimmt:

$$\mu = \min(c \cdot f(\lambda), \lambda) \tag{5.28}$$

Das Verhältnis zwischen Anzahl der Eltern und Anzahl der Nachkommen entspricht der Remontierungsrate:

$$\vartheta = \min\left(\frac{c \cdot f(\lambda)}{\lambda}, 1.0\right) \tag{5.29}$$

Aus der weiter oben eingeführten Schnittselektion kann die Schnittselektion für Wettbewerbsmodelle folgendermaßen abgeleitet werden:

$$\mathbb{S}^{\mathrm{cEA,TR}}_{f,c} = \mathbb{S}^{\mathrm{EA,TR}}_{\frac{c \cdot f(\lambda)}{\lambda},e} \tag{5.30}$$

Die in Wettbewerbsmodellen verwendete Selektion ist elitär, damit das durch eine Migration in die Population gelangte Individuum, das eine Kopie des global besten Individuums ist, nicht verloren geht, ohne daß ein besseres entsteht.

Eine Funktion, die die Remontierungsrate mit zunehmender Populationsgröße verkleinert und somit die Härte der Selektion erhöht, ist die Wurzelfunktion. Die Remontierungsrate der Schnittselektion für Wettbewerbsmodelle in Abhängigkeit der Populationsgröße ist in Abbildung 5.7 dargestellt. Die Abbildung zeigt die Remontierungsraten für verschiedene Werte des Parameters c. Mit Hilfe des Faktors c kann die aus der Funktion f resultierende Remontierungsrate linear modifiziert werden. Falls als Funktion $f(\lambda) = \lambda$ verwendet wird, entspricht diese Selektion der einfachen Schnittselektion $\mathbb{S}^{\mathrm{EA,TR}}$, so daß der Faktor c gleich der Remontierungsrate ist.

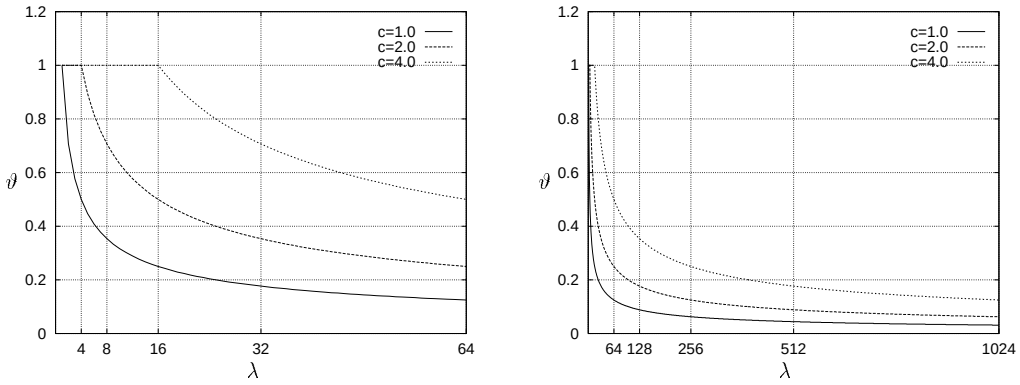

Abbildung 5.7: Remontierungsrate ϑ der Schnittselektion $\mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT},c}$ mit $f(\lambda) = \sqrt{\lambda}$ in Abhängigkeit der Populationsgröße λ.

5.5 Instanzen des Wettbewerbsmodells

5.5.1 Spezifikation von Wettbewerbsverfahren

Ein Wettbewerbsverfahren setzt sich im wesentlichen aus den Komponenten Qualitätskriterium, Ressourcenzuweisung und Strategiemodifikation zusammen. Die Spezifikation eines Wettbewerbsverfahren besteht in der Kombination dieser drei Komponenten. Zusätzlich müssen die Intervalle, in denen Wettbewerb und Strategiemodifikation stattfinden, angegeben werden.

Ein Wettbewerbsverfahren, das sich in vielen Simulationen bewährt hat, ist das WTA-Wettbewerbsverfahren, das im folgenden eingeführt wird.

Spezifikation 13 *WTA-Wettbewerbsverfahren* $\mathbb{W}^{\mathrm{WTA}}_{\eta,\vec{\lambda}^{\min},[\vec{\alpha},\mathbb{V}]}$

$$\mathbb{W}^{\mathrm{WTA}}_{\eta,\vec{\lambda}^{\min},[\vec{\alpha},\mathbb{V}]} ::= \mathbb{W}\left(\mathbb{Q}^{\mathrm{TOP}}_{\mathrm{BF},10}, \mathbb{A}^{\mathrm{WTA}}_{\alpha}, \vec{\lambda}^{\min}, [\vec{\alpha}, \mathbb{V}]\right)$$

Die Spezifikation der Verbrauchsfaktoren $\vec{\alpha}$ und der Strategiemodifikation sind optional. Aus diesem allgemeinen Wettbewerbsverfahren lassen sich durch die Spezifikation der Parameter konkrete Wettbewerbsverfahren bilden.

5.5.2 Spezifikation von Instanzen

Die Verwendung des Wettbewerbsmodells bietet die Möglichkeit, verschiedene Suchstrategien in EA einzubeziehen und so deren Effizienz und Robustheit zu steigern. Gleichzeitig stellt sich die Frage nach der geeigneten Zusammenstellung der Suchstrategien. Die Suchstrategien der konkurrierenden Populationen sollten sich hinreichend unterscheiden, so daß keine unnötige Konkurrenz zwischen ähnlichen Strategien erfolgt.

Im folgenden werden für die verschiedenen EA-Klassen einige Wettbewerbsmodelle eingeführt. Ein wettbewerbsgesteuerter Evolutionärer Algorithmus wird im folgenden durch ein vorgestelltes 'c' und den Namen der EA-Klasse, zu der die Populationen gehören, gekennzeichnet. Falls die konkurrierenden Populationen alle aus einer EA-Klasse stammen, werden die Wettbewerbsmodelle mit cBGA, cES bzw. cGA bezeichnet. Ein Wettbewerbsmodell, das Populationen aus unterschiedlichen EA-Klassen beinhaltet, wird allgemein cEA genannt.

Die hier eingeführten Instanzen der Wettbewerbsmodelle werden durch einen hochgestellten Stern von den in Kapitel 6 spezifizierten Instanzen, mit denen die Leistungsbewertung durchgeführt werden, unterschieden. Gleichnamige Wettbewerbsmodelle unterscheiden sich lediglich in der Parametrisierung voneinander.

cBGA mit unterschiedlichen Schrittweitenparametern

Der in Abschnitt 3.5.5 eingeführte BGA-Mutationsoperator verwendet standardmäßig den Präzisionsparameter $k = 32$ und einen dazu passenden Schrittweitenparameter ρ, der den gewünschten Suchbereich abdeckt. Die Untersuchungen der Mutationsverteilungen in Abschnitt 4.3 zeigten, daß eine BGA-Mutationsverteilung mit steigendem Präzisionsparameter k zwar robuster aber auch ineffizienter wird. Tabelle 4.5 auf Seite 84 enthält die Fortschrittsgeschwindigkeiten für verschiedene Präzisionsparameter. Durch die Reduzierung des Präzisionsparameters von 32 auf 2 wird die zehnfache Fortschrittsgeschwindigkeit erzielt. Aus diesem Grund erscheint es sinnvoll, den Präzisionsparameter zu reduzieren und mit verschiedenen Schrittweitenparametern zu arbeiten, die gleichzeitig den ganzen Suchbereich abdecken und dieselbe Präzision liefern wie der Standardwert $k = 32$. Für den Einsatz unterschiedlicher Schrittweiten eignet sich das Wettbewerbsmodell, in dem jede Population mit einem unterschiedlichen Schrittweitenparameter arbeitet. Die Schrittweitenparameter müssen so gewählt werden, daß die Schrittweiten der beteiligten Populationen den festgelegten Suchraum mit der gewünschten Genauigkeit abdecken.

Folgende Gleichung zeigt, wie der Schrittweitenparameter der i-ten Population aus dem vorgegebenen Schrittweitenparameter $\hat{\rho}$ und dem Präzisionsparameter k berechnet wird. $s^{\min}$ und $s^{\max}$ bezeichnen die minimale bzw. maximale Schrittweite der jeweiligen Mutationsverteilung.

$$s_i^{\min} = s_i^{\max} \cdot 2^{-k} \tag{5.31}$$

$$s_i^{\max} = \begin{cases} \hat{\rho} & : \quad i = 1 \\ s_{i-1}^{\min} \cdot 2 & : \quad i > 1 \end{cases} \tag{5.32}$$

$$\rho_i = s_i^{\max} \tag{5.33}$$

Tabelle 5.1 enthält die Parametereinstellungen der Mutationsverteilung eines Standard-BGA und die eines cBGA. Mit vier Populationen kann der Präzisionsparameter von 32 auf 9 gesenkt werden, ohne daß Mutationsintervall verändert wird.

ALG	κ	P	k	ρ	$s^{\min}$	$s^{\max}$
BGA	1	1	32	1.024E+01	2.384E-09	1.024E+01
cBGA	4	1	9	1.024E+01	2.000E-02	1.024E+01
		2	9	4.000E-02	7.813E-05	4.000E-02
		3	9	1.563E-04	3.052E-07	1.563E-04
		4	9	6.104E-07	1.192E-09	6.104E-07

Tabelle 5.1: Parametereinstellungen und Schrittweitenintervalle eines Standard-BGA und eines cBGA mit unterschiedlichen Schrittweitenparametern. κ gibt die Anzahl der Populationen und P den Index der Population an. k ist der Präzisionsparameter und ρ der Schrittweitenparameter. $s^{\min}$ und $s^{\max}$ geben die minimale bzw. maximale Schrittweite der durch k und ρ spezifizierten BGA-Mutationsverteilung an.

Spezifikation 14 *cBGA-Instanz mit unterschiedlichen Schrittweitenparametern*

$$
\begin{aligned}
\mathsf{cbga1}^* ::= \mathbf{EA}\Big(\ & \mathsf{POP}^{\mathrm{BGA,FR}}(52, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 9, 1.024\cdot 10), \\
& \mathsf{POP}^{\mathrm{BGA,FR}}(\ 4, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 9, 4.000\cdot 10^{-2}), \\
& \mathsf{POP}^{\mathrm{BGA,FR}}(\ 4, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 9, 1.563\cdot 10^{-4}), \\
& \mathsf{POP}^{\mathrm{BGA,FR}}(\ 4, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 9, 6.104\cdot 10^{-7}), \\
& \mathbb{W}^{\mathrm{WTA}}_{4,4^4}, \mathbb{K}^{\mathrm{B2A}}_{16}, \mathbb{T}^{\mathrm{EXP}}\ \Big)
\end{aligned}
$$

Die Abbildungen 5.8 und 5.9 auf den Seiten 114 und 115 zeigen die Entwicklung der besten und mittleren Fitnesswerte sowie der Populationsgrößen und Qualitätsmaße von vier konkurrierenden Populationen. Die Population mit dem größten Schrittweitenparameter macht anfangs, da die Population noch weit vom Optimum entfernt ist, die besten Fortschritte und hält die maximale Populationsgröße. Nach 300 Generationen schwächt sich der Fortschritt aufgrund der Nähe zum Optimum ab und die Qualität der Population mit dem zweitgrößten Schrittweitenparameter übertrifft die der ersten Population. Infolgedessen verliert die erste Population in der nächsten Generation an Populationsgröße, während die Populationsgröße der zweiten steigt. Dieser Vorgang wiederholt sich für die beiden verbliebenen Populationen noch zweimal. Der wellenförmige Verlauf der besten Fitnesswerte entsteht dadurch, daß die momentan beste Population aufgrund ihrer zu großen minimalen Schrittweite in eine Stagnationsphase eintritt bevor die Population mit dem nächstniedrigeren Schrittweitenparameter die Suche fortführt.

Während die besten Fitnesswerte aller Populationen durch die regelmäßige Migration eng beieinander liegen, spiegeln die mittleren Fitnesswerte das Suchverhalten der verschiedenen Strategien wider. Ab einer bestimmten Zielannäherung stagniert die mittlere Fitness einer Population aufgrund der zu großen minimalen Schrittweite.

cBGA mit unterschiedlichen Verbrauchsfaktoren

Dieser cBGA modelliert einen Wettbewerb zwischen einer rekombinationsbasierten Breitensuche $\mathsf{POP}^{\mathrm{BGA,FR}}$ und einer mutationsgetriebenen Richtungssuche $\mathsf{POP}^{\mathrm{BGA,BLR}}$. Mit dieser Kombination ist der cBGA in der Lage, in der Anfangsphase durch die Rekombination gute Regionen von multimodalen oder verrauschten Funktionen zu finden und sich in der Endphase durch die Mutation effizient an das Optimum anzunähern.

Die Population mit Fuzzy-Rekombination benötigt für ihre Breitensuche eine genügende Varianz. Die Varianz ist umso größer je größer die Population ist. Demgegenüber arbeitet die mutationsbasierte Population am effizientesten in einer kleinen Population.

Um beiden Suchstrategien die gewünschte Populationsgröße bereitzustellen, wird das Wettbewerbsmodell mit unterschiedlichen Verbrauchsfaktoren verwendet. Die rekombinationsbasierte Population erhält einen kleineren Verbrauchsfaktor als die mutationsbasierte Population, so daß ein Individuum der Mutationspopulation den gleichen Ressourcenverbrauch wie mehrere Individuen der Rekombinationspopulation hat.

Spezifikation 15 *cBGA-Instanz mit unterschiedlichen Suchstrategien*

$$
\begin{aligned}
\mathsf{cbga2}^{*} ::= \mathbf{EA}\big(\ & \mathsf{POP}^{\mathrm{BGA,FR}}(240, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT,4.0}}, 32, 10.24) \\
& \mathsf{POP}^{\mathrm{BGA,BLR}}(\ 4, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT,1.0}}, 32, 10.24), \\
& \mathbb{W}^{\mathrm{WTA}}_{4,\{16,4\},\{0.25,1\}}, \mathbb{K}^{\mathrm{B2A}}_{16}, \mathbb{T}^{\mathrm{EXP}}\ \big)
\end{aligned}
$$

Die Gesamtkapazität ist $K = 64$. Die Rekombinationspopulation wird mit der maximalen Populationsgröße initialisiert ($\lambda_1^{\max} = (K \Leftrightarrow K_2^{\min}/\alpha_1) = 240$). Die Abbildungen 5.10 und 5.11 auf den Seiten 116 und 117 zeigen die Entwicklung der besten und mittleren Fitnesswerte sowie der Populationsgrößen und Qualitätsmaße von zwei konkurrierenden Populationen mit unterschiedlichen Verbrauchsfaktoren.

Population P1 weist bis zur Generation 100 die bessere Qualität (Q) auf und behält ihre maximale Populationsgröße. Nach einer Stagnationsphase gelingt es der richtungssuchenden Population P2, der Orientierung der Fitnesslandschaft zu folgen. Aufgrund besserer Fitnesswerte steigt ihre Qualität. Ab Generation 200 übersteigt ihre Qualität die der anderen, so daß sie mehr Ressourcen zugeteilt bekommt. Da P2 einen höheren Verbrauchsfaktor aufweist als ihr Wettbewerber, ist ihr Zuwachs geringer als der Verlust der anderen. Die Größe der Gesamtpopulation (SUM) sinkt infolgedessen.

cBGA mit Strategiemodifikation

Die beiden oben spezifizierten cBGA verwenden statische Suchstrategien. Der Effekt des Wettbewerbs besteht ausschließlich in der Anpassung ihrer Populationsgrößen. Im Gegensatz dazu modifiziert dieses Wettbewerbsmodell durch das in Abschnitt 5.4.6 beschriebene Verfahren zur Anpassung der Schrittweitenparameter von Mutationsoperatoren die Suchstrategien der Populationen.

Spezifikation 16 *cBGA-Instanz mit Strategiemodifikation*

$$
\begin{aligned}
\mathsf{cbga3}^* ::= \mathbf{EA}(\ & \mathsf{POP}^{\mathrm{BGA,BM}}(16, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 2, 1.024 \cdot 10^{-3}) \\
& \mathsf{POP}^{\mathrm{BGA,BM}}(16, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 2, 1.024 \cdot 10^{-4}), \\
& \mathsf{POP}^{\mathrm{BGA,BM}}(16, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 2, 1.024 \cdot 10^{-5}), \\
& \mathsf{POP}^{\mathrm{BGA,BM}}(16, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 2, 1.024 \cdot 10^{-6}), \\
& \mathbb{W}^{\mathrm{WTA}}_{4,4^4}, \mathbb{V}^{\mathrm{BGA,RHO}}_{32,10.0}, \mathbb{K}^{\mathrm{B2A}}_{16}, \mathbb{T}^{\mathrm{EXP}}\ )
\end{aligned}
$$

Zur besseren Demonstration des Anpassungsprozesses sind die Schrittweitenparameter der BGA-Populationen bewußt zu niedrig initialisiert. Der niedrige Wert des Präzisionsparameters $k = 2$ erhöht die Fortschrittsgeschwindigkeit und erlaubt eine direktere Anpassung. Die Strategiemodifikation wird in jeder 32-sten Generation aufgerufen. Falls dann eine der Randpopulationen (P1 oder P4) die maximale Populationsgröße aufweist, werden die Schrittweitenparameter aller Populationen in Abhängigkeit der Randpopulation mit 0.1 oder 10.0 multipliziert. Die Strategiemodifikation findet damit nach jeweils vier Wettbewerbsintervallen bzw. zwei Migrationsintervallen statt. Diese externen Parameter sollten grundsätzlich so gewählt werden, daß das Wettbewerbsintervall kleiner als das Migrationsintervall und das Migrationsintervall kleiner als das Intervall zur Strategiemodifikation ist. Die Zusammenstellung der Strategien liegt letztlich in der Verantwortung des Benutzers.

Die Abbildungen 5.12 und 5.13 auf den Seiten 118 und 119 zeigen die Entwicklung der besten und mittleren Fitnesswerte sowie der Populationsgrößen und des Schrittweitenparameters von vier konkurrierenden Populationen. Die anfangs zu kleinen Schrittweitenparameter werden in den ersten 10 Generationen erhöht und, sobald die Zielannäherung es erfordert, wieder heruntergeregelt. Der Verlauf der besten Fitness wird aufgrund der feineren Anpassung gegenüber dem cBGA mit unterschiedlichen Schrittweitenparametern glatter.

cES mit unterschiedlicher Anzahl von Strategieparametern

In Abschnitt 3.3 wurden verschiedene Varianten der Evolutionsstrategie vorgestellt, die sich in der Anzahl der Strategieparameter unterscheiden. Die Güte der Varianten hängt von der Fitnesslandschaft des zu lösenden Problems ab. Für symmetrische Fitnessfunktionen, wie das Kugelmodell, ist die Evolutionsstrategie es1 mit einem Strategieparameter ausreichend. Falls die Objektvariablen, wie beim gewichteten Kugelmodell, dagegen unterschiedliche Gewichtungen aufweisen, ist die Evolutionsstrategie es2 mit n Strategieparametern notwendig. Fitnesslandschaften, die beliebige Orientierungen aufweisen, erfordern den Einsatz der Evolutionsstrategie es3 mit Rotationswinkeln.

Da die Eigenschaften des Optimierungsproblems in der Regel nicht bekannt sind, ist die Wahl der passenden Evolutionsstrategie schwierig. Statt die verschiedenen Evolutionsstrategien nacheinander anzuwenden, bietet das Wettbewerbsmodell die Möglichkeit, alle Varianten der Evolutionsstrategie in den Algorithmus aufzunehmen und gegeneinander konkurrieren zu lassen. Beispielhaft wird im folgenden ein Wettbewerb zwischen einer es1- und es2-Population modelliert:

Spezifikation 17 *cES mit unterschiedlicher Anzahl von Strategieparametern*

$$\begin{aligned}\mathsf{ces}^* ::= \mathbf{EA}(\ & \mathsf{POP}^{\mathrm{ES1}}(50, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}),\\ & \mathsf{POP}^{\mathrm{ES2}}(50, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}),\\ & \mathbb{W}^{\mathrm{WTA}}_{4,4^2}, \mathbb{K}^{\mathrm{B2A}}_{16}, \mathbb{T}^{\mathrm{EXP}}\ )\end{aligned}$$

Die Populationsgrößen beider Populationen sind zu Beginn $\lambda = 50$. Die Anzahl der Eltern, die bei dieser Selektion funktional von der jeweiligen Populationsgröße abhängig ist, ist anfangs $\mu = \lfloor\sqrt{50}\rfloor = 7$. In diesem Wettbewerbsmodell unterscheiden sich die beiden Populationen in der Anzahl ihrer Strategieparameter und damit in der Art ihrer internen Modelle, die sie von der Fitnesslandschaft bilden. Die damit verbundene Problematik, die bei der Migration von Individuen auftritt, wurde bereits in Abschnitt 5.4.5 aufgezeigt.

Die unterschiedlichen internen Modelle werden in diesem Wettbewerbsmodell bei einer Migration folgendermaßen ineinander überführt: Bei der Migration eines Individuums der es1-Population in die es2-Population wird der Wert seines Strategieparameters in alle n Strategieparameter des Immigranten kopiert. Im umgekehrten Fall, in dem ein Individuum aus der es2-Population in die es1-Population emigriert, wird der Mittelwert seiner n Strategieparameter gebildet und in den einen Strategieparameter des Immigranten kopiert.

In den Abbildungen 5.14 und 5.15 auf den Seiten 120 und 121 ist eine Simulation dieses Wettbewerbsmodells angewandt auf das gewichtete Kugelmodell dargestellt. In den ersten 300 Generationen dominiert die es1-Population. Nachdem die es2-Population ihre Strategieparameter an die Fitnesslandschaft angepaßt hat, verdrängt sie die ES1-Population und hält ihre Populationsgröße für einige hundert Generationen aufrecht. Ab der Generation 900 wechseln sich beide Populationen ständig ab.

cGA mit unterschiedlichen genetischen Operatoren

Der cGA enthält eine typische GA-Population mit Two-Point-Crossover und einer Mutationsrate $p_m = 0.001$ sowie eine Population ohne Rekombinationsoperator und einer dimensionsabhängigen Mutationsrate $p_m = 1/(30n)$. Die erste GA-Population erhält einen niedrigeren Verbrauchsfaktor als die mutationsbasierte Population.

Spezifikation 18 *cGA mit unterschiedlichen genetischen Operatoren*

$$\begin{aligned}\mathsf{cga} ::= \mathbf{EA}(\ & \mathsf{POP}^{\mathrm{GA1}}(224, G, 30, 0.001),\\ & \mathsf{POP}^{\mathrm{GA}}(\ \ 8, G, 30, \mathbb{S}^{\mathrm{GA,R}}_{1.1}, \Leftrightarrow, 1/(30n))\\ & \mathbb{W}^{\mathrm{WTA}}_{4,8^2,\{0.25,1.0\}}, \mathbb{K}^{\mathrm{B2A}}_{16}, \mathbb{T}^{\mathrm{EXP}})\end{aligned}$$

Die Abbildungen 5.16 und 5.17 auf den Seiten 122 und 123 zeigen die Entwicklung der beiden GA-Populationen. Zunächst behauptet sich die typische GA-Population (P1) und schöpft durch ihren Rekombinationsoperator die Varianz der Population aus. Nachdem die Varianz nach 400 Generationen aufgebraucht ist, übernimmt die zweite Population (P2), die ausschließlich den Mutationsoperator verwendet.

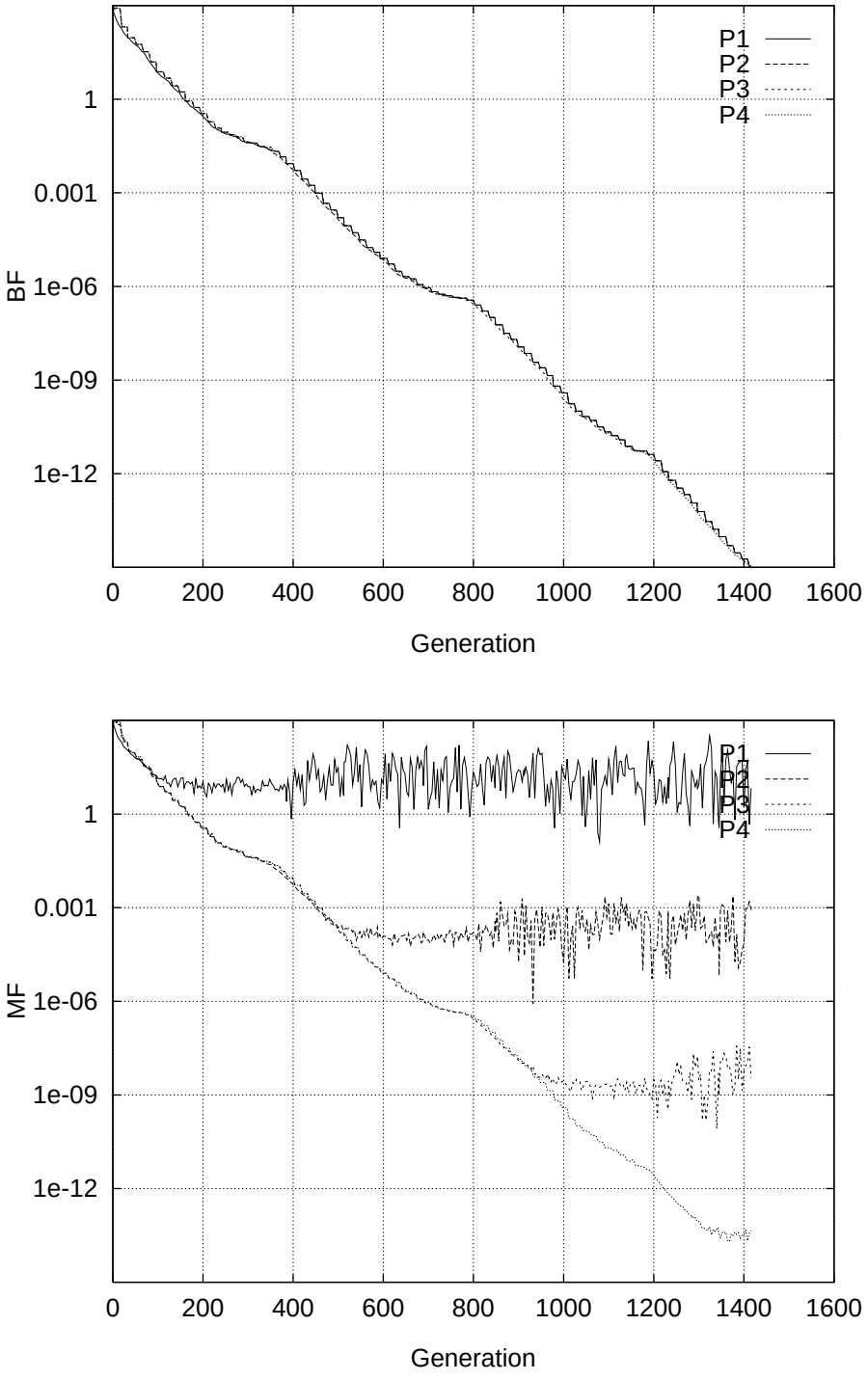

Abbildung 5.8: cBGA mit unterschiedlichen Schrittweitenparametern (Kugelmodell, n=100). Für einen Lauf ist die beste Fitness (BF) und die mittlere Fitness (MF) jeder Population dargestellt.

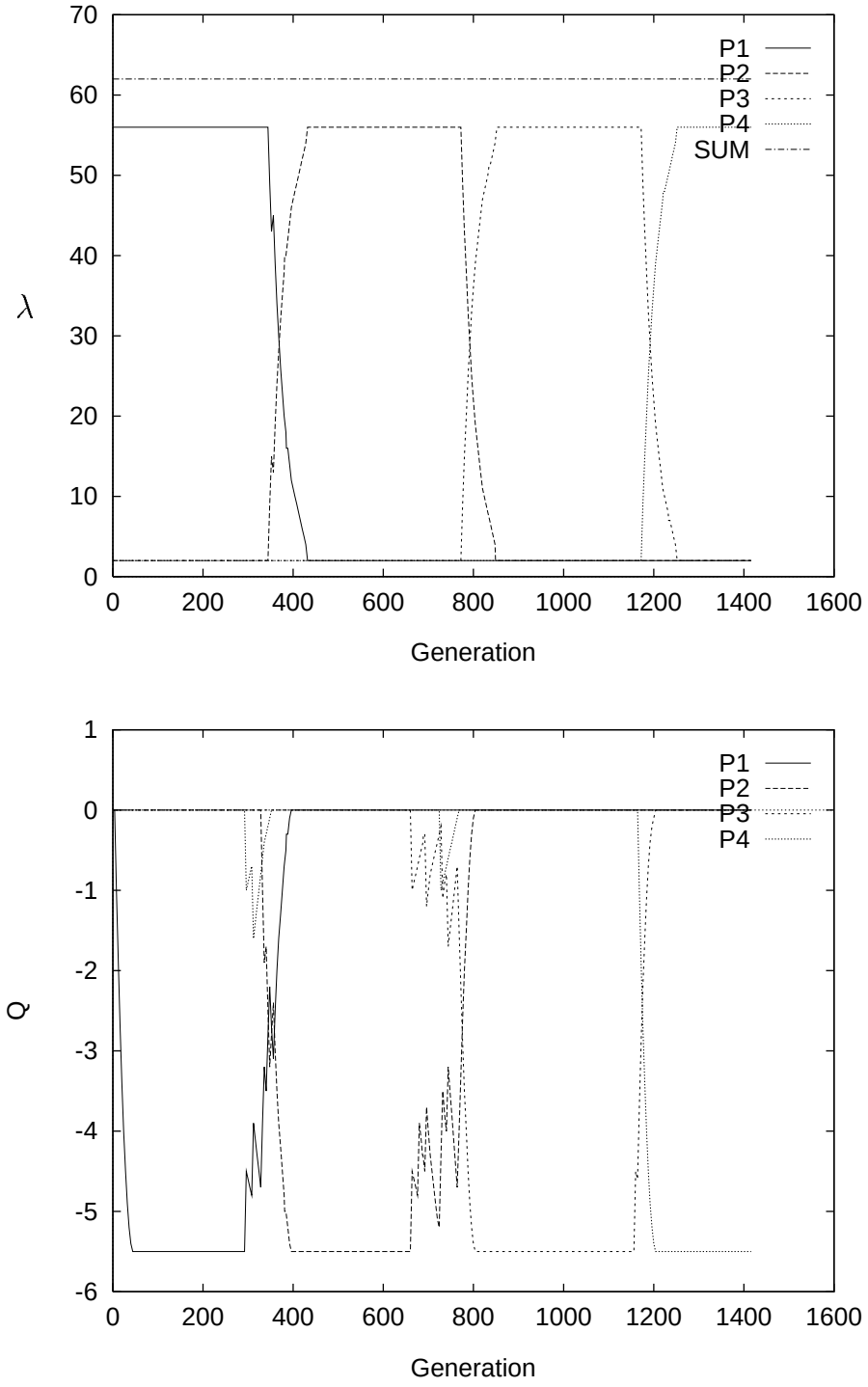

Abbildung 5.9: cBGA mit unterschiedlichen Schrittweitenparametern (Kugelmodell, n=100). Für einen Lauf ist die Populationsgröße (λ) und die Qualität (Q) jeder Population dargestellt. SUM bezeichnet die Gesamtzahl beider Populationen. Die Qualität ist in Analogie zur Fitness negativ eingezeichnet.

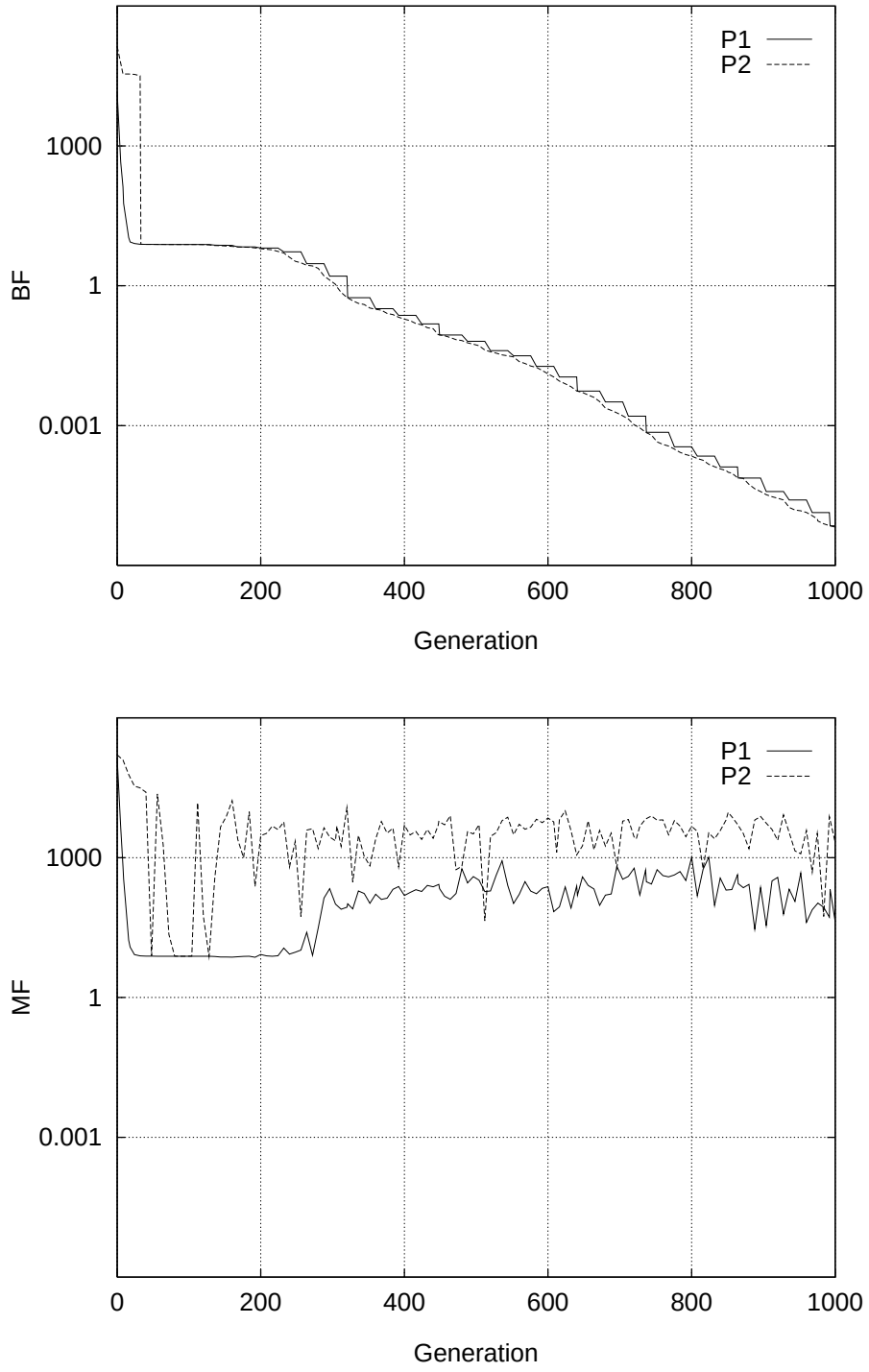

Abbildung 5.10: Wettbewerb zweier BGA-Populationen mit unterschiedlichen Verbrauchsfaktoren (Rosenbrocks Funktion, n=10): beste Fitness (BF) und mittlere Fitness (MF). Population P1 führt eine Breitensuche und Population P2 eine Richtungssuche aus.

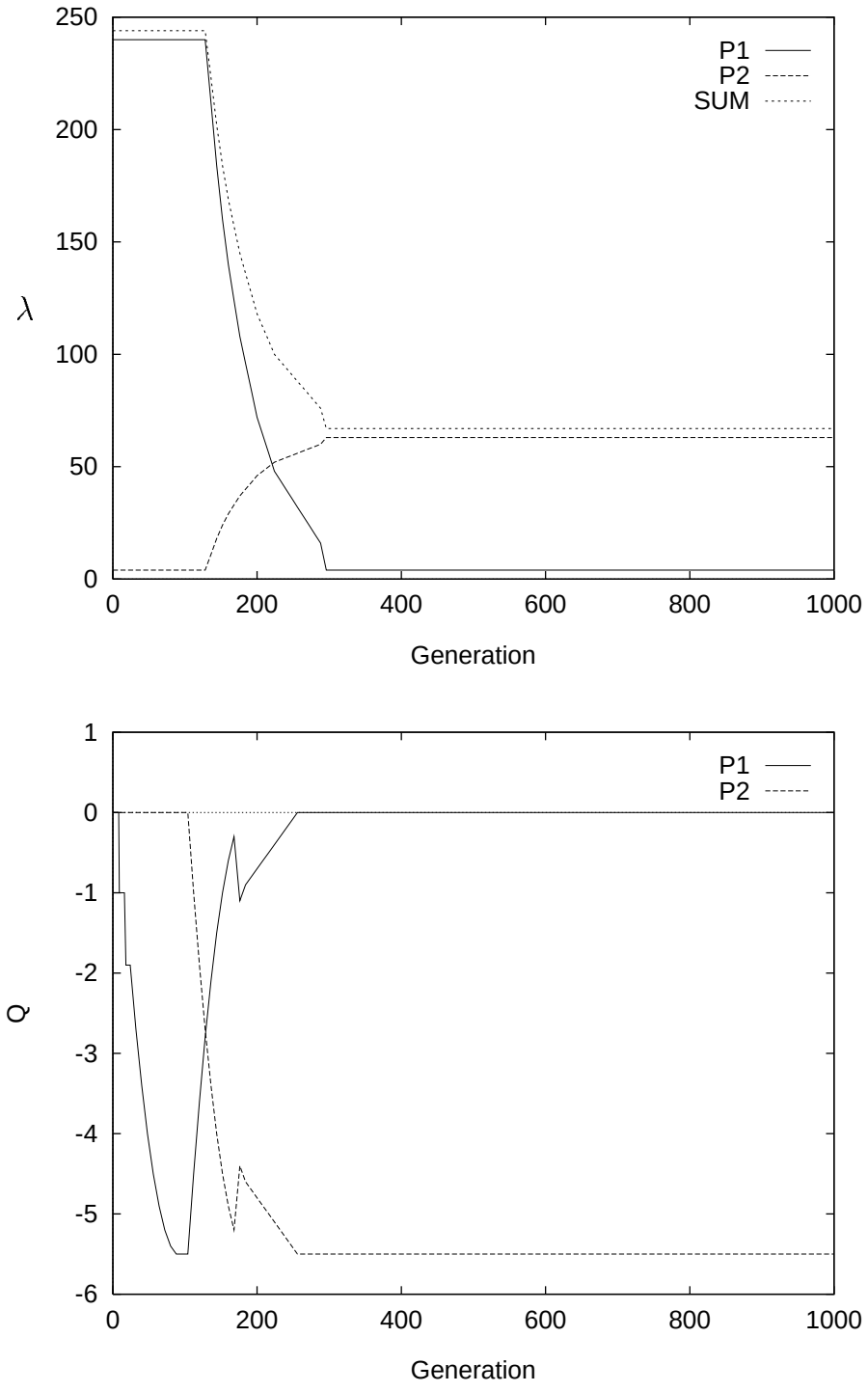

Abbildung 5.11: Wettbewerb zweier BGA-Populationen mit unterschiedlichen Verbrauchsfaktoren (Rosenbrocks Funktion, n=10): Populationsgröße (λ) und Qualität (Q).

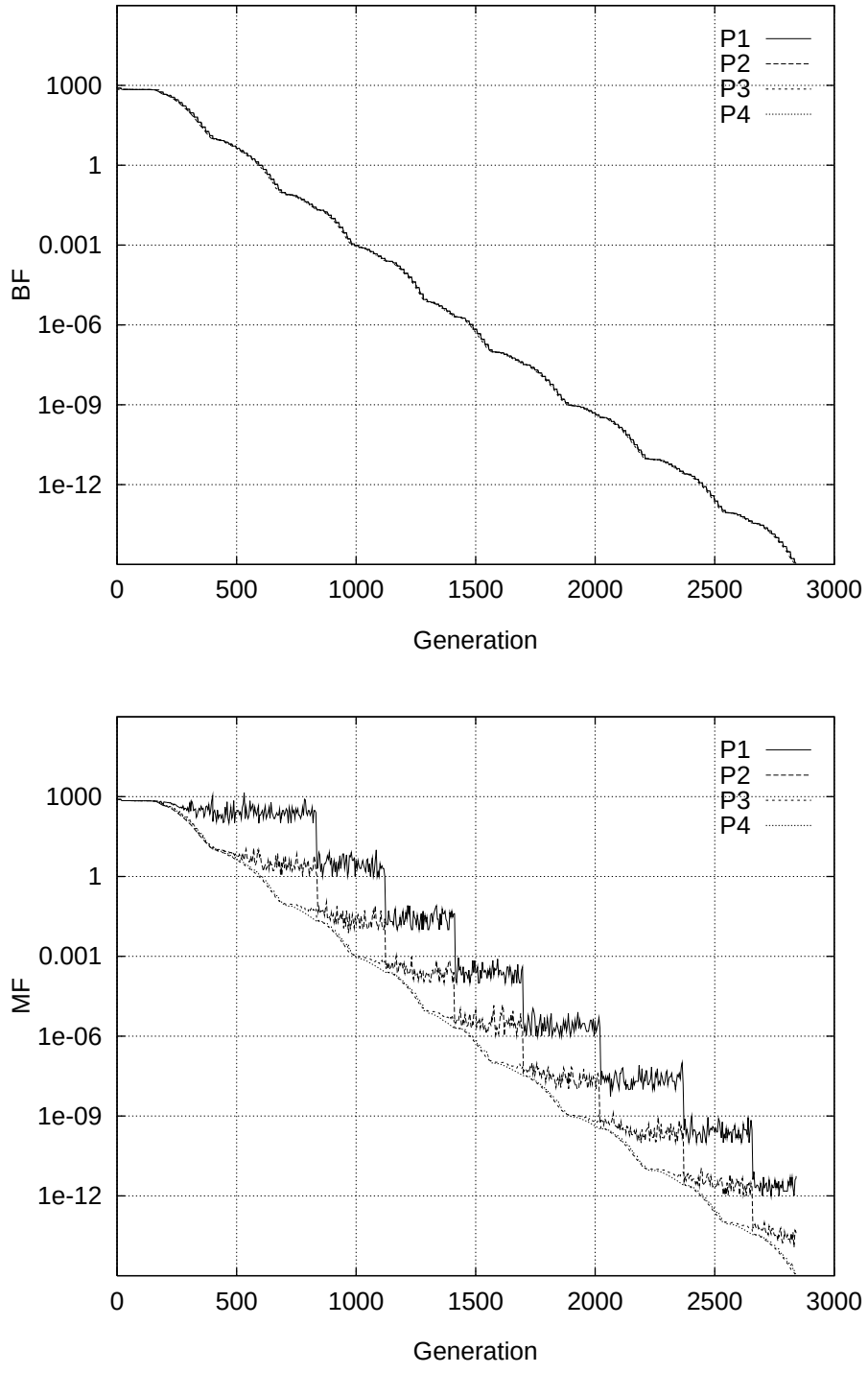

Abbildung 5.12: Wettbewerb mit Strategiemodifikation zur Anpassung der Schrittweitenparameter von BGA-Populationen (Kugelmodell, n=100): beste Fitness (BF) und mittlere Fitness (MF).

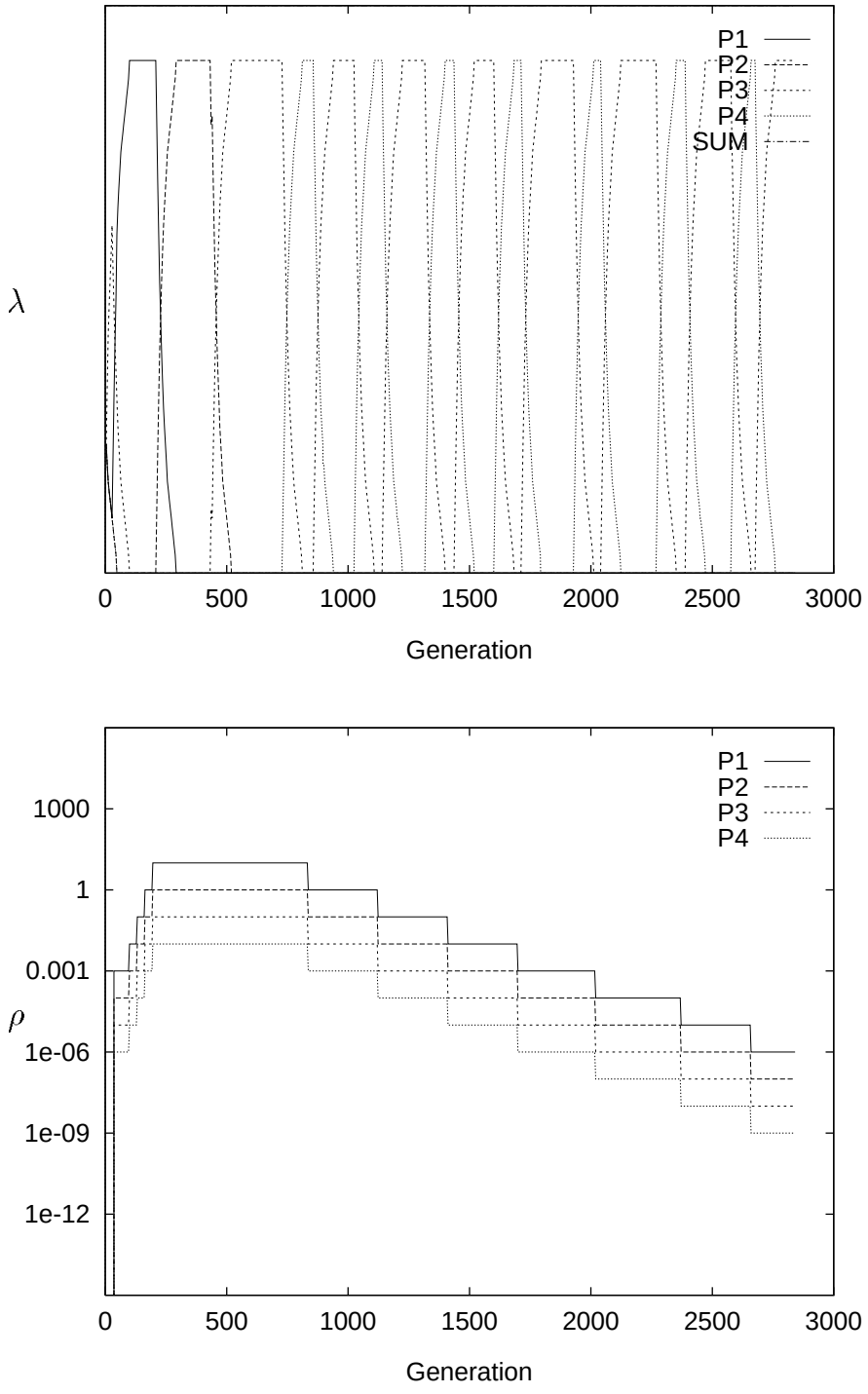

Abbildung 5.13: Wettbewerb mit Strategiemodifikation zur Anpassung der Schrittweitenparameter von BGA-Populationen (Kugelmodell, n=100): Populationsgröße (λ) und Schrittweitenparameter (ρ).

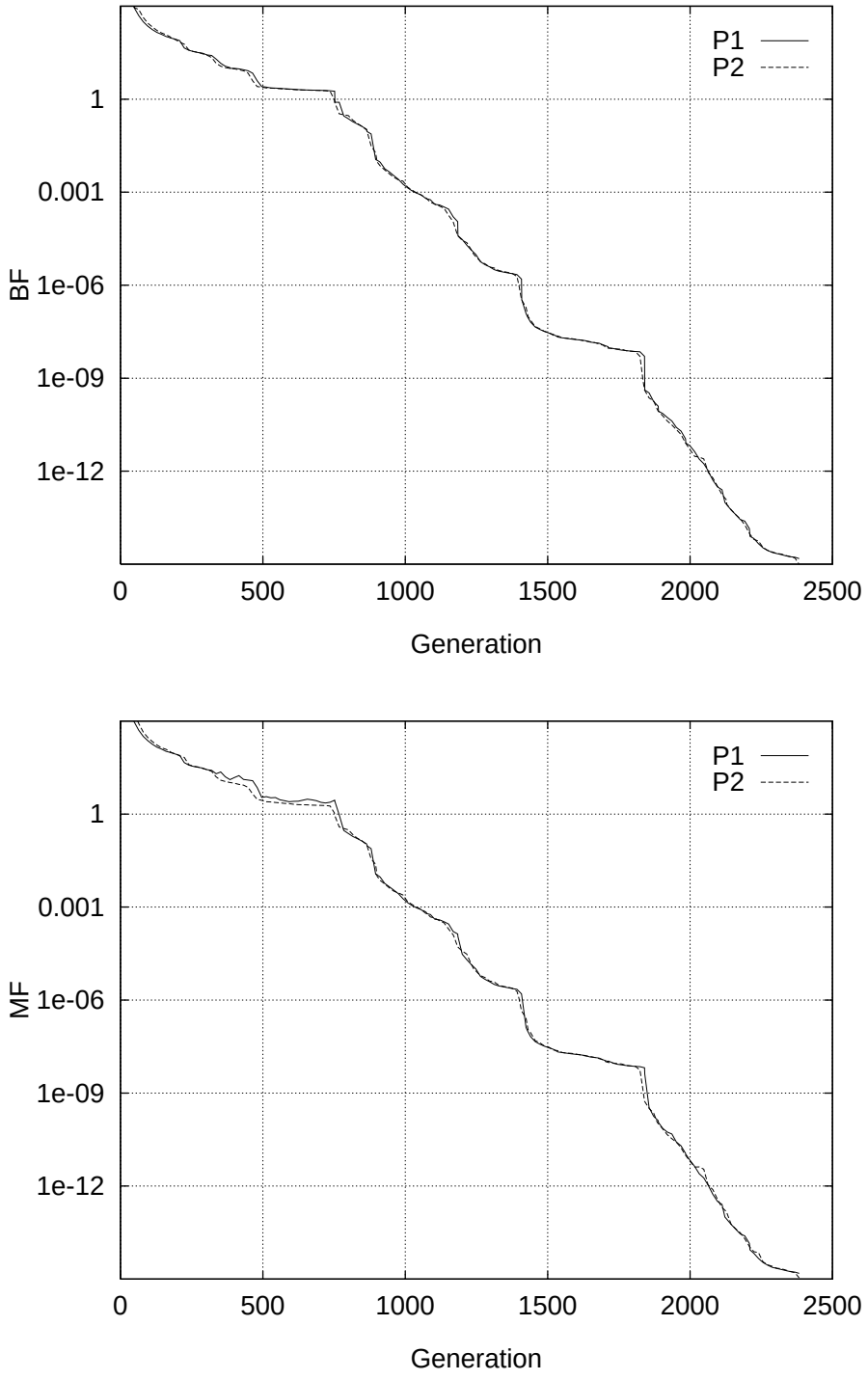

Abbildung 5.14: Wettbewerb zwischen einer es1- und es2-Population (gewichtetes Kugelmodell, n=100): beste Fitness (BF) und mittlere Fitness (MF). P1 bezeichnet die es1- und P2 die es2-Population.

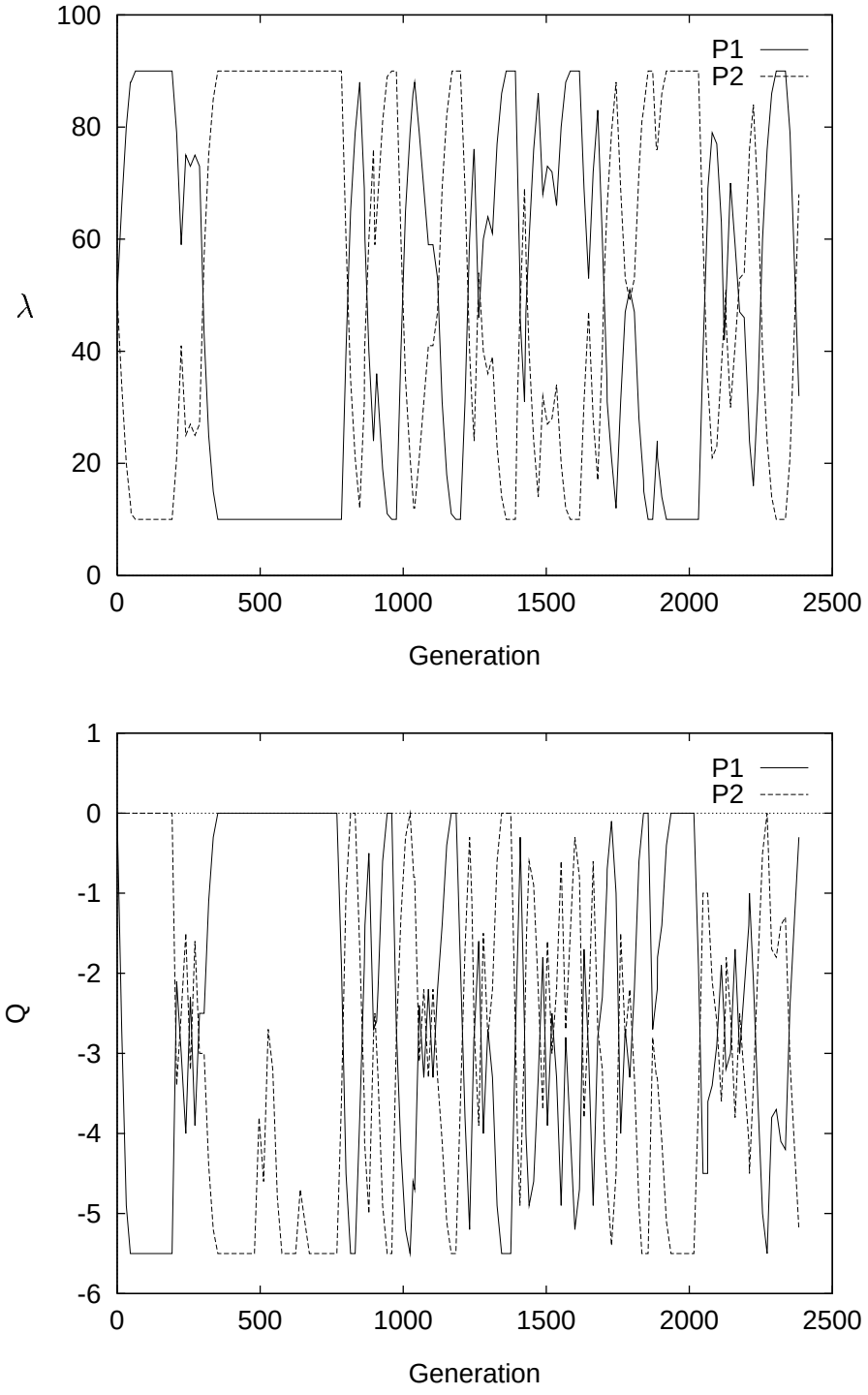

Abbildung 5.15: Wettbewerb zwischen einer es1- und es2-Population (gewichtetes Kugelmodell, n=100): Populationsgröße (λ) und Qualität (Q). P1 bezeichnet die es1- und P2 die es2-Population.

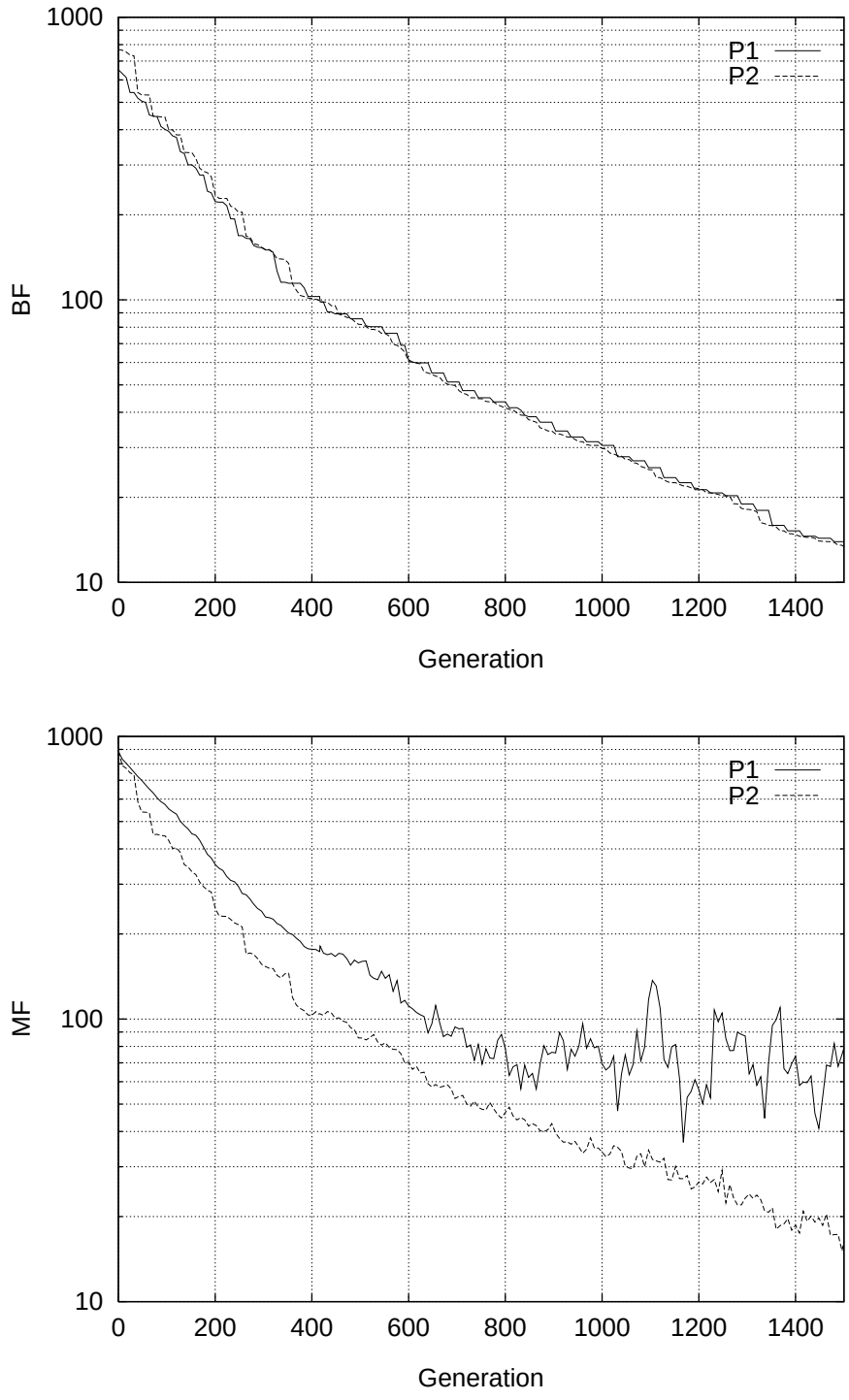

Abbildung 5.16: Zwei konkurrierende GA mit unterschiedlichen genetischen Operatoren (Kugelmodell, n=100): beste Fitness (BF) und mittlere Fitness (MF).

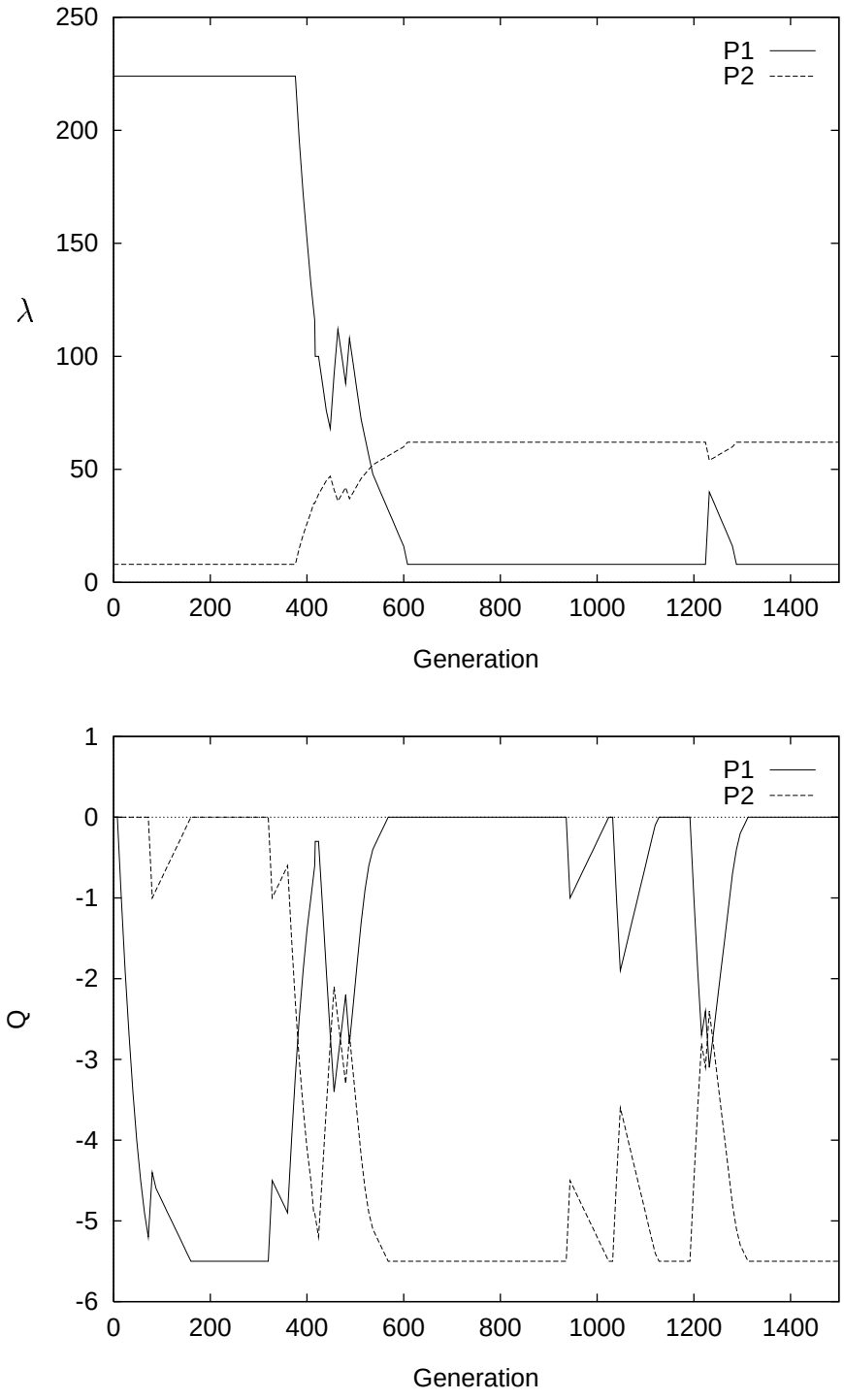

Abbildung 5.17: Zwei konkurrierende GA mit unterschiedlichen genetischen Operatoren (Kugelmodell, n=100): Populationsgröße (λ) und Qualität (Q).

Kapitel 6

Vergleich der Optimierungsverfahren

6.1 Auswertungsmethodik

Die Leistung eines Algorithmus kann analytisch durch Methoden der deduktiven Mathematik oder empirisch durch Simulationen untersucht werden. Die deduktive Vorgehensweise eignet sich nur zur Analyse einfacher Algorithmen. Komplexe Algorithmen, wie sie in der Praxis vorzufinden sind, erfordern in der Regel strenge, unrealistische Annahmen, die die Aussagekraft der theoretischen Analyse erheblich einschränken.

Aus diesem Grund kommt der empirischen Wissenschaft in der Bewertung von heuristischen Algorithmen eine besondere Bedeutung zu. Die empirische Vorgehensweise stellt hohe Anforderungen an Entwurf, Durchführung und Auswertung der Experimente. So kann eine fehlerhafte Implementierung oder die Verwendung unterschiedlicher Entwicklungsumgebungen zu wertlosen Resultaten führen.

Letztlich sollten beide Möglichkeiten ausgeschöpft werden. Auf diese Weise können sich analytische und empirische Betrachtungweisen ergänzen und gegenseitig validieren.

In diesem Kapitel werden die verschiedenen Suchverfahren mit Hilfe von Rechnersimulationen bewertet und miteinander verglichen. Um zu aussagekräftigen Resultaten zu gelangen, müssen die Experimente sorgfältig entworfen werden. Eine Hilfe bieten die vielen wissenschaftlichen Arbeiten, die sich mit dem Entwurf und der Darstellung von Experimenten durch Rechnersimulationen befaßt haben ([Hoo94], [Hoo95], [BGK+95]).

Entwurf von Experimenten

Ein Experiment soll die Leistung eines bestimmten Algorithmus durch Rechnersimulation für ein bestimmtes Problem zeigen. Die Resultate einer Simulation werden nicht nur durch den Algorithmus, sondern auch durch dessen Implementierung und Ausführung beeinflußt. Die Faktoren, die zur Simulation eines Algorithmus notwendig sind, werden unter dem Begriff *Testumgebung* zusammengefaßt.

Eine Möglichkeit, durch unterschiedliche Testumgebungen verursachte Schwankungen auszuschließen, besteht in einer Sichtweise, die den Algorithmus nicht ohne dessen konkrete Implementierung betrachtet (*algorithm-cum-implementation*). Als Spezifikation des Algorithmus wird in diesem Fall nur die Implementierung des Algorithmus zugelassen. Da diese Sichtweise die Allgemeingültigkeit von Simulationsergebnissen erheblich einschränkt, muß eine Trennung zwischen Phänomen (Algorithmus) und Apparat (Testumgebung) vorgenommen werden. Um die Auswirkungen der Testumgebung auf die Resultate gering zu

halten, sollten sie für empirische Vergleiche verschiedener Algorithmen möglichst konstant gehalten werden. Der Entwurf von Experimenten läßt sich also in die folgenden drei Kategorien unterteilen:

- **Problem:** Die Auswahl der Testmenge ist von zentraler Bedeutung für die Aussagekraft eines Experiments. Da das ultimative Ziel heuristischer Methoden in der Lösung realer Probleme liegt, stellen Instanzen des realen Problems die beste Testmenge dar. Da die Bearbeitung realer Problemstellungen häufig zu rechenintensiv ist, können auch repräsentative künstliche Probleme verwendet werden. Künstliche Probleme eignen sich besonders, um bestimmte Merkmale des Algorithmus auszutesten. Zudem können die Eigenschaften künstlicher Probleme besser gesteuert werden.

- **Algorithmus:** Der zu untersuchende Algorithmus muß exakt beschrieben werden. Besonderer Wert muß auch auf vermeintlich nebensächliche Komponenten wie Initialisierung und Terminierung gelegt werden. Zudem muß die verwendete Parametrisierung vollständig angegeben werden.

- **Testumgebung:** Idealerweise sollten die zu vergleichenden Algorithmen von demselben Experten implementiert und auf demselben Rechner ausgeführt werden. Auf diese Weise können Unterschiede, die das Verhalten eines Algorithmus erheblich beeinflussen können, weitgehend vermieden werden. Verschiedene Rechner können sich in Art und Anzahl der Prozessoren, Cache und Hauptspeicher oder den Kommunikationsfähigkeiten unterscheiden. Auf Softwareebene beeinflussen Betriebssystem, Programmiersprache, Compiler, Compileroptionen und verwendete Datenstrukturen eine Simulation. Zudem wirken sich auch Programmierfähigkeiten und Erfahrung des Programmierers auf die Ergebnisse der Experimente aus.

Darstellung der Ergebnisse

Für die Beurteilung eines Algorithmus müssen zunächst geeignete Bewertungskriterien und Leistungsmaße eingeführt werden. Ein heuristisches Suchverfahren läßt sich unter folgenden Gesichtspunkten betrachten.

- **Schnelligkeit:** Rechenaufwand, der für das Erreichen einer bestimmten Lösungsqualität oder Terminierungsbedingung notwendig ist.

- **Genauigkeit:** Beste erzielte Lösung bis zur Terminierung.

- **Robustheit:** Sensitivität gegenüber algorithmusspezifischen Parametereinstellungen. Eine Heuristik, die über einen weiten Bereich von Parameterwerten ein gutes Verhalten zeigt, ist einer Heuristik, deren Parameter für jedes Problem angepaßt werden müssen, überlegen.

- **Einfachheit:** Ein Algorithmus, der einfach zu implementieren ist, reduziert das Risiko einer fehlerhaften Implementierung und ist daher anderen bei gleicher Leistung vorzuziehen.

- **Wichtigkeit:** Der Wert eines Algorithmus steigt mit der Relevanz der Problemklasse, auf die er anwendbar ist.

- **Allgemeinheit:** Größe der Problemklasse, auf die der Algorithmus anwendbar ist.
- **Innovativität:** Durch neue und kreative Ideen werden andere Forscher inspiriert, die diese Verfahren weiterentwickeln und auf andere Problemklassen anwenden.

Die genannten Eigenschaften lassen sich in quantitative und qualitative Merkmale unterteilen. Zu den quantifizierbaren Eigenschaften werden im folgenden Leistungsmaße eingeführt, die zur Bewertung und zum Vergleich der verschiedenen Suchverfahren verwendet werden.

Interpretation der Ergebnisse

Schließlich müssen die erzielten Resultate eines Experiments analysiert und interpretiert werden. Dabei sollten vorhandene Tradeoffs und Faktorkombinationen, die die Leistungsfähigkeit eines Algorithmus beschreiben, herausgearbeitet werden.

- Lösungsqualität vs. Rechenaufwand
- Geschwindigkeit vs. Robustheit
- Rechenaufwand vs. Problemgröße (Skalierung)
- Robustheit vs. Lösungsqualität

Als Vergleich sollte immer das *beste* konkurrierende und/oder ein möglichst einfaches heuristisches Verfahren gewählt werden.

6.2 Künstliche Testfunktionen

Der Vergleich verschiedener Optimierungsverfahren erfordert die Zusammenstellung einer Menge von repräsentativen Testfunktionen. Die Güte eines Suchverfahrens kann letztlich nur bezüglich der zugrundeliegenden Menge von Testfunktionen beurteilt werden. Aus diesem Grunde sollte die Menge der Testfunktionen ein möglichst breites Spektrum an funktionstypischen Eigenschaften abdecken. Zu diesen Eigenschaften zählen im wesentlichen die Art der Interaktion zwischen Objektvariablen sowie die Anzahl und Anordnung lokaler Optima.

Die einfachste Funktionenklasse ist die Klasse der sogenannten *separierbaren* Funktionen. Da zwischen den Variablen einer separierbaren Funktion keine Interaktionen existieren, kann der optimale Wert jeder Variablen unabhängig von den Werten der restlichen Variablen bestimmt werden. Aufgrund ihrer einfachen Skalierbarkeit werden separierbare Funktionen häufig als Testfunktionen verwendet.

Neben der Art der Interaktion zwischen Variablen wird der Schwierigkeitsgrad einer Funktion auch durch den relativen Beitrag der einzelnen Variablen zur Zielfunktion beeinflußt. Je unterschiedlicher die Beiträge der verschiedenen Variablen sind, desto aufwendiger ist die Steuerung der Schrittweiten des Optimierungsverfahrens.

Die Modalität ist ein weiteres Kriterium zur Klassifikation von Testfunktionen. Man unterscheidet hier zwischen unimodalen und multimodalen Funktionen. Multimodale Funktionen weisen im Gegensatz zu unimodalen Funktionen neben dem globalen Optimum mindestens ein weiteres lokales Optimum auf. Der Schwierigkeitsgrad einer Funktion hängt

mehr von der Anordnung und der Größe der Attraktorgebiete der lokalen Optima als von deren Anzahl ab. Eine multimodale Funktion, die eine globale unimodale Struktur mit einer Vielzahl kleiner lokaler Optima aufweist, gleicht einer verrauschten unimodalen Funktion und ist entsprechend einfach zu lösen. Demgegenüber kann eine multimodale Funktion mit wenigen aber weit auseinanderliegenden ähnlichen lokalen Optima eine äußerst schwierige Optimierungsaufgabe darstellen. Schließlich spielt die Anordnung der lokalen Optima eine wichtige Rolle für den Schwierigkeitsgrad einer Testfunktion. Testfunktionen, deren lokale Optima in einer regulären Struktur angeordnet sind, können durch spezialisierte Optimierungsverfahren, die genau diese Regularität ausnutzen (diskrete Rekombination in EA), äußerst effizient gelöst werden. Aus diesem Grund sollten lokale Optima von Fitnessfunktionen zufällig angeordnet werden.

Die in dieser Arbeit untersuchten Testfunktionen enthalten alle oben angesprochenen Schwierigkeitsgrade und stellen somit einen repräsentativen Querschnitt durch die in der Optimierung verwendeten Testfunktionen dar. Die Funktionen reichen von dem einfachen und leicht zu analysierenden Kugelmodell über die nichtseparierbaren Funktionen von Schwefel und Rosenbrock bis zu der multimodalen Funktion von Fletcher und Powell, deren lokale Optima zufällig im Suchraum angeordnet sind.

6.2.1 Kugelmodell (F1)

Das Kugelmodell ist eine unimodale, stetige, streng konvexe Funktion, die in den theoretischen Untersuchungen der Evolutionären Algorithmen eine zentrale Rolle spielt. Diese Funktion dient aufgrund ihrer Einfachheit weniger dazu, die Leistungsfähigkeit eines Optimierungsverfahrens zu demonstrieren, als vielmehr dazu, dessen Verhalten zu studieren. So kann für Optimierungsverfahren, die mit variablen Mutationsschrittweiten arbeiten, die optimale Schrittweite berechnet und mit der tatsächlichen verglichen werden.

$$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2 \tag{6.1}$$

$x_i^{\min}$	$x_i^{\max}$	f^*	$\vec{x}^*$
-5.12	5.12	0.0	$(0,\ldots,0)^T$

6.2.2 Gewichtetes Kugelmodell (F2)

Das gewichtete Kugelmodell ist, wie das Kugelmodell, eine unimodale, stetige, streng konvexe Funktion, die durch die unterschiedliche Gewichtung der einzelnen Variablen eine erste Schwierigkeit aufweist. Aufgrund dieser unterschiedlichen Gewichtung reicht eine gemeinsame Schrittweite für alle Variablen je nach Art der Mutationsverteilung oft nicht aus.

$$f_2(\vec{x}) = \sum_{i=1}^{n} i \cdot x_i^2 \tag{6.2}$$

$x_i^{\min}$	$x_i^{\max}$	f^*	$\vec{x}^*$
-5.12	5.12	0.0	$(0,\ldots,0)^T$

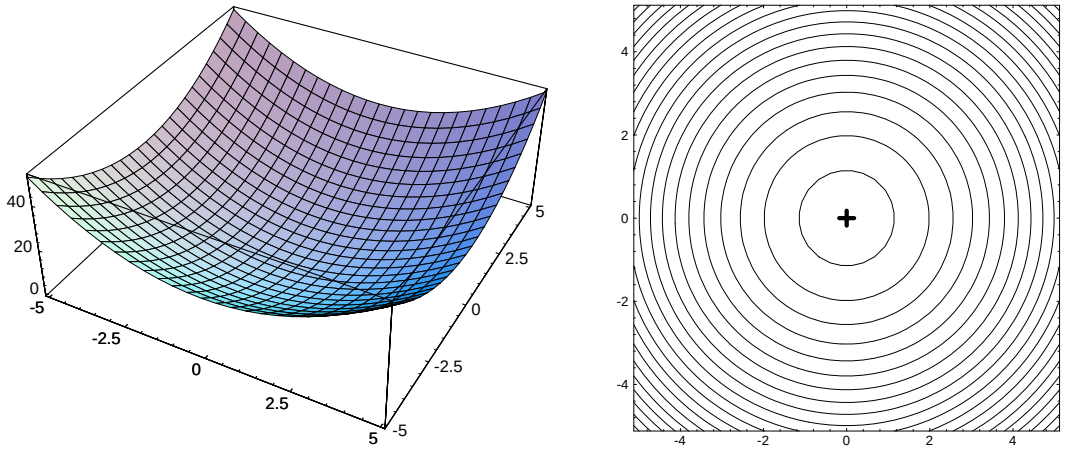

Abbildung 6.1: Kugelmodell (F1)

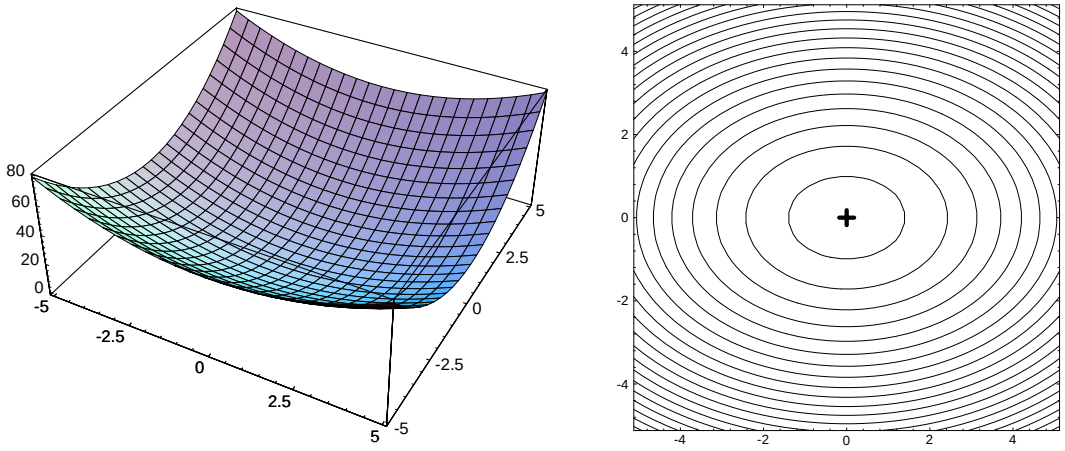

Abbildung 6.2: Gewichtetes Kugelmodell (F2)

6.2.3 Schwefels Funktion (F3)

Dieses quadratische Minimierungsproblem wurde von Schwefel als Problem 1.2 eingeführt ([Sch81], [Sch95]). Obwohl die Funktion, wie auch die beiden Kugelmodelle, unimodal und streng konvex ist, können die Variablen aufgrund der Korrelationen nicht mehr separat eingestellt werden. Die Funktion hat die angenehme Eigenschaft, daß ihre Auswertungszeit trotz der quadratischen Eigenschaften nur linear mit der Dimension n steigt.

$$f_3(\vec{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \tag{6.3}$$

$x_i^{\min}$	$x_i^{\max}$	f^*	$\vec{x}^*$
-65	65	0.0	$(0,\ldots,0)^T$

Die Matrixnotation dieses Problems ist:

$$f(\vec{x}) = \vec{x}^T \mathbf{A} \vec{x}$$

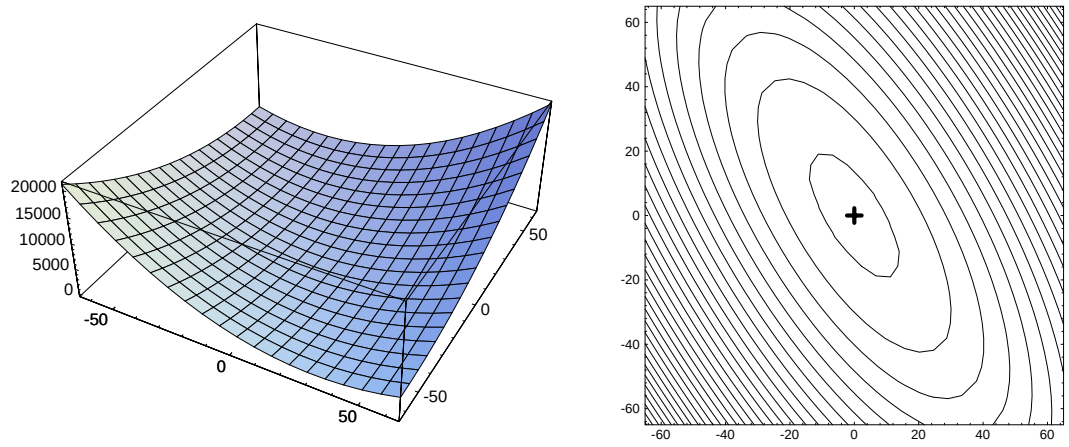

Abbildung 6.3: Schwefels Funktion (Fsch)

Die $n \times n$ Formmatrix $\mathbf{A}$ ist symmetrisch und positiv definit. Als Maß für die numerische Schwierigkeit dieses Problems dient die *Konditionszahl* K, die als Verhältnis zwischen größtem und kleinstem Eigenwert der Matrix definiert ist ([BRS66]). Schwefel hat die Konditionszahlen für verschiedene Dimensionen berechnet ([Sch95]). Die Konditionszahl K wächst mit steigender Dimension in der Ordnung $O(n^2)$.

n	K	K/n^2
1	1.00	1.00
2	6.85	1.71
3	16.40	1.82
6	64.90	1.80
10	175.00	1.75
20	678.00	1.69
30	1500.00	1.67
60	5930.00	1.65
100	16400.00	1.64

6.2.4 Rosenbrocks Funktion (F4)

Diese Funktion stellt eine Verallgemeinerung der von Rosenbrock eingeführten Funktion für den n-dimensionalen Fall dar ([Ros60]). Die Schwierigkeit dieser Funktion besteht in dem gekrümmten Verlauf der Höhenlinien. Neben dem Optimum im Punkt $(1, \ldots, 1)$ hat diese Funktion um den Punkt $(1, 0, \ldots, 0)$ ein starkes Attraktorgebiet, in dem zwar kein lokales Minimum liegt, aus dem aber nur ein schmales gekrümmtes Tal herausführt. Zwischen benachbarten Objektvariablen besteht bei dieser Funktion eine stärkere Korrelation als zwischen entfernteren.

$$f_4(x) = \sum_{i=1}^{n-1} \left(100(x_{i+1} \Leftrightarrow x_i^2)^2 + (1 \Leftrightarrow x_i)^2 \right) \tag{6.4}$$

$x_i^{\min}$	$x_i^{\max}$	f^*	$\vec{x}^*$
-5.12	5.12	0.0	$(1,\ldots,1)^T$

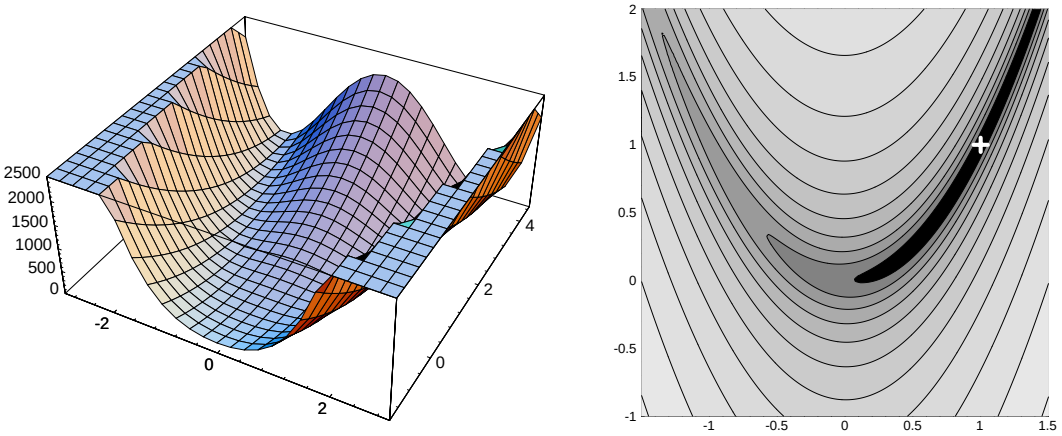

Abbildung 6.4: Rosenbrocks Funktion (F4)

6.2.5 Ackleys Funktion (F5)

Ackleys Funktion ist eine durch schwache Cosinuswellen überlagerte Exponentialfunktion. Die Amplitude der Cosinusfunktion ist so gewählt, daß die unimodale Makrostruktur der Funktion erhalten bleibt ([Ack87], S. 13-14). Die lokalen Minima sind in einem regulären Gitter angeordnet. Die ursprünglich zweidimensionale Funktion ist hier in einer verallgemeinerten mehrdimensionalen Form angegeben.

$$f_5(x) = \Leftrightarrow 20 \exp\left(\Leftrightarrow 0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) \Leftrightarrow \exp\left(\frac{1}{n}\sum_{i=1}^{n} cos\,(2\pi x_i)\right) + 20 + e \qquad (6.5)$$

$x_i^{\min}$	$x_i^{\max}$	f^*	$\vec{x}^*$
-30	30	0.0	$(0,\ldots,0)^T$

6.2.6 Flechter und Powells Funktion (F6)

Fletcher und Powell entwickelten 1966 mit Hilfe trigonometrischer Funktionen ein multimodales Problem, dessen lokale Minima zufällig angeordnet sind ([FP63], [Pow64a]). Diese Funktion ist ein typisches Problem der nichtlinearen Parameterschätzung (Regression), welches ebenfalls von Schwefel ([Sch77] und Bäck ([Bäc96]) zur Untersuchung der Evolutionsstrategie verwendet wurde.

Die Funktion weist im Gegensatz zu den meisten anderen künstlichen Testfunktionen keine Symmetrie oder Regularität auf. Ihre lokalen Optima sind zufällig im Suchraum angeordnet. Deren Positionen werden durch die Zufallsmatrizen $\mathbf{A} = (a_{ij})$ und $\mathbf{B} = (b_{ij})$ spezifiziert. Der Zufallsvektor $\vec{\alpha} = \alpha_i$ definiert die Lage des globalen Optimums.

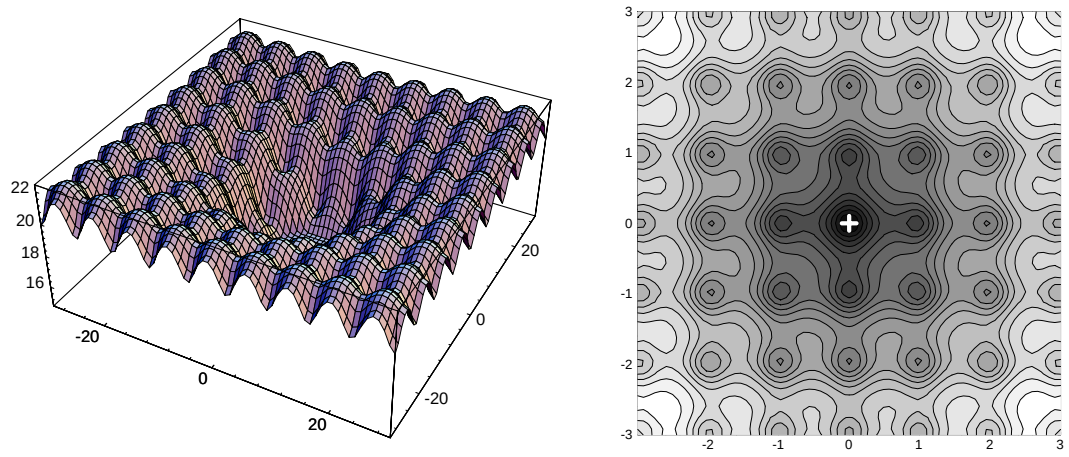

Abbildung 6.5: Ackleys Funktion (F5)

$$f_6(x) = \sum_{i=1}^{n} (A_i \Leftrightarrow B_i(x)) \tag{6.6}$$

$$A_i = \sum_{j=1}^{n} (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j) \tag{6.7}$$

$$B_i(x) = \sum_{j=1}^{n} (a_{ij} \sin x_j + b_{ij} \cos x_j) \tag{6.8}$$

mit

$$a_{ij}, b_{ij} \in [\Leftrightarrow 100, 100]; \alpha_i \in [\Leftrightarrow \pi, \pi]. \tag{6.9}$$

$x_i^{\min}$	$x_i^{\max}$	f^*	$\vec{x}^*$
$\Leftrightarrow\pi$	π	0.0	$(\alpha_1, \ldots, \alpha_n)^T$

Zusammen ergeben sich für $n = 30$ also 1830 Zufallszahlen. Im Anhang A sind die Zufallszahlen aufgeführt, die Bäck in seinen Experimenten verwendete ([Bäc96]).

6.3 Spezifikation der Algorithmen

Die zu vergleichenden Suchverfahren werden mit Hilfe der EA-Syntax aus Abschnitt 3.1.4 spezifiziert. Die Auswahl der Algorithmen besteht aus den in Kapitel 2 eingeführten lokalen Suchverfahren sowie den verschiedenen Evolutionären Algorithmen und Wettbewerbsmodellen. Bei allen Simulationen wird standardmäßig das folgende Terminierungskriterium verwendet:

$$\mathbb{T}^{\text{EXP}} ::= \mathbb{T}^{\text{EA,IMP}}_{100,10^{-15}} \tag{6.10}$$

Die Suche wird beendet, falls nach 100 Generationen keine Verbesserung des Funktionswertes um mindestens 10^{-15} erzielt wurde.

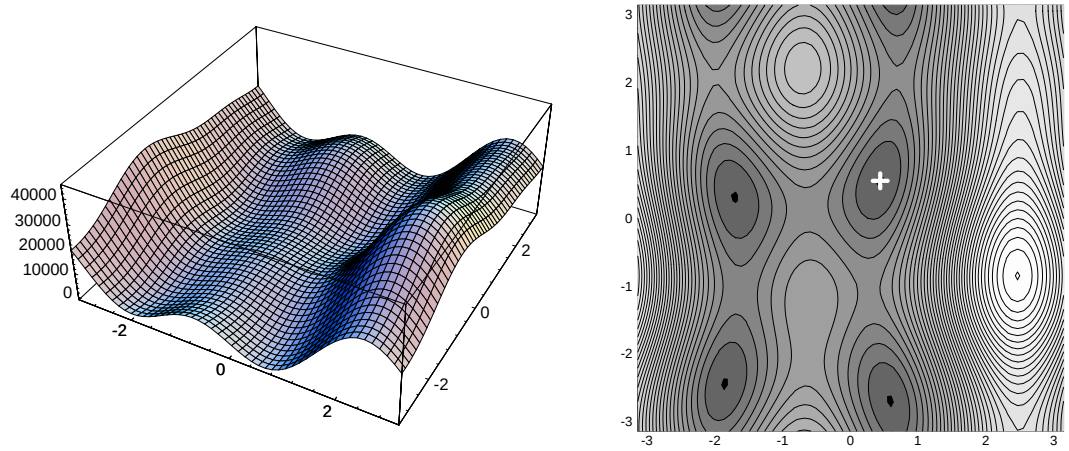

Abbildung 6.6: Funktion von Flechter und Powell (F6)

Iterierte Local Hillclimber

Bei *iterierten Local Hillclimbern* wird das lokale Suchverfahren wiederholt aufgerufen. Im Unterschied zu den sogenannten *Multistart-Hillclimbern*, die jede neue Iteration von einem neuen, zufällig generierten Punkt starten, beginnt die Suche eines iterierten Hillclimbers mit dem Resultat des vorherigen Laufes. Während die in Kapitel 2 eingeführten lokalen Suchverfahren die Suche aufgrund ihrer degenerierten internen Modelle häufig zu früh abbrechen, erfolgt hier mit jeder Iteration eine neue Initialisierung der internen Modelle, so daß die Suche von dem gefundenen Punkt fortgesetzt werden kann.

Ein Iterated Hillclimber läßt sich durch die EA-Syntax spezifizieren, indem eine Population, die aus nur einem Individuum besteht, definiert wird. Eine Generation entspricht einer Iteration des lokalen Suchverfahrens.

Spezifikation 19 $\mathsf{POP}^{\mathsf{LHC}}$ – *LHC-Population*

$$\mathsf{POP}^{\mathsf{LHC}}(\Lambda, \Theta_{\mathbb{L}}) \quad ::= \quad \mathbf{POP}\left(\mathbf{IND}\left(\mathbb{C}^{\mathrm{EA}}\right)^{1}, \mathbb{P}^{\mathrm{EA,S}}, \mathbb{S}^{\mathrm{EA,TR}}_{1.0,p}, \mathbb{L}^{\Lambda}_{\Theta_{\mathbb{L}}}, \mathbb{I}^{\mathrm{EA,U}}\right)$$

Hierbei bezeichnet $\mathbb{L}^{\Lambda}_{\Theta_{\mathbb{L}}}$ das durch Λ spezifizierte lokale Suchverfahren und $\Theta_{\mathbb{L}}$ dessen externe Parameter. Als lokale Suchverfahren werden die in Kapitel 2 vorgestellten ableitungsfreien Optimierungsverfahren verwendet. Die externen Parameter entsprechen den Parametern der dort spezifizierten Algorithmen.

Spezifikation 20 *LHC-Instanzen*

*Die lokalen Suchverfahren sind das Quasi-Newton-Verfahren (*qn*), das Powell-Verfahren (*pow*), das Downhill-Simplex-Verfahren (*dhs*), das Verfahren von Solis und Wets (*sw*) und das Dynamic-Hillclimbing-Verfahren (*dhc*).*

$$\mathsf{qn} ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{LHC}}\left(\mathsf{QN}, 10^9, 10^{-6}, \mathsf{BFGS}\right), \mathbb{T}^{\mathrm{EXP}}\right)$$

$$\mathsf{pow} ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{LHC}}\left(\mathsf{POW}, 10 \cdot n, 10^{-16}\right), \mathbb{T}^{\mathrm{EXP}}\right)$$

$$\mathsf{dhs} ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{LHC}}\left(\mathsf{DHS}, 1, 0.5, 2, \frac{x_i^{max} - x_i^{min}}{100}, 10^5, 10^{-12}\right), \mathbb{T}^{\mathrm{EXP}}\right)$$

$$\mathsf{sw} ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{LHC}}\left(\mathsf{SW}, \frac{x_i^{max} - x_i^{min}}{100}, 10^{-12}, 2.0, 0.5, 5, 3\right), \mathbb{T}^{\mathrm{EXP}}\right)$$

$$\mathsf{dhc} ::= \mathbf{EA}\left(\mathsf{POP}^{\mathsf{LHC}}\left(\mathsf{DHC}, 10^{-16}\right), \mathbb{T}^{\mathrm{EXP}}\right)$$

Evolutionsstrategien

Die ES-Instanz es1 entspricht einer Evolutionsstrategie mit nur einem Strategieparameter für alle Objektvariablen, die ES-Instanz es2 einer Evolutionsstrategie mit einem Strategieparameter für jede Objektvariable.

Spezifikation 21 *ES-Instanzen*

$$\mathsf{es1} ::= \mathsf{ES1}(15, 100, \mathbb{T}^{\mathrm{EXP}})$$

$$\mathsf{es2} ::= \mathsf{ES2}(15, 100, \mathbb{T}^{\mathrm{EXP}})$$

Genetische Algorithmen

Die GA-Instanz wird von der in Spezifikation 6 eingeführten GA-Population abgeleitet. Sie entspricht der GA-Variante in GENESIS mit rangbasierter Selektion.

Spezifikation 22 *GA-Instanz*

$$\mathsf{ga} ::= \mathsf{GA1}(100, \mathbb{T}^{\mathrm{EXP}})$$

Breeder Genetic Algorithms

Der Standard-BGA bga arbeitet mit der BGA-Mutation und der *Fuzzy-Rekombination*. Die Instanz bga-pca verwendet ausschließlich den BGA-Mutationsoperator mit Hauptachsentransformation und ist so in der Lage, sich an Orientierungen in der Fitnesslandschaft anzupassen. Aufgrund seiner Einfachheit wird das Mutations-Selektions-Verfahren bga-1+1 zum Vergleich mit den aufwendigeren BGA herangezogen. Da der bga-1+1 aus nur einem Individuum besteht, wird die Terminierungsbedingung entsprechend angepaßt.

Spezifikation 23 *BGA-Instanzen*

$$\mathsf{bga} \quad ::= \quad \mathsf{BGA}^{\mathsf{FR}}\,(64, 0.125, 32, \tilde{\rho}, \mathbb{T}^{\mathrm{EXP}})$$

$$\mathsf{bga\text{-}pca} \quad ::= \quad \mathsf{BGA}^{\mathsf{PCA}}\,(64, 0.125, 32, \tilde{\rho}, \mathbb{T}^{\mathrm{EXP}})$$

$$\mathsf{bga\text{-}1{+}1} \quad ::= \quad \mathsf{BGA}^{\mathsf{BM}}\left(1, 1.0, 32, \tilde{\rho}, \mathbb{T}^{\mathrm{EA,IMP}}_{6400,10^{-15}}\right)$$

Evolutionäre Algorithmen mit Wettbewerbsmodell

cBGA mit unterschiedlichen Schrittweitenparametern. Aufgrund der in Abschnitt 4.4 durchgeführten Untersuchungen der BGA-Mutationsverteilung, die den Einfluß der externen Parameter Schrittweitenparameter ρ und Präzisionsparameter k deutlich machen, wurde in Spezifikation 14 bereits die Instanz eines Wettbewerbsmodells eingeführt, dessen Populationen mit unterschiedlichen Schrittweitenparametern arbeiten. Durch die Verwendung verschiedener Schrittweitenparameter kann der Präzisionsparameter reduziert werden, ohne das Mutationsintervall einzuschränken. Die Reduzierung des Präzisionsparameters k führt wiederum zu einer erhöhten Konvergenzgeschwindigkeit.

Im folgenden wird eine entsprechende Klasse des Wettbewerbsmodells für eine beliebige Anzahl von Populationen spezifiziert. Die daraus abgeleitete Instanz cbga1 besteht aus vier Populationen.

Spezifikation 24 *cBGA-Instanz mit unterschiedlichen Schrittweitenparametern*

$$\mathsf{cBGA}^{\mathsf{rho}}(\kappa, \vec{\rho}, k) \quad ::= \quad \mathbf{EA}\left(\left\{\mathsf{POP}^{\mathrm{BGA,FR}}\left(\tfrac{\lambda}{\kappa}, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, k, \rho_i\right)\right\}_{i=1}^{\kappa}, \mathbb{W}^{\mathrm{WTA}}_{4,4^{\kappa}}, \mathbb{K}^{\mathrm{B2A}}_{16}, \mathbb{T}^{\mathrm{EXP}}\right)$$

$$\mathsf{cbga1} \quad ::= \mathsf{cBGA}^{\mathsf{rho}}(4, \vec{\rho}, 9)$$

Die Schrittweitenparameter ρ_i bis ρ_κ sind von dem Definitionsbereich des Problems abhängig und werden nach dem in Gleichung 5.31 angegebenen Verfahren berechnet.

cBGA mit unterschiedlichen Suchstrategien. Dieses Wettbewerbsmodell entspricht dem in Spezifikation 15 eingeführten cBGA, dessen Populationen unterschiedliche Verbrauchsfaktoren aufweisen. Durch die unterschiedlichen Verbrauchsfaktoren kann die Populationsgröße an die jeweilige Suchstrategie angepaßt werden. Die Population $\mathsf{POP}^{\mathrm{BGA,FR}}$, deren Strategie Breitensuchcharakter hat, kann durch ihren niedrigen Verbrauchsfaktor sehr viel mehr Individuen als die Population mit richtungsorientierter Suchstrategie enthalten.

Die folgende Spezifikation unterscheidet sich von Spezifikation 15 lediglich im Verhältnis ihrer Verbrauchsfaktoren und damit in den Populationsgrößen.

Spezifikation 25 *cBGA-Instanz mit unterschiedlichen Suchstrategien*

$$\begin{aligned}\mathsf{cbga2} \quad ::= \quad \mathbf{EA}\big(\ & \mathsf{POP}^{\mathsf{BGA,FR}}(960, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT},4.0}, 32, \tilde{\rho}) \\ & \mathsf{POP}^{\mathsf{BGA,BLR}}(\ \ 4, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT},1.0}, 32, \tilde{\rho}), \\ & \mathbb{W}^{\mathrm{WTA}}_{4,\{16,4\},\{0.0625,1.0\}}, \mathbb{K}^{\mathrm{B2A}}_{16}, \mathbb{T}^{\mathrm{EXP}})\end{aligned}$$

cBGA mit Strategiemodifikation. Dieses Wettbewerbsmodell, das bereits in Spezifikation 16 eingeführt wurde, verwendet die Strategiemodifikation zur Anpassung der Schrittweitenparameter.

Spezifikation 26 *cBGA-Instanz mit Strategiemodifikation*

$$\begin{aligned}\mathsf{cbga3} \quad ::= \quad \mathbf{EA}\big(\ & \mathsf{POP}^{\mathrm{BGA,FR}}(16, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 8, \tilde{\rho}/8^0) \\ & \mathsf{POP}^{\mathrm{BGA,FR}}(16, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 8, \tilde{\rho}/8^1), \\ & \mathsf{POP}^{\mathrm{BGA,FR}}(16, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 8, \tilde{\rho}/8^2), \\ & \mathsf{POP}^{\mathrm{BGA,FR}}(16, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}, 8, \tilde{\rho}/8^3), \\ & \mathbb{W}^{\mathrm{WTA}}_{4,4^4,\mathbb{V}^{\mathrm{BGA,RHO}}_{32,8.0}}, \mathbb{K}^{\mathrm{B2A}}_{16}, \mathbb{T}^{\mathrm{EXP}})\end{aligned}$$

cES mit unterschiedlicher Anzahl von Strategieparametern. In diesem Wettbewerbsmodell konkurrieren zwei Evolutionsstrategien mit unterschiedlicher Anzahl von Strategieparametern. Das Wettbewerbsmodell entspricht dem in Spezifikation 17 beschriebenen Wettbewerbsmodell.

Spezifikation 27 *cES mit unterschiedlicher Anzahl von Strategieparametern*

$$\begin{aligned}\mathsf{ces} \quad ::= \quad \mathbf{EA}\big(\ & \mathsf{POP}^{\mathrm{ES1}}(50, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}), \\ & \mathsf{POP}^{\mathrm{ES2}}(50, \mathbb{S}^{\mathrm{cEA,TR}}_{\mathrm{SQRT}}), \\ & \mathbb{W}^{\mathrm{WTA}}_{4,4^2}, \mathbb{K}^{\mathrm{B2A}}_{16}, \mathbb{T}^{\mathrm{EXP}})\end{aligned}$$

6.4 Testumgebung

Die Wahl und Spezifikation der Testumgebung ist, neben der Auswahl der Optimierungsprobleme und Algorithmen, für den Entwurf und die Durchführung eines Experiments von entscheidender Bedeutung. Die Simulation der oben spezifizierten Algorithmen erfolgt, wie in Abschnitt 6.1 motiviert, in einer homogenen Entwicklungsumgebung. Sämtliche Experimente werden mit der Simulationsumgebung PeGAsuS gemacht. PeGAsuS ist eine an der GMD – Forschungszentrum Informationstechnik – entwickelte Simulationsumgebung für Evolutionäre Algorithmen, in die die wichtigsten Evolutionären Algorithmen sowie die in Kapitel 2 vorgestellten lokalen Suchverfahren integriert sind ([SV93]).

Die Evolutionären Algorithmen entsprechen den bekannten Implementierungen ihrer Entwickler. Die in PeGAsuS integrierten Evolutionsstrategien wurden mit Schwefels Implementierung OptimA verifiziert ([Sch95]). Die Genetischen Algorithmen entsprechen der GENESIS 5.0-Implementierung von Grefenstette ([Gre90]).

Die deterministischen lokalen Suchverfahren stammen aus *Numerical Recipes in C* ([PTVF92]). Die Implementierung des Verfahrens von Solis und Wets basiert auf deren Veröffentlichung und wurde nachimplementiert ([SW81]). Der Quellcode des Dynamic-Hillclimbing-Verfahrens stammt von dessen Entwicklern Yuret und de la Maza.

Die Simulationen werden auf verschiedenen Rechnerarchitekturen gemacht. Zu diesen zählen IBM-SP2, Parsytec Explorer und SUN-Workstation und PC. Um sicherzustellen, daß die Resultate, die auf den unterschiedlichen Rechnern erzielt werden, konsistent sind, werden ausgewählte Experimente auf allen Plattformen ausgeführt und deren Resultate verglichen.

6.5 Leistungsmaße

6.5.1 Konvergenzverhalten

Iterative Optimierungsverfahren lassen sich durch ihr Konvergenzverhalten charakterisieren und vergleichen. Das Konvergenzverhalten, das Effizienz und Robustheit eines Verfahrens beschreibt, wird in *Konvergenzgeschwindigkeit* (engl.: *convergence velocity*) und *Konvergenzsicherheit* (engl.: *convergence reliability*) unterteilt.

Bevor Aussagen über die Annäherung an das Optimum gemacht werden können, muß die zugrundeliegende Metrik definiert werden. Als Abstandsmaß kann sowohl der Funktionswert (Froschperspektive) als auch der euklidische Abstand zum Optimum (Vogelperspektive) verwendet werden. Im ersten Fall wird der Qualitätsgewinn, im zweiten Fall die Annäherung an das Optimum als Fortschritt aufgefaßt. Für eine bestimmte Klasse von Funktionen lassen sich beide Metriken ineinander überführen. Der erwartete Fehler bezüglich beider Metriken ist definiert als

$$\boxed{\delta_t^x := E\left(\|\vec{x}_t \Leftrightarrow \vec{x}^*\|\right)} \tag{6.11}$$

und

$$\boxed{\delta_t^f := E\left(f(\vec{x}_t) \Leftrightarrow f^*\right).} \tag{6.12}$$

Da der Funktionswert bei der Simulation eines Evolutionären Algorithmus in jedem Fall vorliegt, wird dieser im folgenden auch als Metrik zugrunde gelegt. Der erwartete Fehler bezüglich des Funktionswertes wird von nun ab mit δ_t bezeichnet.

Die Konvergenzsicherheit eines Optimierungsverfahrens beschreibt dessen Fähigkeit, sich dem Optimum eines Problems bis auf eine bestimmte Genauigkeit anzunähern. Da Aussagen über die Konvergenzsicherheit eines Suchverfahrens nur für eine eingeschränkte Klasse von Funktionen möglich sind und diese im allgemeinen unter der Annahme unendlich vieler Iterationen gemacht werden, sind sie für die Praxis wenig hilfreich. Aus diesem Grund wird in diesem Abschnitt der Schwerpunkt auf den für den Anwender interessanteren Aspekt des Konvergenzverhaltens, die Konvergenzgeschwindigkeit, gelegt. Weitere Überlegungen zur Konvergenzsicherheit sind im nächsten Abschnitt, der die Robustheit von Suchverfahren behandelt, zu finden.

Konvergenzordnung

Lokale Konvergenzaussagen über die Konvergenzgeschwindigkeit können unter der Annahme gemacht werden, daß sich der Suchpunkt $\vec{x}_t$ in der Nachbarschaft des lokalen Optimums $\vec{x}^*$ im Laufe der Suche monoton an $\vec{x}^*$ annähert. Zur Beschreibung der Konvergenzgeschwindigkeit eines Suchverfahrens bedient man sich Modellen, die den funktionalen Zusammenhang zwischen dem Fehler und der Zeit wiedergeben. Häufig werden anstelle der Zeit auch abstraktere Maße für den Aufwand, wie Anzahl der Funktionsauswertungen oder Anzahl der Generationen, verwendet.

Zur groben Klassifikation von Optimierungsverfahren bezüglich ihrer Konvergenzgeschwindigkeit wurde der Begriff der *Konvergenzordnung* eingeführt ([Fle87]). Die Konvergenzordnung unterteilt die möglichen Funktionen, die die Entwicklung des Fehlers über die Zeit beschreiben, in verschiedene Funktionsklassen.

Definition 17 (Konvergenzordnung) *Falls sich der Fehler, der für $t \to \infty$ gegen Null geht, wie*

$$\frac{\delta_{t+1}}{\delta_t^p} \to a, \qquad mit \quad a > 0 \tag{6.13}$$

verhält, hat das Verfahren eine Konvergenz p-ter Ordnung.

Die Konvergenz erster Ordnung (*lineare Konvergenz*) läßt sich als Ungleichung wie folgt ausgedrücken:

$$\frac{\delta_{t+1}}{\delta_t} \leq a \qquad \text{oder} \qquad \delta_{t+1} = O(\delta_t) \tag{6.14}$$

Die Konvergenz zweiter Ordnung (*quadratische Konvergenz*) ist

$$\frac{\delta_{t+1}}{\delta_t^2} \leq a \qquad \text{oder} \qquad \delta_{t+1} = O(\delta_t^2) \tag{6.15}$$

Man spricht von *sublinearer* Konvergenz, falls in Definition 17 $p < 1$ ist. Die Konvergenzordnung eines Verfahrens, für das $p > 1$ ist, wird als *superlinear* bezeichnet. Die Erfahrung läßt vermuten, daß stochastische Suchverfahren, die keine Ableitung verwenden, im besten Falle linear konvergieren. Eines der wenigen Verfahren mit quadratischer Konvergenzordnung ist der bekannte Newton-Raphson-Algorithmus, dessen hohe Konvergenzgeschwindigkeit allerdings nur für einen eingeschränkten Problemkreis erreicht wird.

Da stochastische Suchverfahren aufgrund ihrer Zufallskomponente mehr oder weniger starken Schwankungen unterworfen sind, ist die obige Definition der Konvergenzordnung zu verallgemeinern. Anstelle der Relation von zwei aufeinanderfolgenden Fehlern verwendet man für stochastische Verfahren die Entwicklung des Erwartungswertes des Fehlers über die Zeit. Aus der obigen Definition der Konvergenzordnung lassen sich für spezielle Konvergenzordnungen entsprechende Konvergenzordnungen für stochastische Verfahren ableiten.

Definition 18 (Stochastische lineare Konvergenz) *Ein Suchverfahren konvergiert* stochastisch linear, *falls sich der Erwartungswert des Fehlers δ_t relativ zum Anfangsfehler δ_0 folgendermaßen entwickelt:*

$$\frac{\delta_t}{\delta_0} \leq a^t \qquad mit \quad 0 < a < 1 \tag{6.16}$$

Definition 19 (Stochastische quadratische Konvergenz) *Ein Suchverfahren konvergiert* stochastisch quadratisch, *falls sich der Erwartungswert des Fehlers δ_t relativ zum Anfangsfehler δ_0 entwickelt wie:*

$$\frac{\delta_t}{\delta_0^{2^t}} \leq a^{2^t-1} \qquad mit \quad 0 < a < 1 \tag{6.17}$$

Rappl führte zur Klassifikation von stochastischen Verfahren den Begriff der *Konvergenzrate* ein ([Rap84]), der später von Bäck und Schwefel zur Klassifikation der Evolutionsstrategie verwendet wurde ([SB95]). Die Definition der Konvergenzrate basiert, ebenso wie die der stochastischen Konvergenzordnung, auf der Entwicklung des Fehlers über die Zeit. Aufgrund des funktionalen Zusammenhangs zwischen Fehler und Zeit lassen sich die Suchverfahren mit Hilfe von bestimmten Funktionsklassen klassifizieren. Im folgenden wird die *exponentielle* und die *polynomielle* Konvergenzrate definiert.

Definition 20 (Konvergenzrate) *Ein Optimierungsverfahren hat eine* exponentielle *(oder* geometrische*)* Konvergenzrate, *falls sich der Fehler verhält wie:*

$$\delta_t = O(\beta^t) \qquad mit \quad \beta \in (0, 1] \tag{6.18}$$

Ein Optimierungsverfahren hat eine polynomielle Konvergenzrate, *falls sich der Fehler verhält wie:*

$$\delta_t = O(t^{-b}) \qquad mit \quad b \in (0, 1]. \tag{6.19}$$

Die polynomielle Konvergenzrate wird auch als *sublineare* Konvergenzrate, und die exponentielle als *geometrische* Konvergenzrate bezeichnet.

Aus der Konvergenzordnung bzw. der Konvergenzrate des Algorithmus und der dazugehörigen Ratenkonstante läßt sich die Anzahl der Schritte abschätzen, die erforderlich sind, um bei einem gegebenen Anfangsfehler eine gewünschte Zielannäherung zu erreichen.

Satz 12 *Gegeben ist ein Startpunkt, der δ_0 vom Optimum entfernt ist. Ein Verfahren mit linearer Konvergenz und Ratenkonstante a benötigt*

$$\boxed{t \approx \Leftrightarrow \frac{1}{\ln a} \ln \frac{\delta_0}{\epsilon}} \tag{6.20}$$

Schritte, um den Abstand zum Optimum auf ϵ zu reduzieren.

Beweis 1 *Setzt man ϵ in Definition 18 ein, ergibt sich als obere Abschätzung:*

$$\begin{aligned}
& \frac{\epsilon}{\delta_0} \approx a^t \\
\Leftrightarrow \quad & \epsilon \approx a^t \cdot \delta_0 \\
\Leftrightarrow \quad & \Leftrightarrow t \cdot \ln a \approx \ln \frac{\delta_0}{\epsilon} \\
\Leftrightarrow \quad & t \approx \Leftrightarrow \frac{1}{\ln a} \ln \frac{\delta_0}{\epsilon}
\end{aligned}$$

∎

Satz 13 *Gegeben ist ein Startpunkt, der δ_0 vom Optimum entfernt ist. Ein Verfahren mit quadratischer Konvergenz und Ratenkonstante a benötigt*

$$\boxed{t \approx \ln\left(\frac{\ln(a \cdot \epsilon)}{\ln(a \cdot \delta_0)}\right)} \tag{6.21}$$

Schritte, um den Abstand zum Optimum auf ϵ zu reduzieren.

Beweis 2 *Definition 19 für stochastische quadratische Konvergenz liefert für t eine obere Abschätzung:*

$$\begin{aligned}
& \frac{\epsilon}{\delta_0^{2^t}} \approx a^{2^t-1} \\
\Leftrightarrow \quad & \epsilon \approx a^{2^t-1}\delta_0^{2^t} \\
\Leftrightarrow \quad & \epsilon \approx \frac{1}{a}(a \cdot \delta_0)^{2^t} \\
\Leftrightarrow \quad & a \cdot \epsilon \approx (a \cdot \delta_0)^{2^t} \\
\Leftrightarrow \quad & \ln(a \cdot \epsilon) \approx 2^t \ln(a \cdot \delta_0) \\
\Leftrightarrow \quad & 2^t \approx \frac{\ln(a \cdot \epsilon)}{\ln(a \cdot \delta_0)} \\
\Leftrightarrow \quad & t \approx \ln\left(\frac{\ln(a \cdot \epsilon)}{\ln(a \cdot \delta_0)}\right)
\end{aligned}$$

■

Satz 14 *Gegeben ist ein Startpunkt, der δ_0 vom Optimum entfernt ist. Ein Verfahren mit polynomieller Konvergenzrate und Ratenkonstante a benötigt*

$$\boxed{t \approx \sqrt[b]{a \cdot \frac{\delta_0}{\epsilon}}} \tag{6.22}$$

Schritte, um den Abstand zum Optimum auf ϵ zu reduzieren.

Beweis 3 *Nach Definition 20 erfüllt ein Verfahren mit polynomieller Konvergenzrate folgende Gleichung:*

$$\frac{\delta_t}{\delta_0} \leq a \cdot t^{-b}$$

Ersetzt man δ_t durch ϵ ergibt sich als obere Abschätzung:

$$\begin{aligned}
& \frac{\epsilon}{\delta_0} \approx a \cdot t^{-b} \\
\Leftrightarrow \quad & t^b \approx a \cdot \frac{\delta_0}{\epsilon} \\
\Leftrightarrow \quad & t \approx \sqrt[b]{a \cdot \frac{\delta_0}{\epsilon}}
\end{aligned}$$

■

Praktische Konvergenzmaße

Da Aussagen über die Konvergenzordnung eines Algorithmus nicht für beliebige Problemklassen möglich sind, werden analytische Betrachtungen der Konvergenzordnung üblicherweise auf die Klasse der streng konvexen Probleme beschränkt ([Rap84],[BS95]). Strenge Konvexität impliziert angenehme Eigenschaften wie Unimodalität und stetige Differenzierbarkeit.

Die bisherigen Betrachtungen zur Konvergenzgeschwindigkeit legen ein asymptotisches Verhalten der Suchverfahren zugrunde. Falls der Fehler gegen Null geht, ist die Konvergenzordnung ein eindeutiges Maß für die Qualität des Algorithmus. In der Praxis, in der das Optimierungsziel oft nur mit einer gewissen Genauigkeit erreicht werden muß, gibt die Konvergenzordnung nur einen ersten Eindruck. Von der Größe der Ratenkonstanten hängt letztlich die Anwendbarkeit eines Algorithmus ab. Die erforderliche Anzahl der Funktionsauswertungen kann für eine bestimmte Konvergenzordnung erheblichen Schwankungen unterworfen sein. So kann ein Verfahren mit sublinearer Konvergenzordnung aufgrund günstigerer Ratenkonstanten in einem gewissen Bereich wesentlich besser sein als ein Verfahren mit linearer oder gar superlinearer Konvergenzordnung.

Ferner ist zu beachten, daß das Konvergenzverhalten eines Algorithmus von dem Suchbereich abhängen kann. In diesem Fall muß die Konvergenzordnung eines Suchverfahrens bezüglich eines festgelegten Suchbereichs betrachten werden. Ebenso kann der funktionale Zusammenhang zwischen Fehler und Rechenaufwand so beschaffen sein, daß er sich durch keines der einfachen Modelle (Funktionen) beschreiben läßt. Die Konvergenzordnung ist also ein Maß, das in der Praxis aufgrund seiner strengen Annahmen, die die Problemklasse und den Bereich der Suche betreffen, vorsichtig verwendet werden muß.

Da der in Abschnitt 6.5.1 eingeführte Begriff der Konvergenzordnung zum einen ein asymptotisches Verhalten des Optimierungsverfahrens voraussetzt und zum anderen zufällige Schwankungen, die in heuristischen Methoden auftreten, nicht berücksichtigt, werden in der Praxis häufig numerische Maße verwendet, die für den Anwender aussagekräftiger und einfacher zu ermitteln sind.

Die Konvergenzgeschwindigkeit ist ein Maß, das die Effizienz eines Optimierungsverfahrens beschreibt. Dazu wird die Beziehung zwischen Berechnungsaufwand und Güte der Lösung betrachtet. Der Berechnungsaufwand kann sich auf verschiedene Größen beziehen. Diese Größen können Rechenzeit, Anzahl der Funktionsauswertungen oder auch die Anzahl der Ausführung bestimmter Prozeduren des Suchverfahrens sein. Für EA wird auch häufig die Anzahl der Generationen als Maß für den Berechnungsaufwand verwendet. Die Wahl dieser Größe hängt letztlich von der zu treffenden Aussage und der Handhabbarkeit ab.

Obwohl die Rechenzeit das objektivste Maß darstellt, wird sie aufgrund des Aufwandes, der für ihre Ermittlung und Aufbereitung betrieben werden muß, nur sehr selten verwendet. Um verschiedene Suchverfahren zu vergleichen, müssen alle Simulationen einer Meßreihe auf identischen Rechnern unter gleichen Bedingungen ausgeführt werden. Das abstraktere Maß *Anzahl der Funktionsauswertungen* (FE) gibt den tatsächlichen Rechenaufwand zwar weniger exakt wieder, ist gegenüber der Rechenzeit aber wesentlich einfacher zu handhaben. Dieses Maß wird verwendet, falls der für die Funktionsauswertungen erforderliche Rechenaufwand die Suche dominiert. Ein kostspieliger Operator innerhalb des Suchverfahrens, der keine Funktionsauswertungen ausführt, wird von diesem Maß nicht erfaßt. Das Maß *Anzahl der Generationen* (GEN) berücksichtigt, daß die Evaluierungen der Individuen einer Population parallel durchgeführt werden können. Falls die Größe einer

BF	$\bar{x}$	s	$s_{\bar{x}}$	$x_{\min}$	$x_{\max}$	SUCC
10^{+3}	34010	1899	396	31300	38270	100.0
10^{+2}	100200	2937	612	93890	105700	100.0
10^{+1}	164600	3219	671	158500	169400	100.0
10^{+0}	228300	4266	890	219100	237400	100.0
10^{-1}	291500	4711	982	281900	302000	100.0
10^{-2}	355400	5355	1117	346300	365600	100.0
10^{-3}	421000	6542	1364	411600	434500	100.0
10^{-4}	486400	6140	1280	474800	497000	100.0
10^{-5}	553900	6277	1309	540800	564500	100.0
10^{-6}	620000	6987	1457	606500	637100	100.0
10^{-7}	688600	7207	1503	677800	702800	100.0
10^{-8}	758700	6453	1346	746800	771100	100.0
10^{-9}	833200	6901	1439	819500	845200	100.0
10^{-10}	911600	7089	1478	898400	923000	100.0
10^{-11}	997300	7246	1511	982100	1009000	100.0
10^{-12}	1100000	8137	1697	1083000	1114000	100.0
10^{-13}	1262000	9810	2045	1239000	1280000	100.0
10^{-14}	-	-	-	-	-	50.0
10^{-15}	-	-	-	-	-	0.0

Tabelle 6.1: Beispiel für die Darstellung der Konvergenzgeschwindigkeit eines Algorithmus (bga, Kugelmodell, n=1000). Nur 50 % der Läufe erreichen einen Funktionswert, der kleiner als 10^{-14} ist.

Population während der Suche unverändert bleibt, können die beiden Maße FE und GEN ineinander umgerechnet werden.

Da die meisten realen Anwendungen aufwendige Funktionsauswertungen aufweisen, wird im folgenden als Maß für den Berechnungsaufwand die Anzahl der Funktionsauswertungen zugrunde gelegt. Die Güte der Lösung wird am zur Zeit besten Funktionswert (BF) gemessen. Die Effizienz eines Suchverfahrens wird im folgenden also durch den funktionalen Zusammenhang, der zwischen FE und BF besteht, beschrieben.

Da die Wertebereiche der Testfunktionen bekannt sind, wird in den folgenden Messungen zu bestimmten Zielfunktionswerten die Anzahl der Funktionsauswertungen angegeben. Die Wahl des zu untersuchenden Bereichs der Zielfunktionswerte ist letztlich willkürlich und hängt von der benötigten Genauigkeit ab. Falls nicht explizit spezifiziert, werden in den folgenden Messungen als Fitnesswerte alle Zehnerpotenzen von 10^3 bis 10^{-15} verwendet (siehe Tabelle 6.1 auf Seite 142). Damit überdecken die so gewonnenen 19 Datenpunkte $(\mathsf{BF}_i, \mathsf{FE}_i)$ einen hinreichend großen Bereich von Zielfunktionswerten und stellen die Grundlage für die Darstellung der Simulationsergebnisse dar.

Die Darstellung der Konvergenzgeschwindigkeit erfolgt in der in Tabelle 6.1 dargestellten Form. Für ausgewählte Fitnesswerte (BF) werden Statistiken der zu deren Erreichen benötigten Funktionsauswertungen angegeben. Die Statistik besteht aus Mittelwert ($\bar{x}$), Standardabweichung (s), Standardfehler des Mittelwertes ($s_{\bar{x}}$), den Extrema ($x_{\min}$, $x_{\max}$) sowie der prozentualen Erfolgsrate (SUCC).

Die in Tabelle 6.1 angegebenen Werte lassen sich graphisch als Funktion der jeweiligen Maße über die Fitnesswerte darstellen. In Abbildung 6.7 sind zunächst für zwei verschiedene Algorithmen die Mittelwerte und Standardabweichungen der benötigten Funktionsauswertungen eingezeichnet. Der erste Algorithmus ist ein einfaches Mutations-Selektions-Verfahren mit BGA-Mutationsverteilung (bga-1+1). Der zweite Algorithmus ist der Standard-BGA mit Fuzzy-Rekombination (bga). Die zweite Zeile enthält die mittlere Anzahl von Funktionsauswertungen, die das Verfahren benötigt, um die Fitness um eine Zehnerpotenz zu reduzieren (DIFF). In der dritten Zeile ist der Prozentsatz der Läufe eingetragen, die den jeweiligen Fitnesswert erreicht haben (SUCC). Das Mutations-Selektions-Verfahren in der linken Spalte erreicht in allen Läufen alle Fitnesswerte, während das Rekombinationsverfahren in der rechten Spalte den vorletzten Fitnesswert nur in 50 Prozent und den letzten in keinem Lauf erreicht. Die Mittelwerte werden nur für Fitnesswerte aufgenommen, die mindestens eine 80-prozentige Erfolgsrate haben.

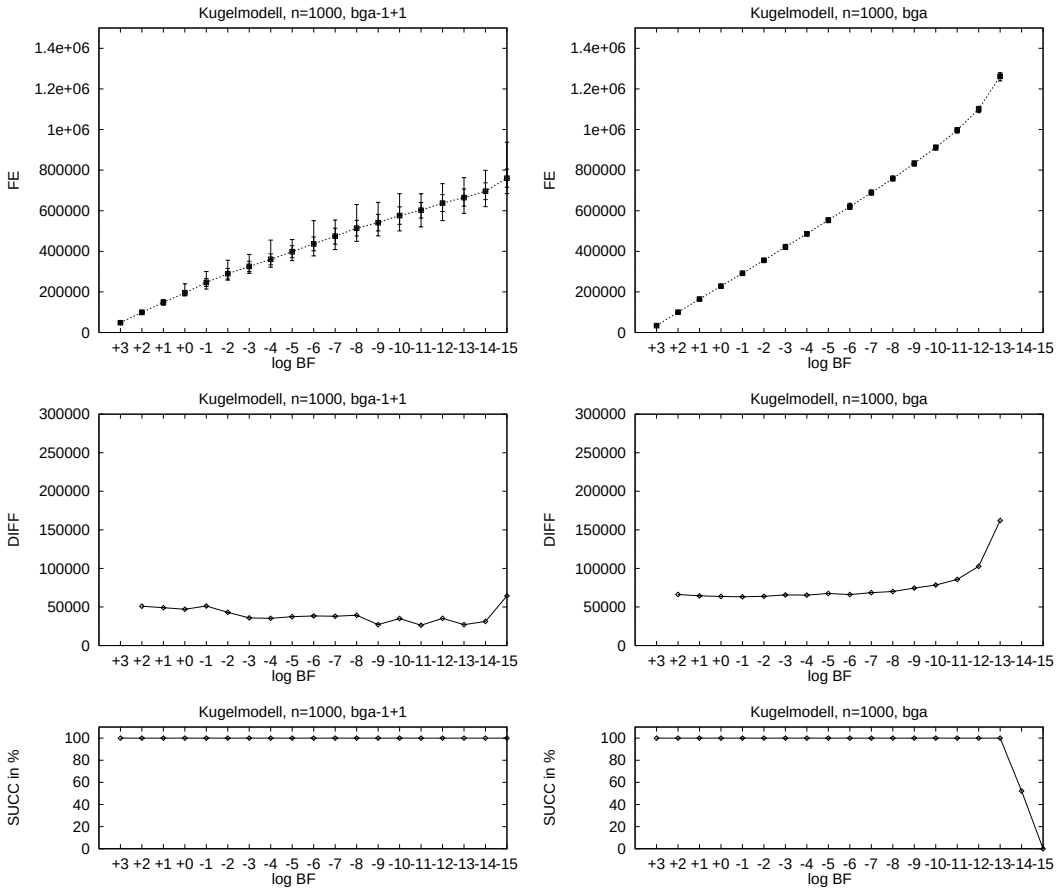

Abbildung 6.7: Beispiel für die graphische Darstellung des Konvergenzverhaltens von Suchverfahren am Beispiel des 1000-dimensionalen Kugelmodells. In der linken Spalte sind die Werte eines Mutations-Selektions-Verfahrens mit BGA-Mutationsverteilung (bga-1+1) und in der rechten Spalte die Werte eines BGA mit Rekombination dargestellt (bga). Die erste Zeile enthält Mittelwerte und Standardabweichungen der Funktionsauswertungen, die bis zur Unterschreitung der in der Abszisse eingetragenen Fitnesswerte benötigt werden (FE). Die zweite Zeile enthält die mittlere Anzahl von Funktionsauswertungen, die für die Reduzierung der Fitness um eine Zehnerpotenz benötigt werden (DIFF). In der letzten Zeile ist der Prozentsatz der Läufe eingetragen, die den jeweiligen Fitnesswert erreicht haben (SUCC).

6.5.2 Robustheit

Die Robustheit eines Optimierungsverfahrens setzt sich aus verschiedenen Aspekten zusammen. Dazu zählen neben der Konvergenzsicherheit, die die Fähigkeit zum Auffinden des globalen Optimums betrifft, auch die Wiederholbarkeit von Experimenten und die Anwendbarkeit des Verfahrens.

Grundsätzlich gilt, daß Robustheit und Effizienz gegenläufige Eigenschaften sind. Die Gewichtung beider Eigenschaften hängt letztlich von den Anforderungen der Anwendung ab. Die Bearbeitung eines speziellen eng umrissenen Optimierungsproblems kann in der Regel am effizientesten durch ein darauf zugeschnittenes Suchverfahren gelöst werden.

Wiederholbarkeit

Verschiedene Läufe stochastischer Optimierungsverfahren weisen in der Regel nicht dasselbe Verhalten auf. Die Schwankungsbreite des Konvergenzverhalten gibt Aufschluß über die Robustheit eines Algorithmus. Ein Verfahren, das in seinem Konvergenzverhalten geringen Schwankungen unterliegt, ist einem Verfahren mit hohen Schwankungen vorzuziehen. Als Maß für die Schwankung eignet sich die Standardabweichung der Zeit, die ein Suchverfahren benötigt wird, um ein bestimmtes Kriterium zu erfüllen.

Anwendbarkeit

Ein wichtiger Aspekt der Robustheit betrifft die Anwendbarkeit des Algorithmus auf verschiedene Problemstellungen. So gilt ein Algorithmus, der für unterschiedliche Probleme ein ähnliches Konvergenzverhalten aufweist, als robust, während ein auf eine bestimmte Problemklasse zugeschnittener Algorithmus in diesem Zusammenhang als weniger robust anzusehen ist.

In diesem Zusammenhang sei das von Wolpert und Macready entwickelte *No Free Lunch Theorem (NFL-Theorem)* erwähnt ([WM95]), das die Grenzen des obigen Robustheitsaspekts aufzeigt. Das NFL-Theorem sagt aus, daß alle Algorithmen gemittelt über alle Zielfunktionen dieselbe Güte aufweisen. Anders ausgedrückt bedeutet dies, daß, falls kein Wissen über die Zielfunktion vorliegt, keine Aussage über die Eignung eines Algorithmus gemacht werden kann.

Konvergenzsicherheit

Der wesentliche Robustheitsaspekt eines Suchverfahrens betrifft die Fähigkeit eines Algorithmus, das Optimum eines Problems bis auf eine vorgegebene Genauigkeit zu approximieren. Diese Fähigkeit wird als Konvergenzsicherheit eines Suchverfahrens bezeichnet.

In diesem Zusammenhang spielt der Begriff der *globalen Konvergenz* eine wichtige Rolle. Unter globaler Konvergenz ist die Fähigkeit eines Algorithmus zu verstehen, das globale Optimum eines multimodalen Problems zu finden. In diesem Zusammenhang werden Suchverfahren häufig als *lokal* oder *global* charakterisiert. Obwohl diese beiden Begriffe im Sprachgebrauch der Optimierung sehr häufig verwendet werden, ist eine präzise Definition für ableitungsfreie Verfahren recht schwierig. Auf der einen Seite kann ein lokales Suchverfahren nicht dadurch definiert werden, daß es immer in das nächstgelegene lokale Optimum läuft, da in Abhängigkeit der initialen Schrittweite anfangs ein Punkt eines anderen Attraktorgebiets generiert werden kann. Auf der anderen Seite ist der Begriff eines

globalen Optimierers zu eng gefaßt, falls darunter ausschließlich ein Verfahren verstanden wird, das für jede Funktion immer das globale Optimum findet [1].

Aus diesem Grund sind die Begriffe *lokales* und *globales Suchverhalten* besser geeignet, um ein Suchverfahren zu charakterisieren. Empirische Leistungsmaße, die das globale Suchverhalten von Suchverfahren beschreiben, werden im folgenden eingeführt. Die Darstellung der Konvergenzsicherheit im Rahmen der multimodalen Optimierung ist recht schwierig und läßt sich kaum in einer Zahl ausdrücken. Mögliche Werte sind der Prozentsatz der erfolgreichen Läufe oder der Durchschnitt der erreichten Fitness.

Bäck verwendet in diesem Zusammenhang eine aussagekräftigere Darstellung, die die relativen Häufigkeiten der bis zum Eintreten der Terminierungsbedingung erreichten Fitnesswerte wiedergibt ([Bäc96]). Da die Darstellung relativer Häufigkeiten für menschliche Betrachter nicht einfach zu interpretieren ist, basiert das in dieser Arbeit verwendete Robustheitsmaß auf der akkumulierten Verteilung der Fitnesswerte.

Aus der Häufigkeitsverteilung der erreichten Fitnesswerte lassen sich signifikante Größen gewinnen, die die Robustheit eines Verfahrens widerspiegeln. Im folgenden wird mit f_i^* die beste erreichte Fitness des i-ten Experiments bezeichnet. Die Anzahl der Experimente wird durch N gekennzeichnet. Aus $f_1^*, \ldots, f_N^*$ lassen sich zunächst Mittelwert $\bar{f}^*$, Standardabweichung f_{sd}^* sowie Minimum $f_{\min}^*$ und Maximum $f_{\max}^*$ berechnen. Des weiteren eignet sich das Integral über die akkumulierte Verteilung in einem festgelegten Intervall als weiteres Robustheitsmaß.

$$A(c) = \int_{f_{\min}^*}^{c} H(x)\, dx \tag{6.23}$$

wobei H die akkumulierten Häufigkeit der Fitnesswerte $f_1^*, \ldots, f_N^*$ ist. Die Wahl der oberen Integrationsgrenze c beeinflußt die Signifikanz der Größe A. Während ein zu klein gewähltes c den entscheidenden Bereich nicht erfaßt, führt ein zu großes c zu einem Verlust der Unterschiede. Der Parameter c hängt letztlich von den Werten der verschiedenen Suchverfahren ab. Ein möglicher Wert für die rechte Integrationsgrenze ist der durchschnittliche Wert von $f_{\max}^*$ aller zu vergleichenden Suchverfahren. Ein Suchverfahren ist umso robuster, je größer der Wert A ist.

Ein Beispiel für eine solche Darstellung zeigt Abbildung 6.8. Hier ist die Robustheit zweier unterschiedlicher Suchverfahren dargestellt. Aus 100 Testläufen wurden die erreichten Fitnesswerte verwendet, um die relativen Häufigkeiten zu berechnen. Das in Abbildung (a) dargestellte Suchverfahren verhält sich weniger robust, da die meisten Läufe weit entfernt vom Optimum enden. Numerische Werte, die Aufschluß über die Robustheit eines Optimierungsverfahrens geben, sind für die beiden Experimente in Tabelle 6.2 dargestellt.

6.5.3 Skalierung

Ein weiterer wichtiger Aspekt eines Optimierungsverfahrens betrifft die Abhängigkeit von der Problemgröße. Zur Beschreibung dieses Verhaltens wird das Komplexitätsmaß *Skalierung* eingeführt. Die Skalierung eines Algorithmus bezüglich einer Zielfunktion wird durch den funktionalen Zusammenhang zwischen der Problemgröße und dem Aufwand, der für eine bestimmte Zielannäherung notwendig ist, beschrieben. Eine Aussage über die Ska-

[1] Rechenberg vergleicht die Suche nach einem globalen Optimierungsverfahren in diesem Sinne mit der Suche nach dem Perpetuum Mobile ([Rec94], S.152).

	Suchverfahren	
	a	b
$\bar{f}^*$	150457	7205
f^*_{sd}	67601	8866
$f^*_{\min}$	44591	12
$f^*_{\max}$	353475	36450
$A(\bar{f}^*_{\max})$	55150	184750

Tabelle 6.2: Numerische Robustheitsmaße für die in Abbildung 6.8 dargestellten Optimierungsverfahren. Die Robustheitsmaße sind: Durchschnitt, Standardabweichung, Minimum und Maximum der erzielten Fitnesswerte sowie die Fläche unterhalb der akkumulierten Häufigkeit im Intervall $[0, \bar{f}^*_{\max}]$. Aus der Tabelle ergibt sich für $\bar{f}^*_{\max}] = 194963$.

lierung bezieht sich also immer auf eine bestimmte Zielfunktion und eine vorgegebene Zielannäherung.

Zur Beschreibung der Skalierung eines Algorithmus werden Modelle verwendet, die aus den empirisch ermittelten Daten gewonnen werden. Die Daten enthalten die Anzahl der Funktionsauswertungen, die für verschiedene Dimensionen (n) erforderlich sind, um die vorgegebene Zielannäherung zu erreichen. Die Modelle sind einfache Funktionen, die durch ihre Koeffizienten an die beobachteten Daten angepaßt werden. Die Basisfunktionen sind so gewählt, daß alle typischen Skalierungsmuster dargestellt werden können.

$$\begin{aligned}
\text{M1} &: f_1(n) = a + b \cdot n^{0.75} \\
\text{M2} &: f_2(n) = a + b \cdot n \\
\text{M3} &: f_3(n) = a + b \cdot n \ln n \\
\text{M4} &: f_4(n) = a + b \cdot n^2 \\
\text{M5} &: f_5(n) = a + b \cdot n^{2.5} \\
\text{M6} &: f_6(n) = a + b \cdot n^3
\end{aligned} \tag{6.24}$$

Die Bestimmung der Koeffizienten basiert auf der Methode der gewichteten kleinsten Quadrate ([PTVF92]). Der zu minimierende Fehler ist die Summe der durch ihre Standardabweichungen gewichteten Fehlerquadrate der Datenpunkte:

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i \Leftrightarrow f(n_i; a, b)}{s_i} \right)^2 \tag{6.25}$$

wobei y_i die Anzahl der Funktionsauswertungen und N die Anzahl der Datenpunkte (n_i, y_i) beschreibt [2].

Diese Methode, die für $s_i = 1.0 \quad (i = 1, \ldots, N)$ der Methode der kleinsten Quadrate entspricht, erlaubt unter der Annahme einer normalverteilten Grundgesamtheit, die Güte des berechneten Modells zu quantifizieren. Das Gütemaß wird aus dem χ^2-Wert und

[2] Zur Berechnung der Koeffizienten a und b sowie des dazugehörigen χ^2 wurde die Prozedur *lfit* aus den *Numerical Recipes in C* verwendet ([PTVF92])

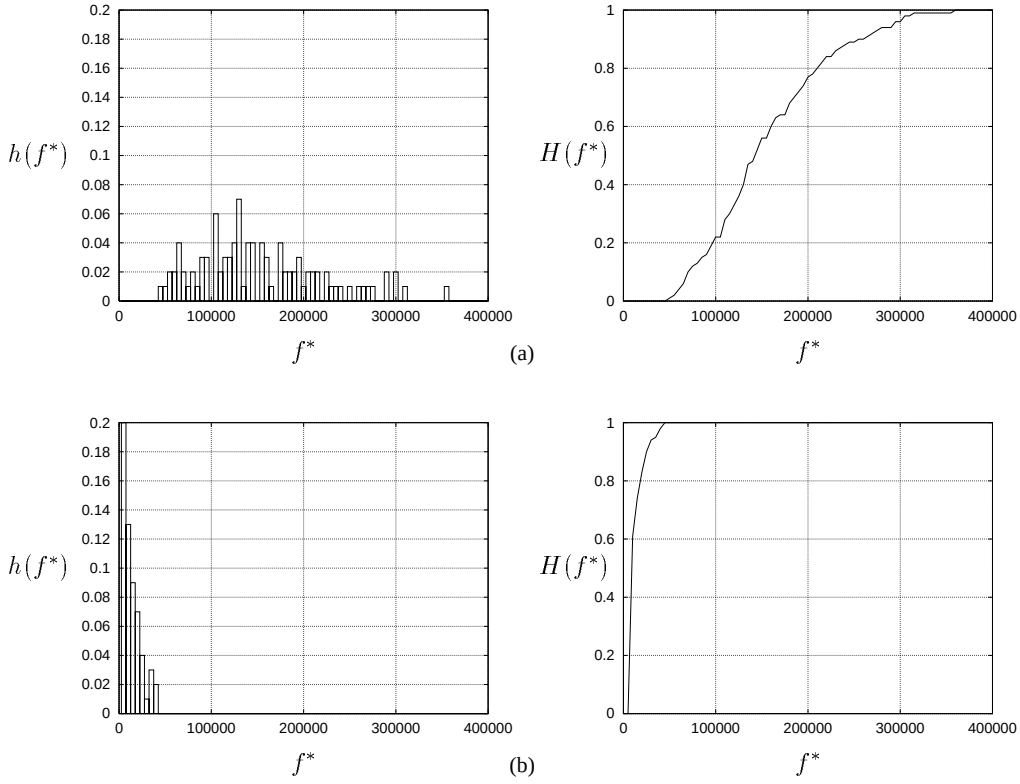

Abbildung 6.8: Darstellung der Robustheit zweier Optimierungsverfahren, angewandt auf eine multimodale Funktion (Funktion von Fletcher und Powell, n=30). Die Abszisse zeigt den Bereich der besten Funktionswerte der Zielfunktion, die in 100 unabhängigen Läufen nach Erreichen der Terminierungsbedingung erzielt wurden. Die einzelnen Segmente haben eine Länge von 5000.0. Die Ordinate gibt die relative Häufigkeit der erreichten Funktionswerte in dem jeweiligen Segment an.

der Anzahl der Freiheitsgrade ν mit Hilfe der unvollständigen Gamma-Funktion $Q(a, x)$ berechnet:

$$Q\left(\frac{\nu}{2}, \frac{\chi^2}{2}\right) \tag{6.26}$$

Üblicherweise wird ein Modell akzeptiert, falls dessen Q-Wert größer als 10^{-3} ist ([PTVF92]). Die Güte einer Modellanpassung kann auch aus dem χ^2-Wert geschätzt werden. Als Daumenregel für ein akzeptables Modell sollte $\chi^2 \approx \nu$ sein.

In Tabelle 6.3 sind die χ^2- und Q-Werte sowie die Koeffizienten der sechs Modelle für die Simulation eines Standard-BGA angegeben. Das Modell M2 weist den niedrigsten χ^2-Wert (und höchsten Q-Wert) auf und liefert damit die beste Beschreibung der vorliegenden Daten. Somit skaliert der EA für diese Funktion linear. Die zugrundeliegenden Daten sind in Abbildung 6.9 dargestellt.

6.6 Simulationsergebnisse

Das Konvergenz- und Skalierungsverhalten der in Abschnitt 6.3 spezifizierten Suchverfahren wird für die verschiedenen Testfunktionen bestimmt.

Modell	a	b	χ^2	Q
M1	-6824.4	1765.3	10.8	0.005
M2	-1862.2	533.24	0.6	0.740
M3	1254.5	112.53	1.8	0.391
M4	4688.2	5.070	55.4	9.1E-13
M5	6250.0	0.0483	136.7	2.2E-30
M6	7229.0	$4.48E-16$	233.9	0.0

Tabelle 6.3: Beispiel für die Darstellung der Skalierung des Standard-BGA (bga) für das Kugelmodell. Die Ergebnisse basieren auf Messungen für die Dimensionen $n = 10, 30, 50$ und 100 mit jeweils 30 Läufen.

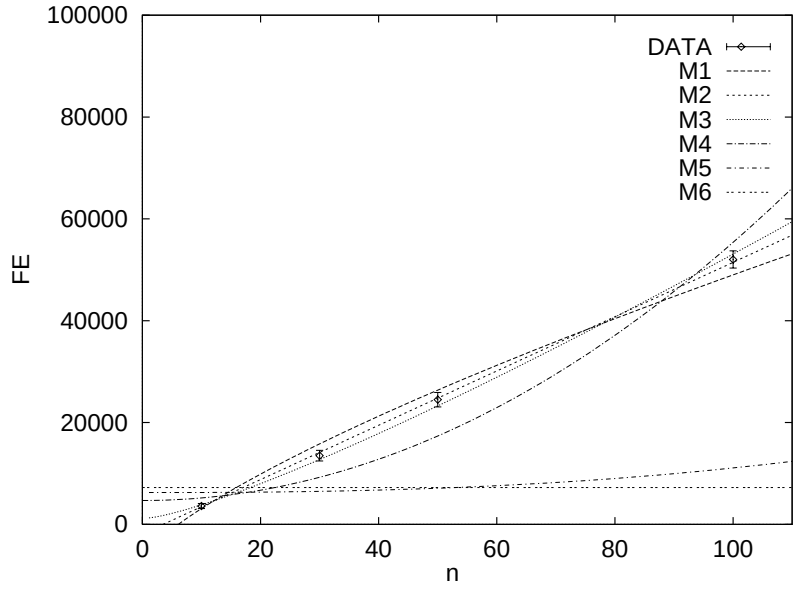

Abbildung 6.9: Modellierung der Skalierung eines EA für das Kugelmodell. Die Koeffizienten sind in Tabelle 6.3 angegeben.

6.6.1 Konvergenzverhalten

Zunächst wird das Konvergenzverhalten der Suchverfahren für ausgewählte Instanzen der in Abschnitt 6.2 beschriebenen Testfunktionen beschrieben. Die Messungen bezüglich der beiden Kugelmodelle, Schwefels und Ackleys Funktion beziehen sich auf deren 100-dimensionale Instanz. Von Rosenbrocks Funktion und von der Funktion von Flechter und Powell wird die 30-dimensionale Instanz betrachtet.

Die Resultate werden, wie in Abschnitt 6.5.1 beschrieben, graphisch dargestellt. Auf der Abszisse sind die Zielfunktionswerte (BF) eingetragen, auf die sich die Simulationsergebnisse beziehen. Da der optimale Zielfunktionswert aller Testfunktionen Null ist, entsprechen diese Werte der Abweichung des Funktionswerts vom Optimum. Der obere Graph jeder Abbildung zeigt die mittlere Anzahl der Funktionsauswertungen, die für das Erreichen des jeweiligen Zielfunktionswerts benötigt werden (FE). Der mittlere Graph zeigt die Anzahl der Funktionsauswertungen, die für das Erreichen des jeweils nächsten in der Abszisse eingetragen Zielfunktionswertes notwendig sind (DIFF). Der untere Graph gibt

den prozentualen Anteil der Läufe an, die den jeweiligen Zielfunktionswert erreicht haben (SUCC). Die FE- und DIFF-Werte zu einem Zielfunktionswert (BF) werden nur dann in die Graphik aufgenommen, falls mindestens 80 % der Läufe diesen Zielfunktionswert erreicht haben.

Bei der Betrachtung der Simulationsergebnisse ist zu beachten, daß das verwendete Maß für den Berechnungsaufwand, nämlich die Anzahl der Funktionsauswertungen, das Beschleunigungspotential, das populationsbasierte Suchverfahren durch die parallele Bearbeitung der Individuen erzielen können, nicht berücksichtigt. Wie in Kapitel 8 gezeigt wird, kann bei einer kostspieligen Zielfunktion die Ausführungszeit durch den Einsatz eines Parallelrechners um den Faktor λ, der die Populationsgröße bezeichnet, reduziert werden.

Kugelmodell (F1)

Das Konvergenzverhalten der lokalen Suchverfahren für das 100-dimensionale Kugelmodell ist in Abbildung 6.10 auf Seite 152 dargestellt. Da sich die Ergebnisse der drei deterministischen lokalen Suchverfahren (qn, pow, dhs) sehr stark von denen der restlichen Suchverfahren unterscheiden, ist deren Konvergenzverhalten in Abbildung 6.14 auf Seite 156 noch einmal gesondert mit anderen Skalierungen dargestellt.

Das Powell-Verfahren (pow) und das Quasi-Newton-Verfahren (qn) weisen erwartungsgemäß bis zum Fitnesswert 10^{-10} die mit Abstand höchste Konvergenzgeschwindigkeit auf. Ihre FE- und DIFF-Werte sind so gering, daß sie mit der in Abbildung 6.10 verwendeten Skalierung nicht mehr darstellbar sind. Wie in Abbildung 6.14 zu sehen ist, erreichen das Quasi-Newton-Verfahren und das Powell-Verfahren den Zielfunktionswert 10^{-10} mit ungefähr 500 beziehungsweise 1000 Funktionsauswertungen. Da der verwendete Simulator innerhalb der eindimensionalen Optimierer keine Zielfunktionswerte erfaßt, kann aufgrund der Graphik keine exaktere Aussage über das Konvergenzverhalten der beiden Suchverfahren getroffen werden. Das Downhill-Simplex-Verfahren erreicht den vorgegebenen Zielfunktionswert von 10^{-15} in allen Läufen nach zwei Iterationen des Basisverfahrens. In der ersten Iteration treten deutliche Schwierigkeiten bei der Anpassung des hochdimensionalen Simplex an die Fitnesslandschaft auf. Nach 1000000 Funktionsauswertungen bricht die erste Iterationsstufe mit einem Zielfunktionswert, der über 0.1 liegt, aufgrund der oberen Schranke für die Funktionsauswertungen ab. Die zweite Iterationsstufe (nach Neustart) erreicht mit dem neu initialisierten Simplex ein erheblich besseres Verhalten. Das Downhill-Simplex-Verfahren konvergiert nun linear mit einem DIFF-Wert von exakt 2000. Der Grund für dieses Verhalten liegt in der Parametrisierung des Basisverfahrens, dessen Eignung von dem Abstand zum Optimum abhängt. Ab dem Zielfunktionswert 10^{-8} schwanken die DIFF-Werte zwischen 4000 und 50000. Das Downhill-Simplex-Verfahren ist das einzige lokale Suchverfahren, das den vorgegebenen Zielfunktionswert von 10^{-15} in allen Läufen erreicht. Dieses Ergebnis verdeutlicht, daß auch die Parameter, die nicht zum eigentlichen Basisverfahren zählen, erheblichen Einfluß auf die Suche haben. So regelt das Abbruchkriterium des Basisverfahrens den Übergang in die nächste Iterationsstufe. In diesem Fall hätte ein früherer Abbruch der ersten Iteration, die über lange Zeit eine geringe Konvergenzverhalten aufweist, zu einem deutlich besseren Gesamtergebnis geführt.

Das Verfahren von Solis und Wets (sw) konvergiert linear mit einem mittleren DIFF-Wert von 1000. Es zeichnet sich durch eine sehr geringe Schwankungsbreite der Simulationsergebnisse aus. Mit einem Zielfunktionswert von 10^{-13} erzielt das Verfahren von Solis und Wets nach dem Downhill-Simplex-Verfahren die beste Zielannäherung.

Das Dynamic-Hillclimbing-Verfahren (dhc) erweist sich beim Kugelmodell als sehr instabiles Verfahren. Der Zielfunktionswert 10^{-7} wird bereits von 20 % der Läufe nicht erreicht. Zudem sind die Simulationsläufe recht hohen Schwankungen unterworfen.

In Abbildung 6.11 auf Seite 153 sind die Ergebnisse der Evolutionsstrategien und des Genetischen Algorithmus dargestellt. Im Gegensatz zu den lokalen Suchverfahren erreichen alle Evolutionären Algorithmen den Fitnesswert 10^{-15}. Die Evolutionsstrategie es1 mit einem Strategieparameter konvergiert über den gesamten Fitnessbereich linear mit einem durchschnittlichen DIFF-Wert von 5000. Die Evolutionsstrategie es2 mit n Strategieparametern erreicht mit DIFF-Werten, die zwischen 30000 und 40000 liegen, eine deutlich geringere Konvergenzgeschwindigkeit[3]. Der Grund dafür liegt in dem Aufwand, der mit der Anpassung der n Strategieparameter verbunden ist. Da sich bei dieser Zielfunktion die Beiträge aller Objektvariablen zum Fitnesswert nicht unterscheiden, reicht die Anpassung eines Strategieparameters allerdings aus.

Das Wettbewerbsmodell ces, das beide Evolutionsstrategien kombiniert, erzielt mit einem durchschnittlichen DIFF-Wert von 4500 eine etwas bessere Konvergenzgeschwindigkeit als die Evolutionsstrategie es1. Die geeignetere Evolutionsstrategie ist offenbar in der Lage, sich im Wettbewerbsmodell durchzusetzen. Die leichte Verbesserung gegenüber der Evolutionsstrategie es1 ist auf die im Wettbewerbsmodell verwendete geringfügig härtere Selektion zurückzuführen.

Der Genetische Algorithmus (ga) zeigt schon bei dieser einfachen Funktion bezüglich Zielannäherung und Konvergenzgeschwindigkeit ein sehr schlechtes Konvergenzverhalten. Aufgrund der zu geringen Konvergenzgeschwindigkeit terminiert er in allen Läufen mit Zielfunktionswerten, die über 10^{-3} liegen. Das Konvergenzverhalten des Genetischen Algorithmus wurde eingehend von Rudolph untersucht ([Rud96]). Er wies nach, daß der Genetische Algorithmus nie konvergiert, sondern zuletzt um einen Fixpunkt fluktuiert. Dieser Fixpunkt muß nicht mit dem Optimum übereinstimmen.

In Abbildung 6.12 auf Seite 154 ist das Konvergenzverhalten der BGAs dargestellt. Als effizientester BGA erweist sich bei dieser einfachen Funktion das aus einem Individuum bestehende Mutations-Selektions-Verfahren bga-1+1. Es konvergiert über den gesamten Fitnessbereich linear mit einem durchschnittlichen DIFF-Wert von 3000. Zum Vergleich ist in Abbildung 6.15 das Konvergenzverhalten eines anderen Mutations-Selektions-Verfahren, der sogenannten 1+1-Evolutionsstrategie, dargestellt. Diese einfachste Variante der Evolutionsstrategie, die aus einem Individuum besteht und analog zu es1 nur einen Strategieparameter verwendet, erzielt bei einer höheren Schwankungsbreite ein noch besseres Ergebnis als das bga-1+1-Verfahren. Die DIFF-Werte liegen hier um die 2000.

Der Standard-BGA (bga) konvergiert bis zum Zielfunktionswert 10^{-10} ebenfalls linear. Ab da verringert sich die Konvergenzgeschwindigkeit mit zunehmender Zielannäherung aufgrund seiner zu großen minimalen Schrittweite.

Wie erwartet, verhält sich der BGA mit Hauptachsentransformation (bga-pca) wie der Standard-BGA. Die Veränderung des Koordinatensystems hat aufgrund der Kugelsymmetrie dieser Funktion keinen Einfluß auf die Suche. Aufgrund der Rotation des Koordinatensystems können die Objektparameter hier kleinere Werte als die minimale Schrittweite

[3]Messungen von Schwefel ergaben für es1 und es2 DIFF-Werte von 2850 beziehungsweise 27450. Die Abweichungen sind vermutlich auf die unterschiedliche Implementierung und die Parametrisierung der Evolutionsstrategie zurückzuführen.

der BGA-Mutationsverteilung annehmen. Die Verringerung der Konvergenzgeschwindigkeit, die in der Endphase mit zunehmender Zielannäherung auftritt, ist daher weit weniger dramatisch als beim Standard-BGA.

Die drei cBGA, deren Konvergenzverhalten in Abbildung 6.13 dargestellt sind, zeigen ein annähernd lineares Konvergenzverhalten. Die beiden cBGA mit unterschiedlichen Schrittweitenparametern (cbga1 und cbga3) sind in der Lage, die für die BGA-Mutation typische Stagnationsphase durch Auswahl beziehungsweise Anpassung der Schrittweitenparameter zu vermeiden. Typisch für den cbga1 sind dessen wellenförmig verlaufende DIFF-Werte, die aus den unterschiedlichen Schrittweitenparametern der konkurrierenden Populationen resultieren. Mit einem durchschnittlichen DIFF-Wert von 5000 erzielen sie eine ähnliche Konvergenzgeschwindigkeit wie die Evolutionsstrategie es1. Der cbga3 benötigt aufgrund der aufwendigen Einschwingphase ungefähr 40000 Funktionsauswertungen mehr als der cbga1, um den Zielfunktionswert 100.0 zu erreichen.

Der cbga2, der Breiten- und Richtungssuche kombiniert, weist aufgrund seiner zeitweisen hohen Populationsgröße eine geringere Effizienz auf. Sein durchschnittlicher DIFF-Wert liegt bei 14000. Die geringen Schwankungen seiner Meßergebnisse weisen ihn als sehr robustes Verfahren aus.

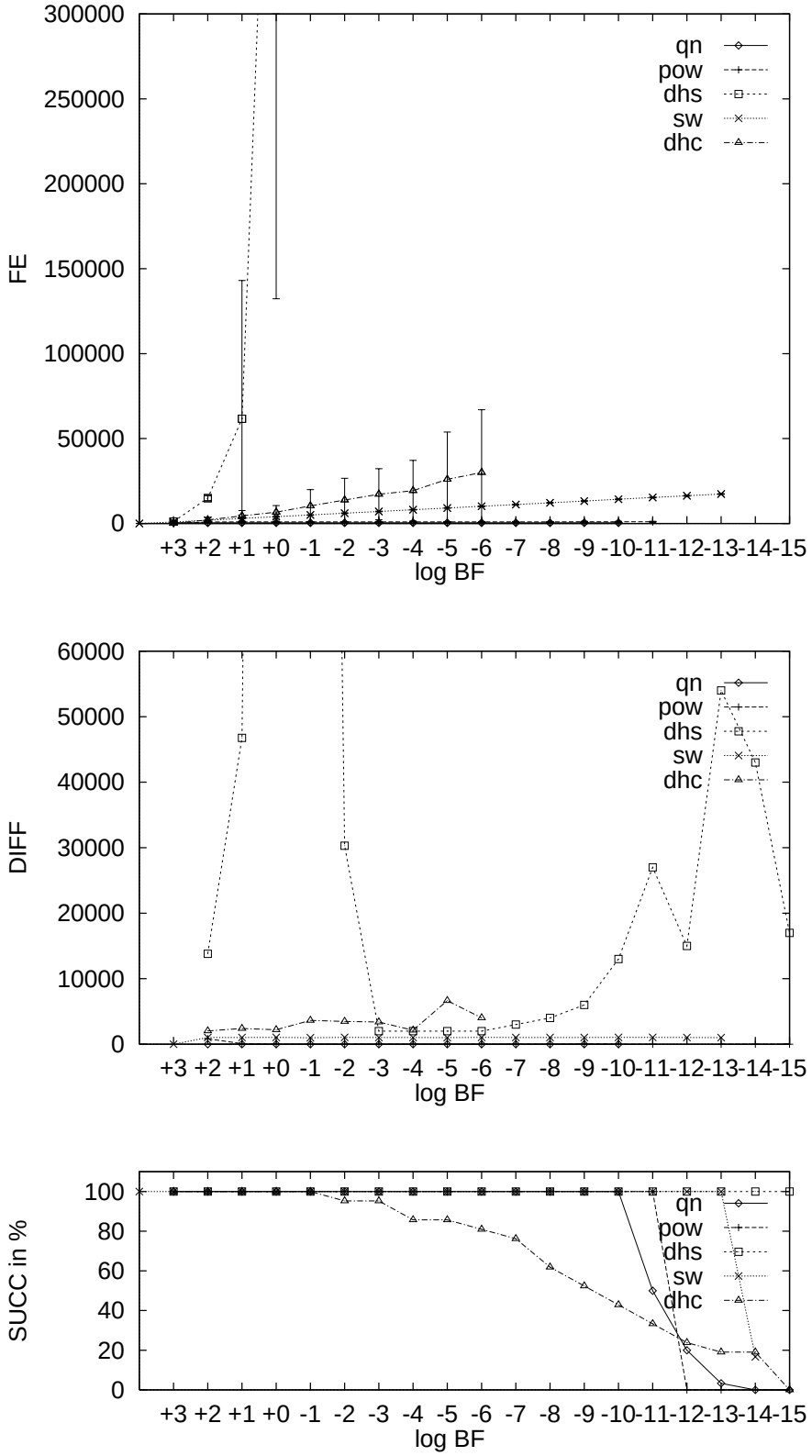

Abbildung 6.10: **Kugelmodell** (F1, n=100): Konvergenzverhalten von lokalen Suchverfahren.

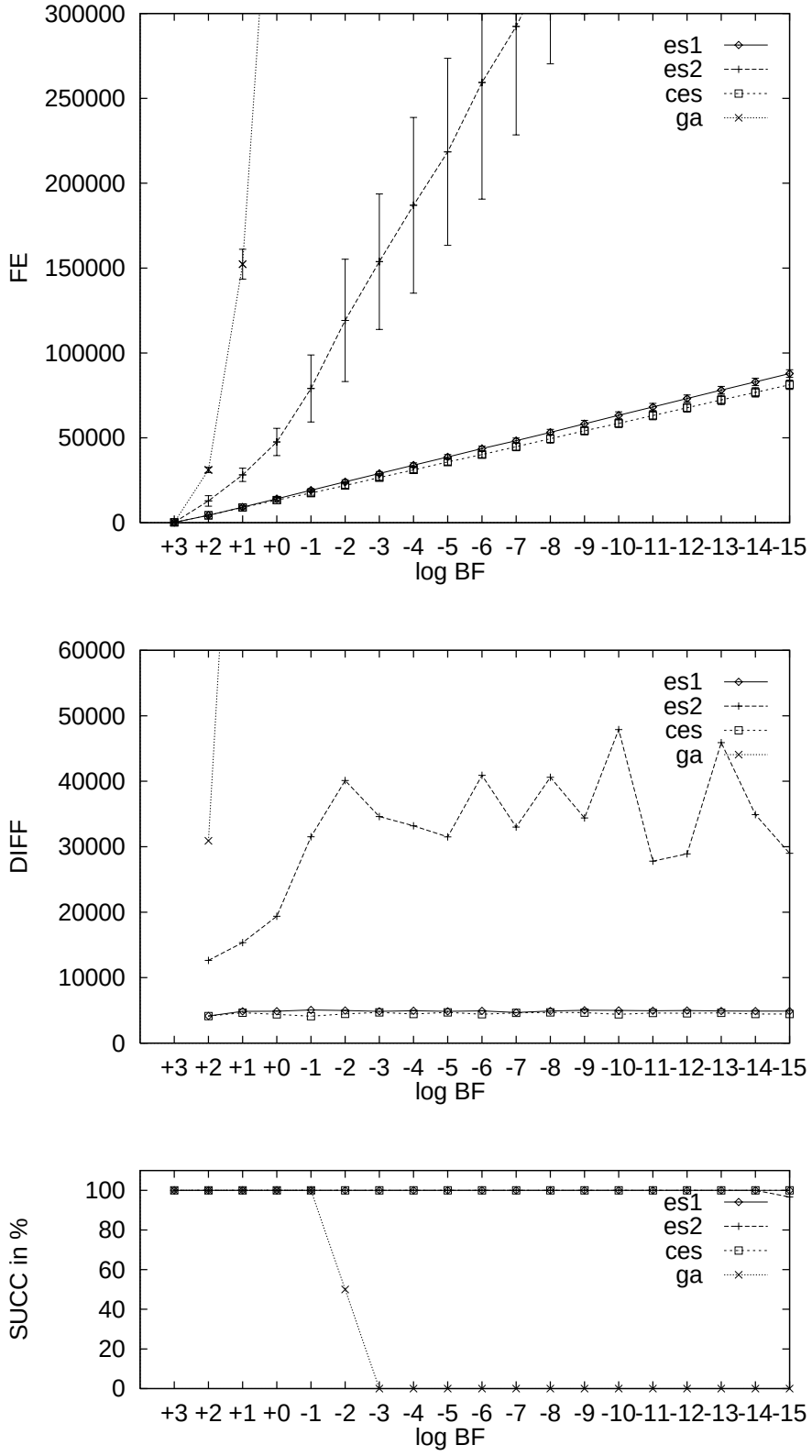

Abbildung 6.11: **Kugelmodell** (F1, n=100): Konvergenzverhalten von Evolutionsstrategien und Genetischem Algorithmus.

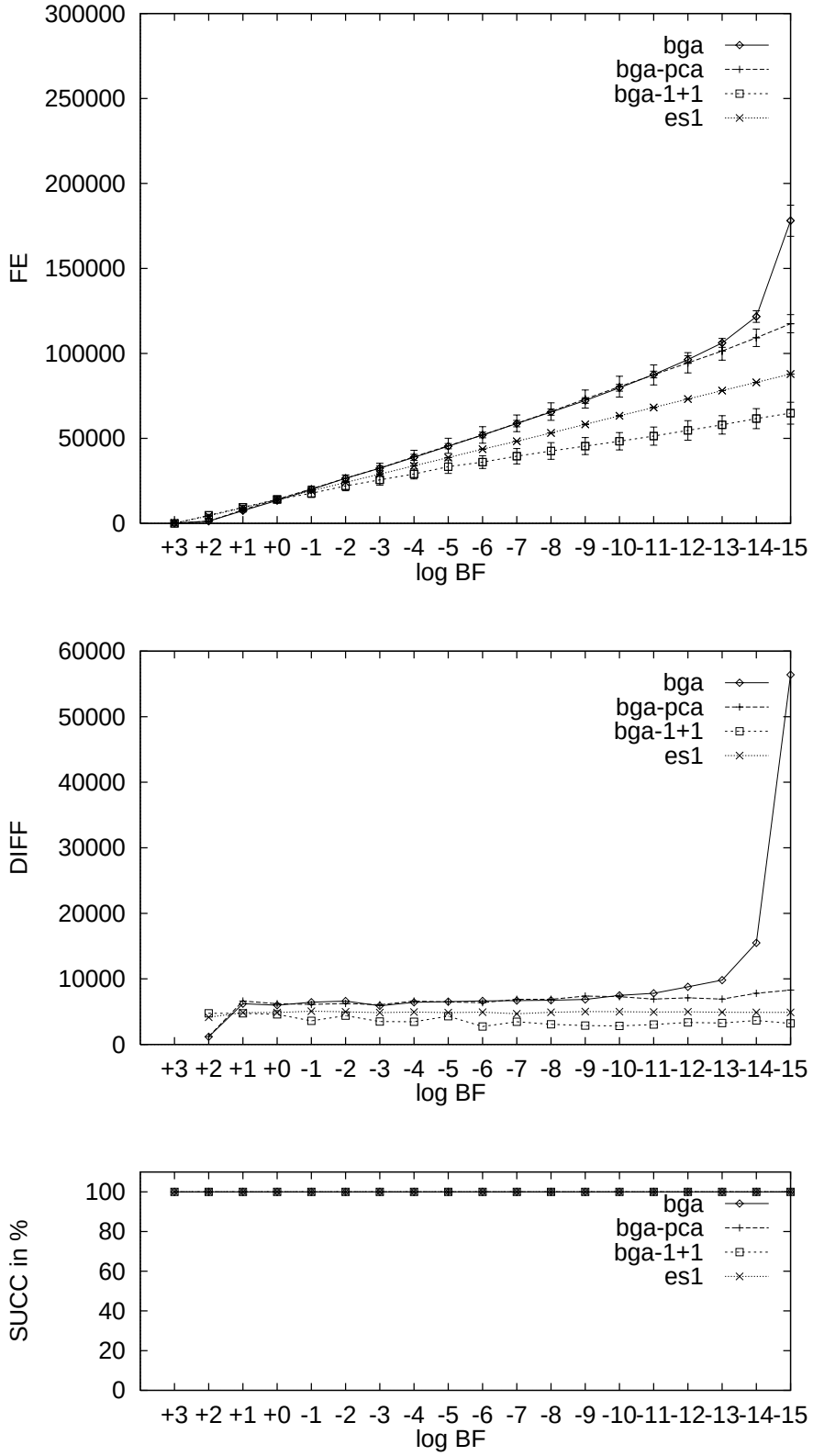

Abbildung 6.12: **Kugelmodell** (F1, n=100): Konvergenzverhalten von BGA-Varianten und Evolutionsstrategie es1.

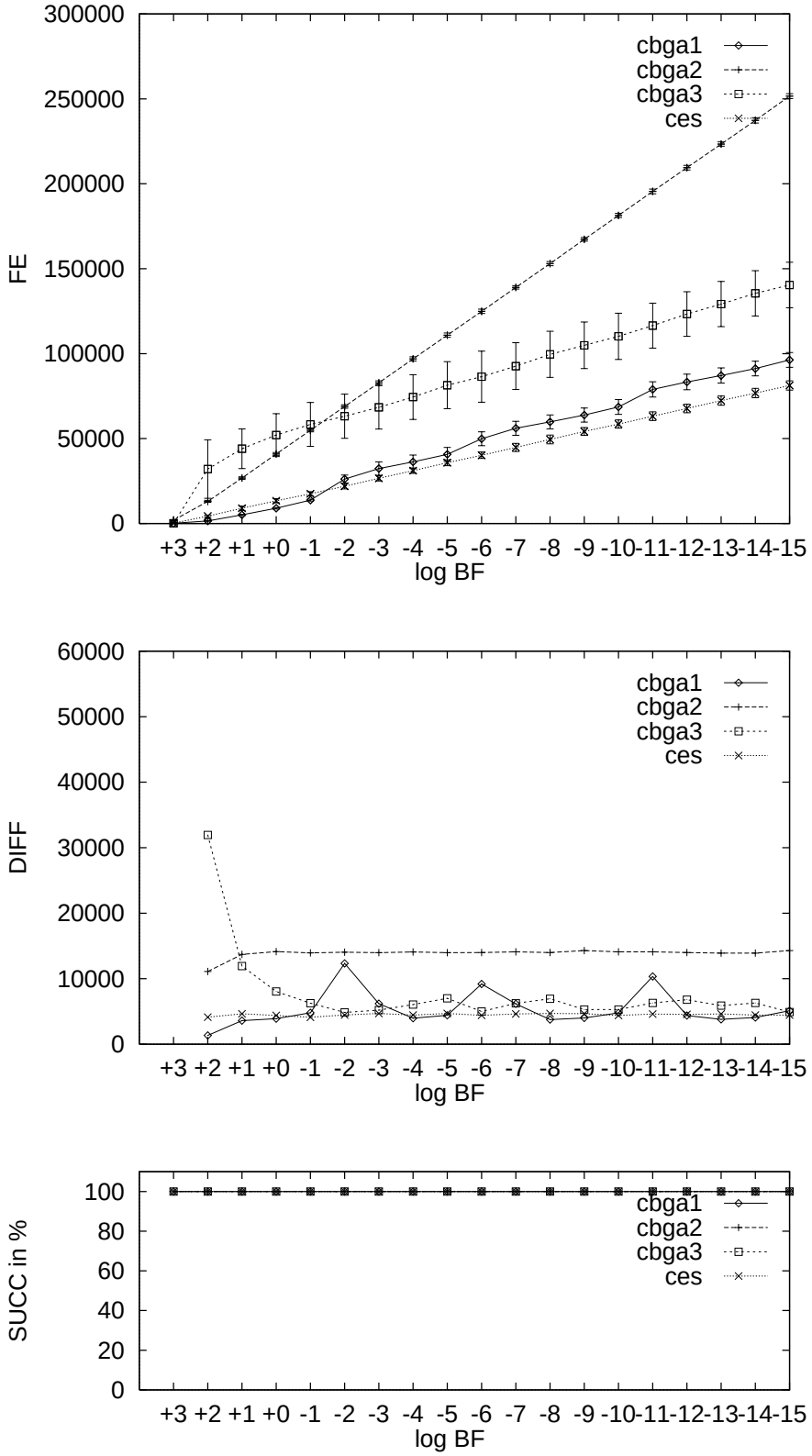

Abbildung 6.13: **Kugelmodell** (F1, n=100): Konvergenzverhalten von Wettbewerbsmodellen.

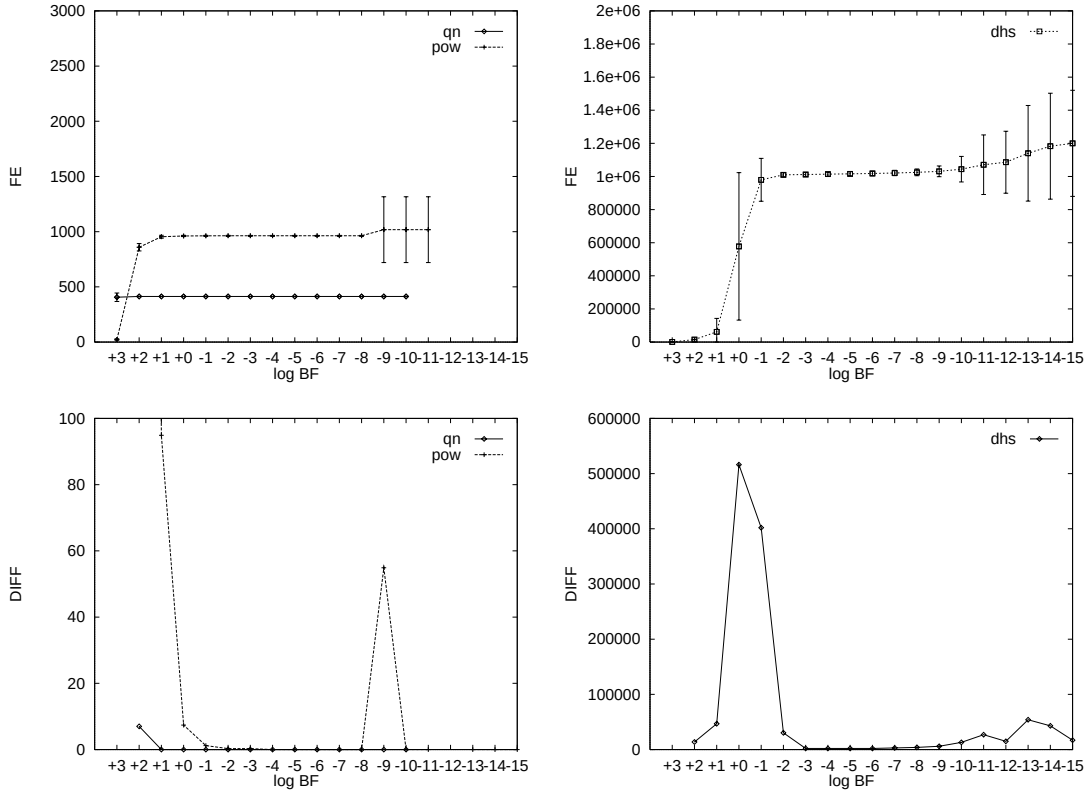

Abbildung 6.14: **Kugelmodell** (F1, n=100): Konvergenzverhalten von Quasi-Newton-Verfahren, Powell-Verfahren und Downhill-Simplex-Verfahren. Da der verwendete Simulator innerhalb der eindimensionalen Optimierer, die das Quasi-Newton-Verfahren und das Powell-Verfahren verwenden, keine Zielfunktionswerte erfaßt, erscheinen deren Kurvenverläufe sprunghaft. Das Quasi-Newton-Verfahren benötigt nur etwa halb soviele Funktionsauswertungen wie das Powell-Verfahren.

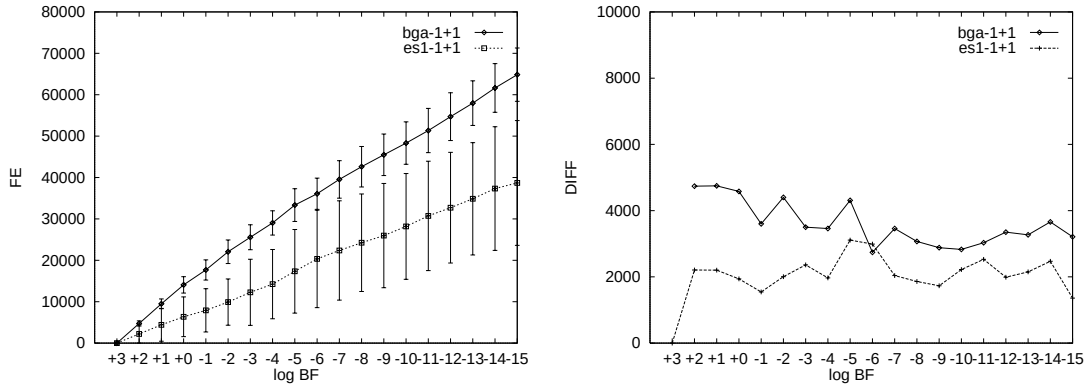

Abbildung 6.15: **Kugelmodell** (F1, n=100): Vergleich der beiden Mutations-Selektions-Verfahren bga-1+1 und es-1+1.

Gewichtetes Kugelmodell (F2)

Die Abbildungen 6.16 bis 6.19 auf den Seiten 159 bis 162 geben das Konvergenzverhalten der Suchverfahren für das 100-dimensionale gewichtete Kugelmodell wieder. Obwohl sich diese Funktion lediglich in der unterschiedlichen Gewichtung der Objektvariablen vom Kugelmodell unterscheidet, zeigen viele der Suchverfahren ein erheblich abweichendes Konvergenzverhalten.

Mit Ausnahme des Powell-Verfahrens (pow) weisen die lokalen Suchverfahren durchweg eine deutlich geringere Konvergenzgeschwindigkeit auf als beim Kugelmodell. So ist der DIFF-Wert des Verfahrens von Solis und Wets (sw), das nach einer Einschwingphase linear konvergiert, mit 30000 dreißig mal höher als beim Kugelmodell. Das Quasi-Newton-Verfahren (qn), das beim Kugelmodell die höchste Konvergenzgeschwindigkeit aufwies, benötigt hier eine recht aufwendige Einschwingphase, nach der es für einen bestimmten Bereich superlinear konvergiert. Das Dynamic-Hillclimbing-Verfahren (dhc) erweist sich in der Anfangsphase als äußerst effizient, termininiert aber häufig mit hohen Zielfunktionswerten. Wie beim Kugelmodell weichen die Resultate des Powell-Verfahrens und des Downhill-Simplex-Verfahrens erheblich von denen der restlichen Suchverfahren ab. Aus diesem Grund ist deren Konvergenzverhalten in Abbildung 6.20 auf Seite 163 noch einmal mit andereren Skalierungen dargestellt. Das Powell-Verfahren benötigt wie beim Kugelmodell 1000 Funktionsauswertungen, um einen Zielfunktionswert von 10^{-10} zu erreichen. Es läßt sich von der unterschiedlichen Gewichtung der Objektvariablen nicht beeinflussen. Das Downhill-Simplex-Verfahren zeigt ein ähnliches Verhalten wie beim Kugelmodell. Nach der ersten Iteration, die aufgrund der oberen Schranke für die Funktionsauswertungen abbricht, erreicht es den Zielfunktionswert 10.0. In den zweiten Iteration konvergiert es zunächst linear mit einem DIFF-Wert von 5000. Ab dem Zielfunktionswert 10^{-5} bewegen sich die DIFF-Werte zwischen 30000 und 150000. Wiederum erreichen alle Läufe den Zielfunktionswert 10^{-15}.

Die Evolutionsstrategie es1 mit einem Strategieparameter weist gegenüber dem Kugelmodell eine drastische Verschlechterung der Konvergenzgeschwindigkeit auf. Ihre Resultate sind aufgrund der großen Unterschiede zu den anderen Suchverfahren in Abbildung 6.21 auf Seite 163 noch einmal gesondert mit einer angepaßten Skalierung dargestellt. Nach einer langen Startphase konvergiert die Evolutionsstrategie es1 ab dem Zielfunktionswert 10^{-4} linear mit einem durchschnittlichen DIFF-Wert von 200000 [4]. Sie erzielt damit nur 1/40 der Konvergenzgeschwindigkeit, die sie beim Kugelmodell erreicht. Zudem weist sie zwischen den Läufen wesentlich größere Schwankungen als beim Kugelmodell auf. Bei einer Zielfunktion mit unterschiedlich gewichteten Objektvariablen reicht ein einziger Strategieparameter für eine effiziente Suche offensichtlich nicht aus.

Die Evolutionsstrategie es2 zeigt sich von der unterschiedlichen Gewichtung der Objektvariablen unbeeindruckt. Sie liefert mit DIFF-Werten zwischen 30000 und 40000 das gleiche Konvergenzverhalten wie beim Kugelmodell. Der Grund für dieses Verhalten liegt darin, daß sie unabhängig von der Zielfunktion alle n Strategieparameter anpaßt. Der damit verbundene Aufwand ist für beide Funktionen gleich groß. Während beim Kugelmodell aufgrund seiner Kugelsymmetrie ein Strategieparameter für alle Objektvariablen ausreicht, sind beim gewichteten Kugelmodell n Strategieparameter erforderlich. Der Aufwand, der

[4] Messungen von Schwefel ergaben für es1 und es2 DIFF-Werte von 97200 beziehungsweise 25500. Der Grund für die Abweichung liegt, wie schon beim Kugelmodell, vermutlich in der unterschiedlichen Implementierung und Parametrisierung der Evolutionsstrategie.

für die Anpassung der n Strategieparameter betrieben werden muß, rentiert sich also erst beim gewichteten Kugelmodell.

Das Wettbewerbsmodell ces, das beide ES-Varianten beinhaltet, liefert mit einem DIFF-Wert von 15000 mehr als die doppelte Konvergenzgeschwindigkeit der Evolutionsstrategie es2. Gegenüber dem Kugelmodell wird die Suche hier nicht durchgängig von einer der beiden Evolutionsstrategien dominiert. Die Entwicklung der Populationsgrößen für diese Zielfunktion ist in Abbildung 5.15 auf Seite 121 dargestellt. Durch die dynamische Anpassung der Populationsgrößen werden beide Suchstrategien in Abhängigkeit ihrer Eignung für die momentane Situation eingesetzt. Wie in Abbildung 6.17 auf Seite 160 zu sehen ist, erzielt das Wettbewerbsmodell in der Anfangsphase durch den verstärkten Einsatz der Population mit nur einem Strategieparameter dieselbe hohe Konvergenzgeschwindigkeit wie die Evolutionsstrategie es1. Sobald die Anpassung der einzelnen Strategieparameter für die Suche vorteilhaft wird, werden mehr Versuche mit der Evolutionsstrategie es2 gemacht. Neben der Konvergenzgeschwindigkeit wird durch das Wettbewerbsmodell auch dessen Robustheit bezüglich der Funktionsauswertungen gesteigert. Die in Abbildung 6.17 eingezeichneten Standardabweichungen des Wettbewerbsmodells sind erheblich geringer als bei den beiden einfachen Evolutionsstrategien.

Der Genetische Algorithmus terminiert, wie schon beim Kugelmodell, aufgrund der zu geringen Konvergenzgeschwindigkeit weit entfernt von dem hier betrachteten Fitnessbereich. Kein Lauf erreicht den Zielfunktionswert 1000.0.

Die BGA lassen sich durch die unterschiedlichen Gewichtungen der Objektvariablen kaum beeinflussen. Die in Abbildung 6.18 dargestellten Resultate entsprechen im wesentlichen denen des Kugelmodells. Der Grund für dieses Verhalten ist auf die robuste Mutationsverteilung des BGA zurückzuführen, die in der Lage ist, unterschiedliche Gewichtung von Objektvariablen zu kompensieren. Eine Ausnahme bildet der BGA mit Hauptachsentransformation (bga-pca), der zunächst das Koordinatensystem an die Orientierung der Zielfunktion anpassen muß. Nachdem die Anpassung erfolgt ist, konvergiert er wie beim Kugelmodell linear mit einem DIFF-Wert von 6500. Im Unterschied zum Standard-BGA tritt in dem vorliegenden Fitnessbereich keine Verschlechterung der Konvergenzgeschwindigkeit in der Nähe des Optimums auf, da die Objektvariablen durch die Rotation des Koordinatensystems beliebig kleine Werte annehmen können.

Die cBGA zeigen hier das gleiche Konvergenzverhalten wie beim Kugelmodell. Sie konvergieren über den gesamten vorgegebenen Fitnessbereich linear mit den gleichen DIFF-Werten wie beim Kugelmodell.

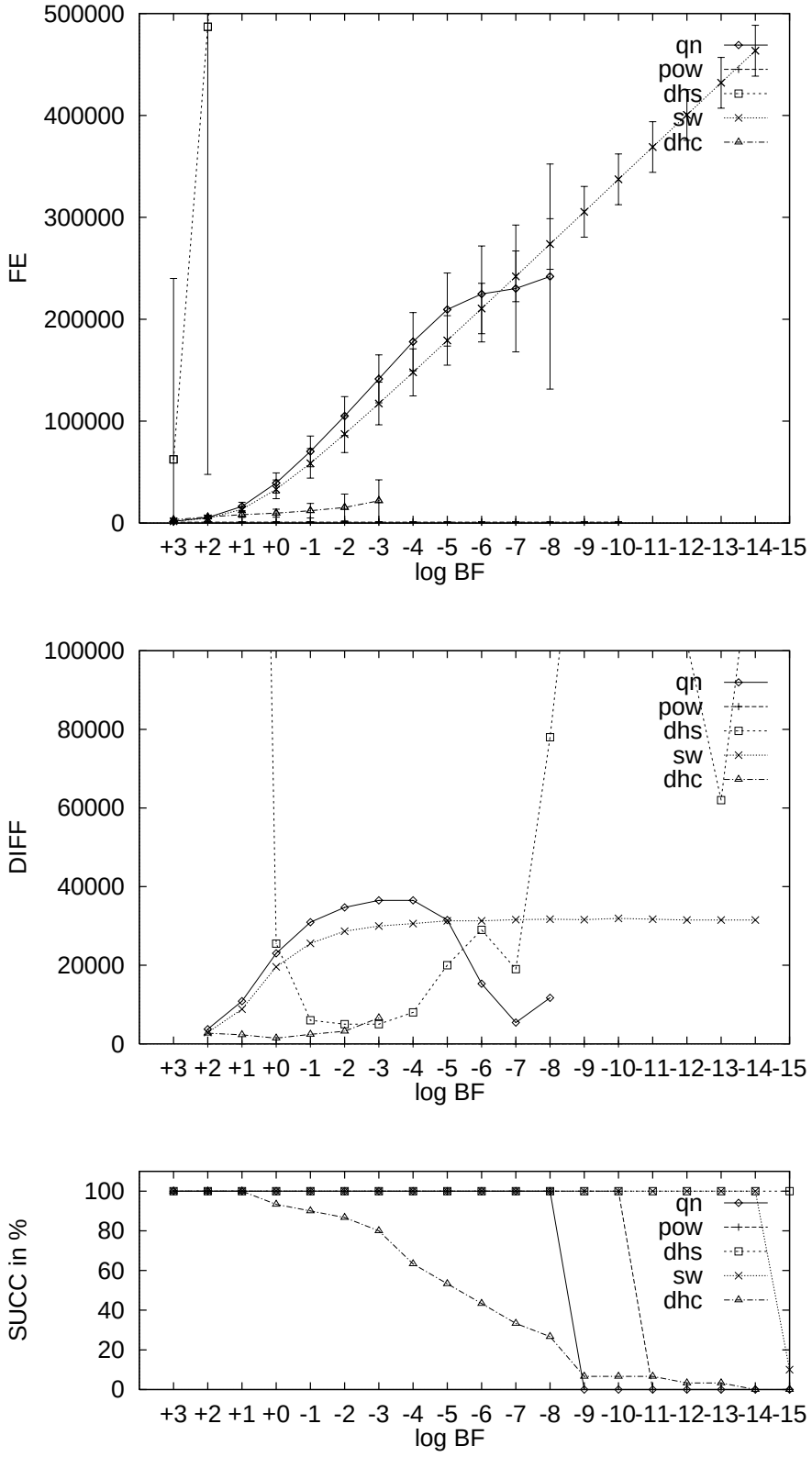

Abbildung 6.16: **Gewichtetes Kugelmodell** (F2, n=100): Konvergenzverhalten von lokalen Suchverfahren.

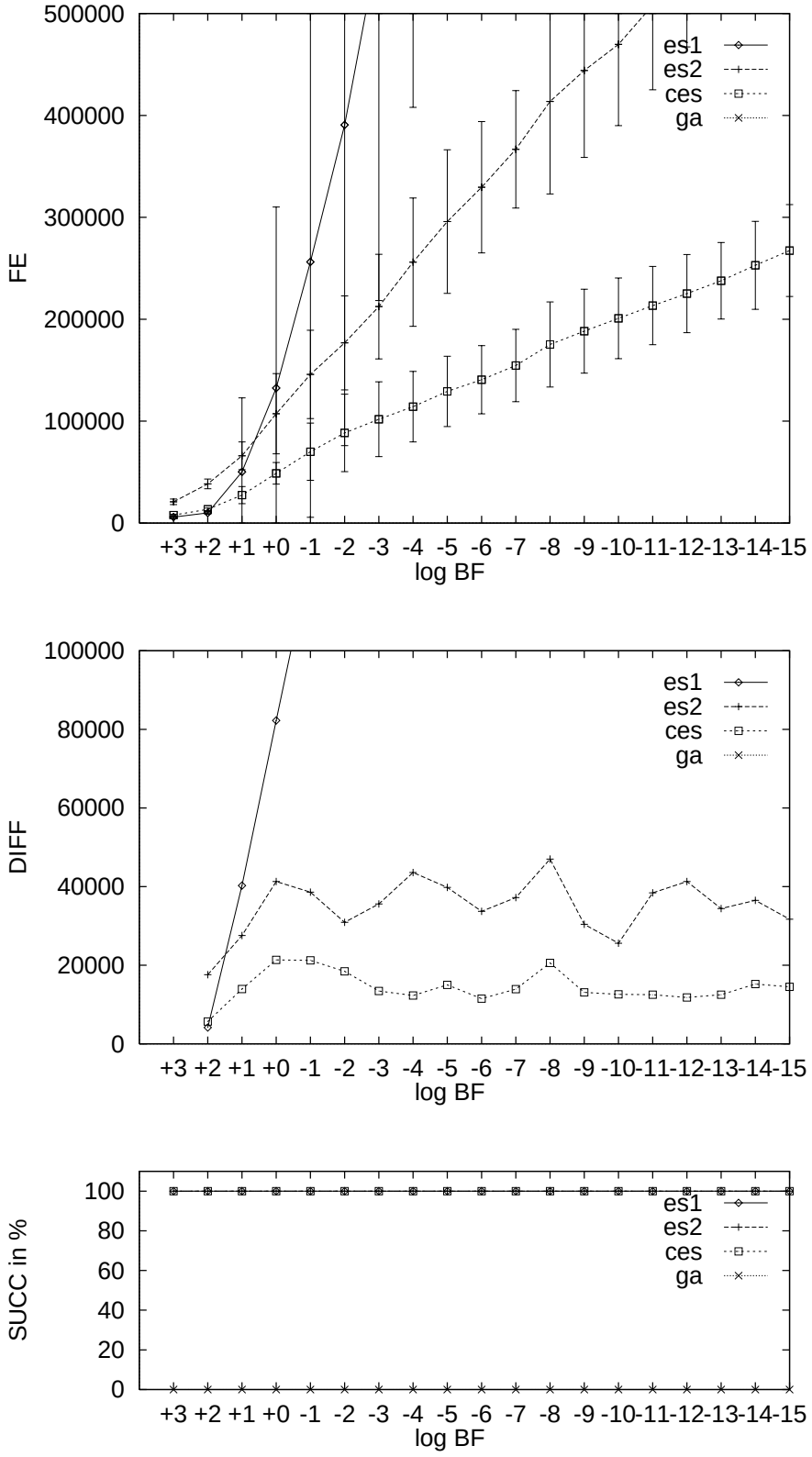

Abbildung 6.17: **Gewichtetes Kugelmodell** (F2, n=100): Konvergenzverhalten von Evolutionsstrategien und Genetischem Algorithmus.

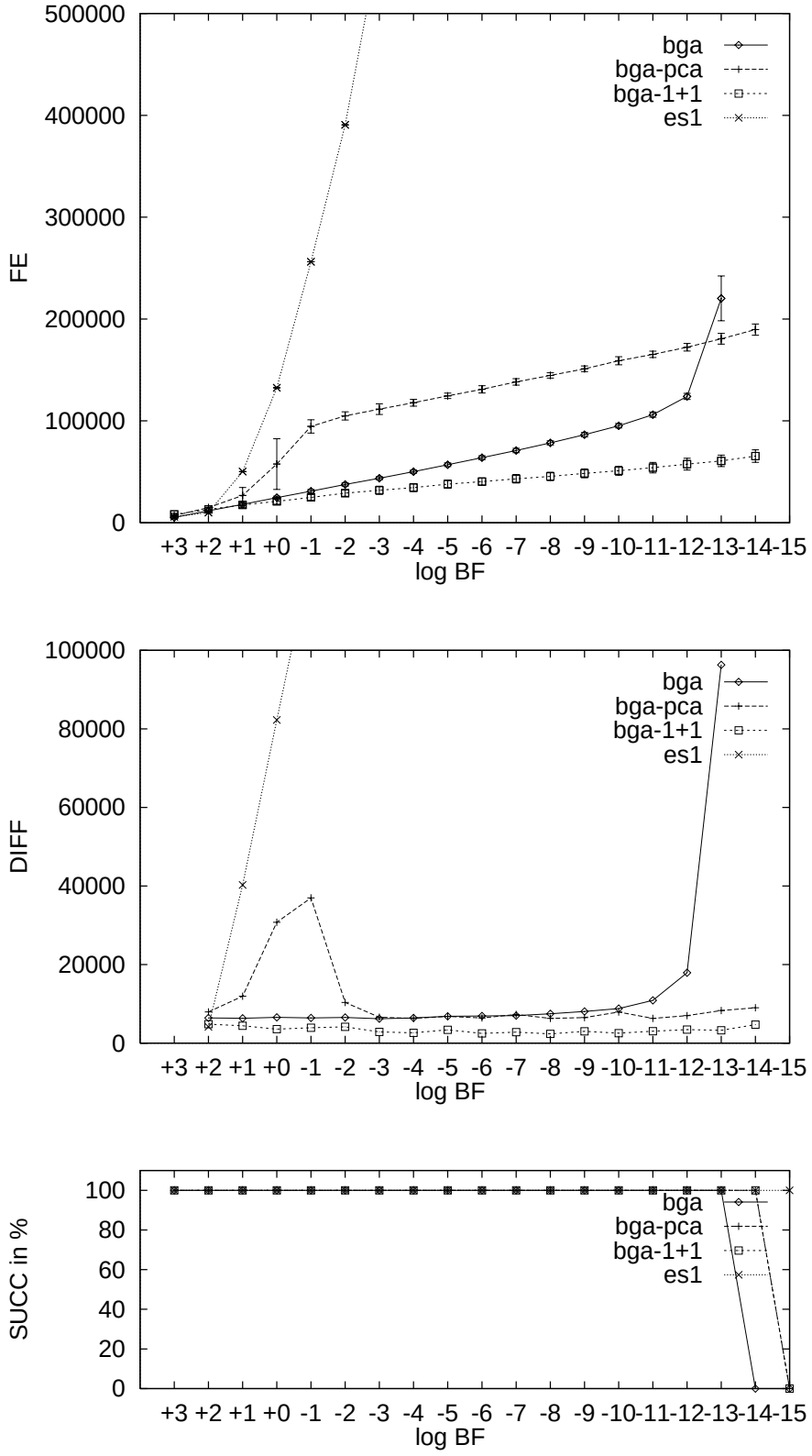

Abbildung 6.18: **Gewichtetes Kugelmodell** (F2, n=100): Konvergenzverhalten von BGA-Varianten und Evolutionsstrategie `es1`.

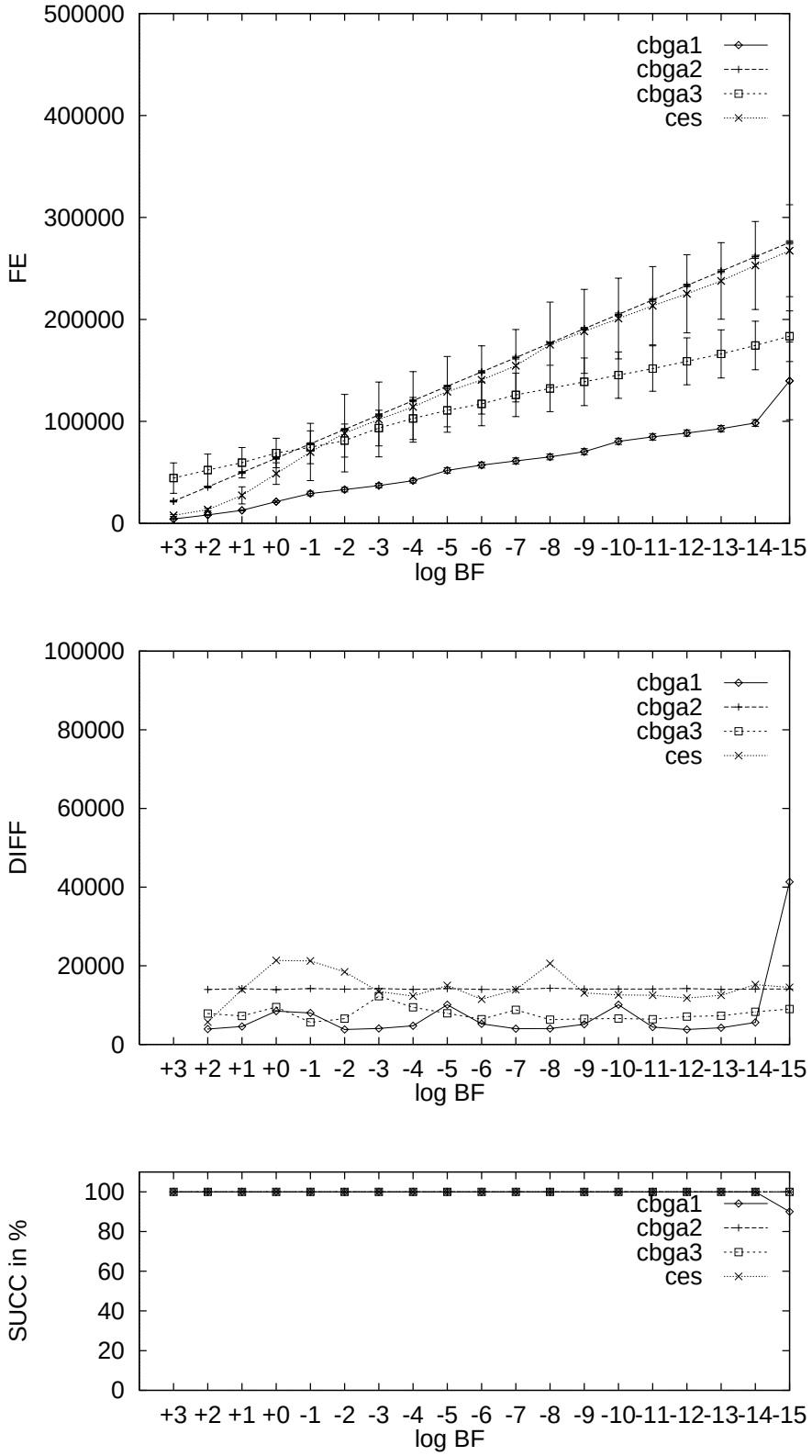

Abbildung 6.19: **Gewichtetes Kugelmodell** (F2, n=100): Konvergenzverhalten von Wettbewerbsmodellen.

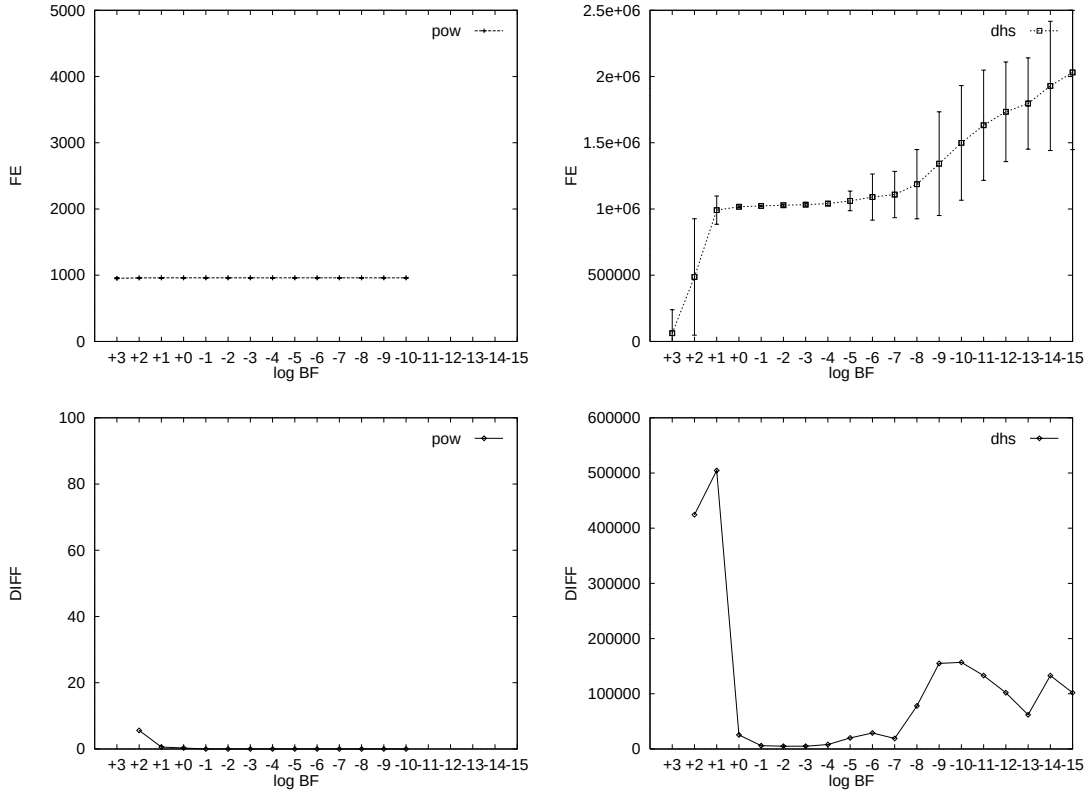

Abbildung 6.20: **Gewichtetes Kugelmodell** (F2, n=100): Konvergenzverhalten von Powell-Verfahren und Downhill-Simplex-Verfahren. Da der verwendete Simulator innerhalb des eindimensionalen Optimierers, den das Powell-Verfahren verwendet, keine Zielfunktionswerte erfaßt, erscheinen dessen Kurvenverläufe sprunghaft.

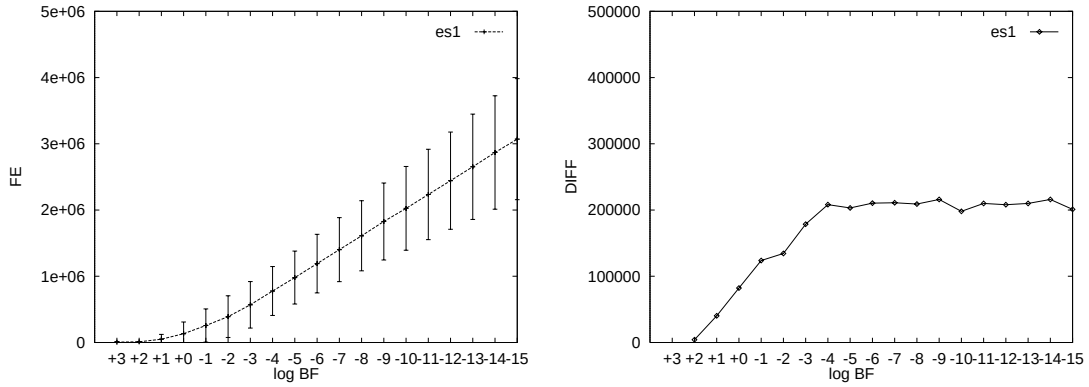

Abbildung 6.21: **Gewichtetes Kugelmodell** (F2, n=100): Konvergenzverhalten der Evolutionsstrategie es1.

Schwefels Funktion (F3)

Die Resultate der Suchverfahren für die 100-dimensionale Funktion von Schwefel sind in den Abbildungen 6.22 bis 6.25 auf den Seiten 166 bis 169 dargestellt. Wie an der hohen Anzahl der Funktionsauswertungen, die zur Annäherung an die vorgegebenen Fitnesswerte benötigt werden, zu sehen ist, bereitet diese Funktion allen Suchverfahren aufgrund der korrelierten Objektvariablen und der hohen Konditionszahl von 16400 (siehe Abschnitt 6.2.3) große Schwierigkeiten.

Das Konvergenzverhalten der Suchverfahren unterscheidet sich bei dieser Funktion zum Teil erheblich von dem Konvergenzverhalten für die beiden Kugelmodelle. So erreicht das Powell-Verfahren in nur 80 % der Läufe den Zielfunktionswert 10.0. Dieses schlechte Verhalten deckt sich mit den Ergebnissen, die Schwefel mit dem Powell-Verfahren bei dieser Funktion in hohen Dimensionen erzielte ([Sch77], S.320). Die hohe Konditionszahl führt bei den analytischen Suchverfahren zu numerischen Instabilitäten. Das Quasi-Newton-Verfahren konvergiert nur sublinear und terminiert mit einem durchschnittlichen Zielfunktionswert von $8 \cdot 10^{-7}$. Dennoch erzielt es bis zum Fitnesswert 10^{-4} die höchste Konvergenzgeschwindigkeit der lokalen Suchverfahren. Das Downhill-Simplex-Verfahren, das bei beiden Kugelmodellen durch eine sehr geringe Konvergenzgeschwindigkeit auffiel, erweist sich hier als das beste der lokalen Suchverfahren. Bis zum Zielfunktionswert 10^{-5} konvergiert es annähernd linear mit DIFF-Werten zwischen 150000 und 200000. Um von 10^{-5} auf den Zielfunktionswert 10^{-15} zu kommen, benötigt das Verfahren nur noch weitere 170000 Funktionsauswertungen. Die beiden stochastischen lokalen Suchverfahren (dhc, sw) brechen bei dieser Funktion vollständig ein. Während das Dynamic-Hillclimbing-Verfahren durchschnittlich mit einem Zielfunktionswert von 385300 terminiert, zeigt das Verfahren von Solis und Wets mit einem DIFF-Wert von 6000000 eine sehr niedrige Konvergenzgeschwindigkeit. Aufgrund der großen Unterschiede zu den anderen Suchverfahren ist das Konvergenzverhalten des Verfahrens von Solis und Wets noch einmal in Abbildung 6.26 auf Seite 170 dargestellt.

Die Evolutionsstrategie es1 weist über den gesamten Fitnessbereich ein lineares Konvergenzverhalten mit einem DIFF-Wert von 430000 auf. Der Evolutionsstrategie es2 mit n Strategieparametern divergiert dagegen in allen Läufen [5]. Das Wettbewerbsmodell erreicht mit einem durchschnittlichen DIFF-Wert von 200000 die doppelte Konvergenzgeschwindigkeit der Evolutionsstrategie es1. Der Genetische Algorithmus (ga) terminiert mit einem durchschnittlichen Fitnesswert von 155500.

Die einfachen BGA, deren Simulationsergebnisse in Abbildung 6.24 auf Seite 168 dargestellt sind, weisen bis zur Fitness 10^{-9} ein annähernd lineares Konvergenzverhalten auf. Sie brechen dann, wie auch bei den anderen Funktionen, aufgrund ihrer zu großen minimalen Schrittweite ein. Das Mutations-Selektions-Verfahren (bga-1+1) und der Standard-BGA (bga), deren Mutationsoperator nur Schritte entlang der Koordinatenachsen macht, haben bei dieser Funktion erhebliche Schwierigkeiten. Die DIFF-Werte liegen hier bei 950000 beziehungsweise 1100000. Der BGA bga-pca, der das Koordinatensystem an die Orientierung der Fitnesslandschaft anpaßt, konvergiert annähernd linear. Die DIFF-Werte nehmen mit

[5] Die Divergenz kann darauf zurückzuführen sein, daß die hier eingesetzte Rekombination $\mathbb{R}^{\mathrm{ES}}_{di\_}$, die im Gegensatz zur empfohlenen Standardrekombination $\mathbb{R}^{\mathrm{ES}}_{dI\_}$ aus Gleichung 3.51 nur sexuelle Rekombinationsoperatoren verwendet, für diese Funktion ungünstig ist.

sinkendem Fitnesswert von 300000 bis auf 400000 zu. Der bga-pca erzielt damit eine mehr als doppelt so hohe Konvergenzgeschwindigkeit wie der Standard-BGA.

Wie in Abbildung 6.25 auf Seite 169 zu sehen ist, führt das Wettbewerbsmodell cbga1 zu einer erheblichen Verbesserung der Konvergenzeigenschaften des Standard-BGA. Es konvergiert bis zur Fitness 10^{-11} linear mit einem DIFF-Wert von 380000 und erreicht damit die dreifache Konvergenzgeschwindigkeit des Standard-BGA. Ein noch besseres Konvergenzverhalten erzielt das Wettbewerbsmodell cbga3 mit Strategiemodifikation, dessen Konvergenzgeschwindigkeit mit einem DIFF-Wert von 200000 fünf mal so hoch ist wie die des Standard-BGA.

Abbildung 6.27 auf Seite 170 zeigt eindrucksvoll, wie das Konvergenzverhalten des BGA durch die Hinzunahme von konkurrierenden Populationen verbessert wird. Dargestellt sind die Simulationsergebnisse eines Standard-BGA und dreier Wettbewerbsmodelle, die sich lediglich in der Anzahl der Populationen unterscheiden. Mit zunehmender Populationsanzahl kann die Konvergenzgeschwindigkeit sukzessive gesteigert werden. Die DIFF-Werte können so von 1100000 für den Standard-BGA auf 240000 für den cBGA mit acht Populationen reduziert werden.

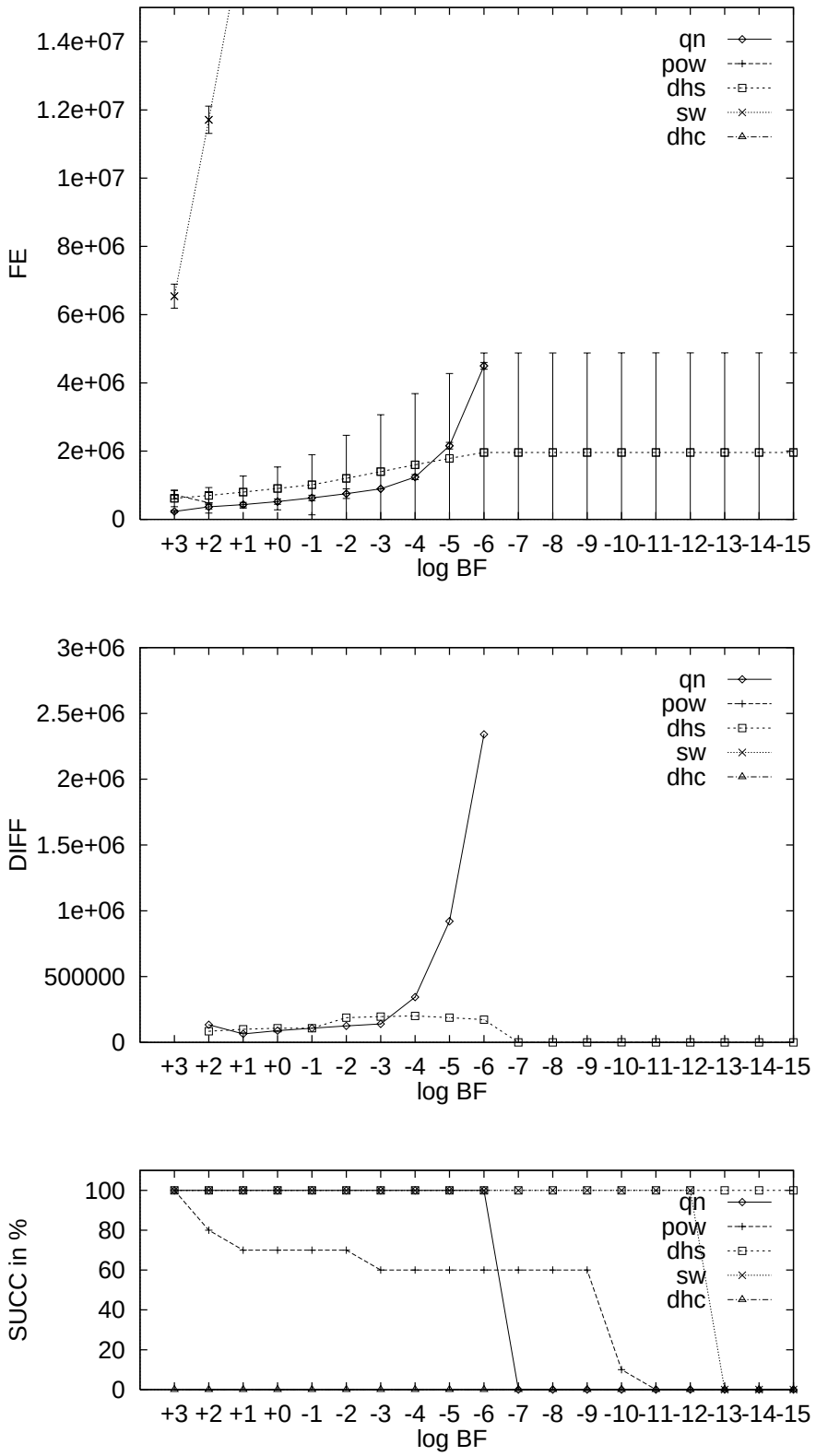

Abbildung 6.22: **Schwefels Funktion** (F3, n=100): Konvergenzverhalten von lokalen Suchverfahren.

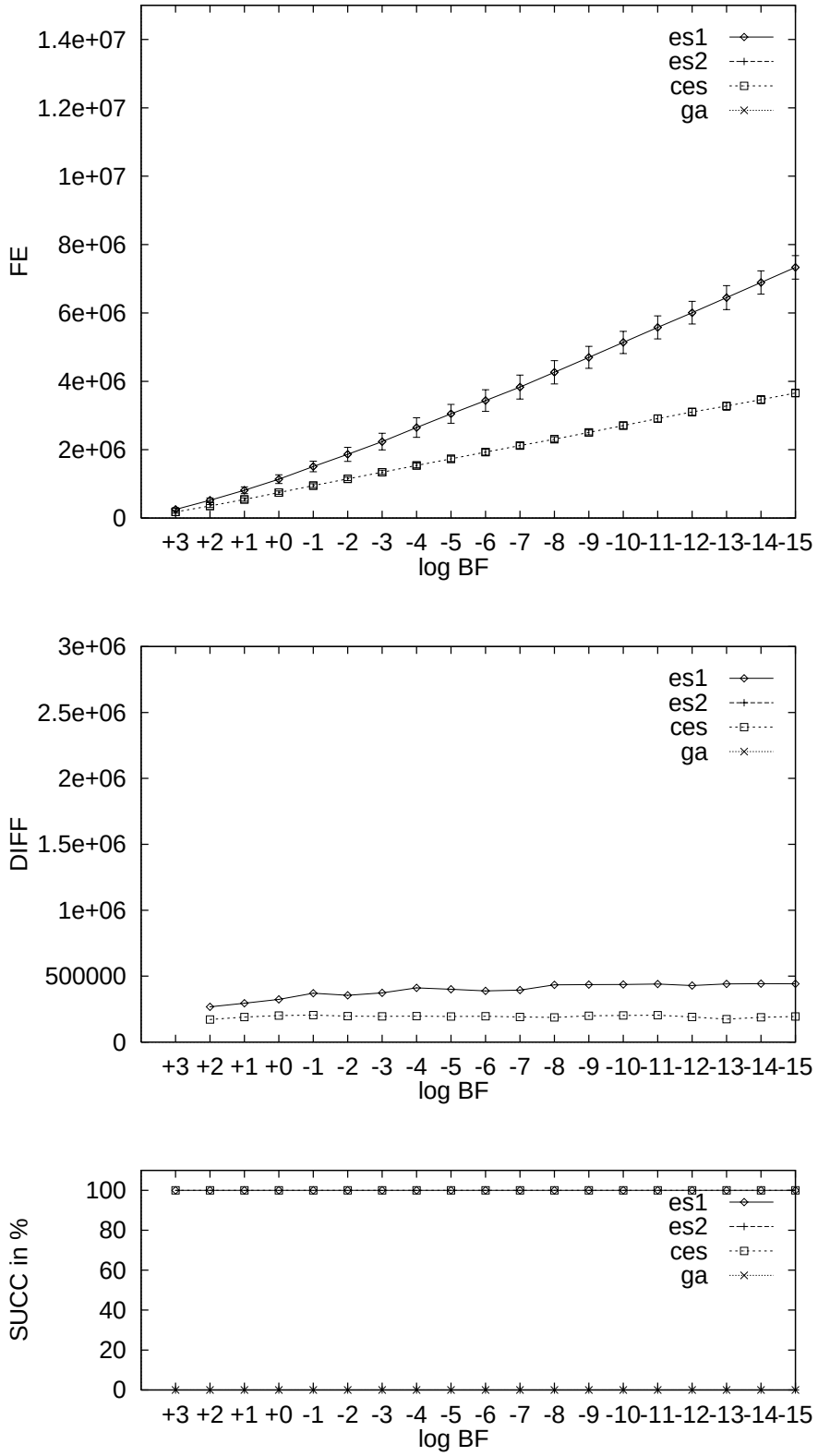

Abbildung 6.23: **Schwefels Funktion** (F3, n=100): Konvergenzverhalten von Evolutionsstrategien und Genetischem Algorithmus.

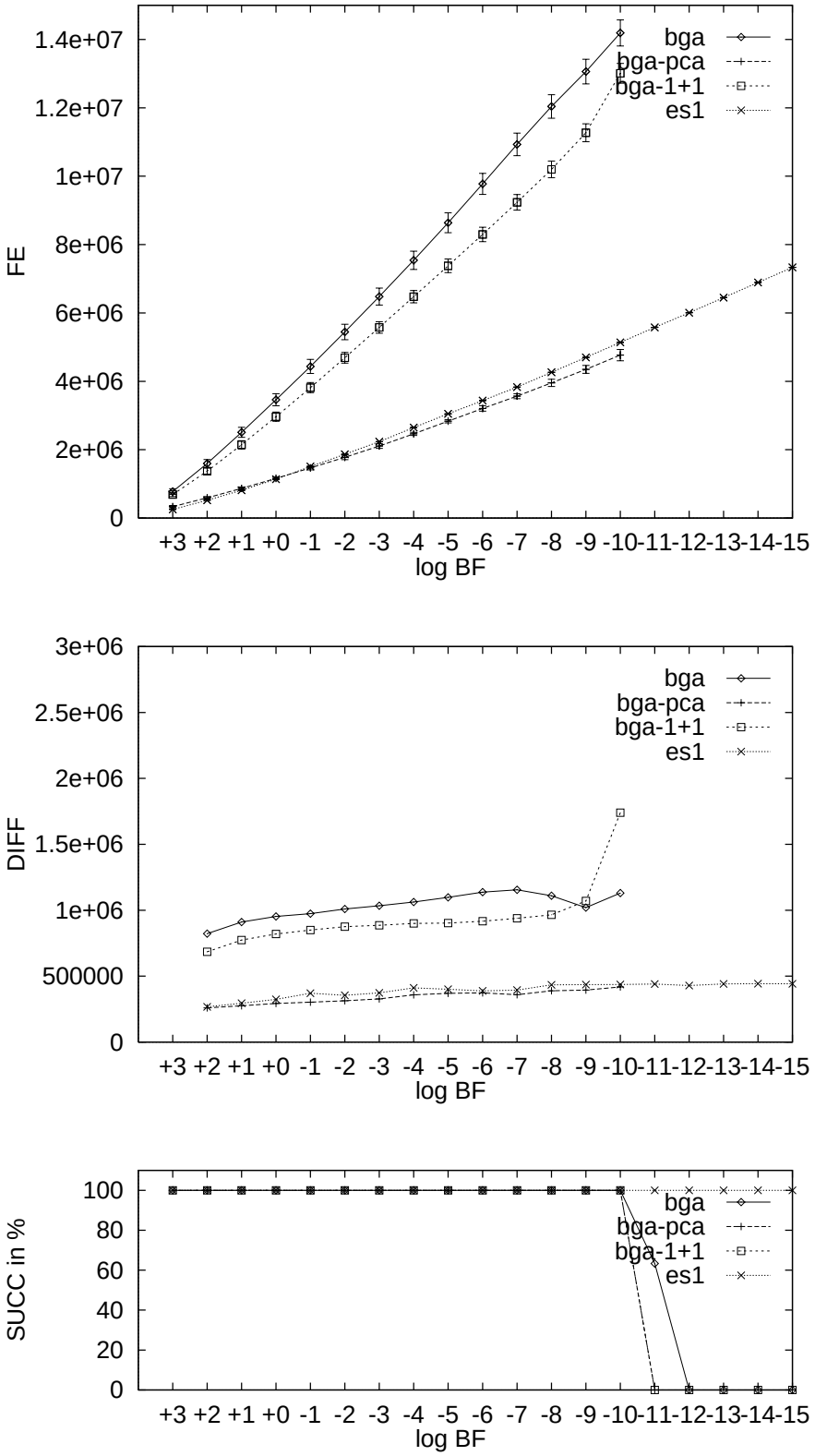

Abbildung 6.24: **Schwefels Funktion** (F3, n=100): Konvergenzverhalten von BGA-Varianten und Evolutionsstrategie es1.

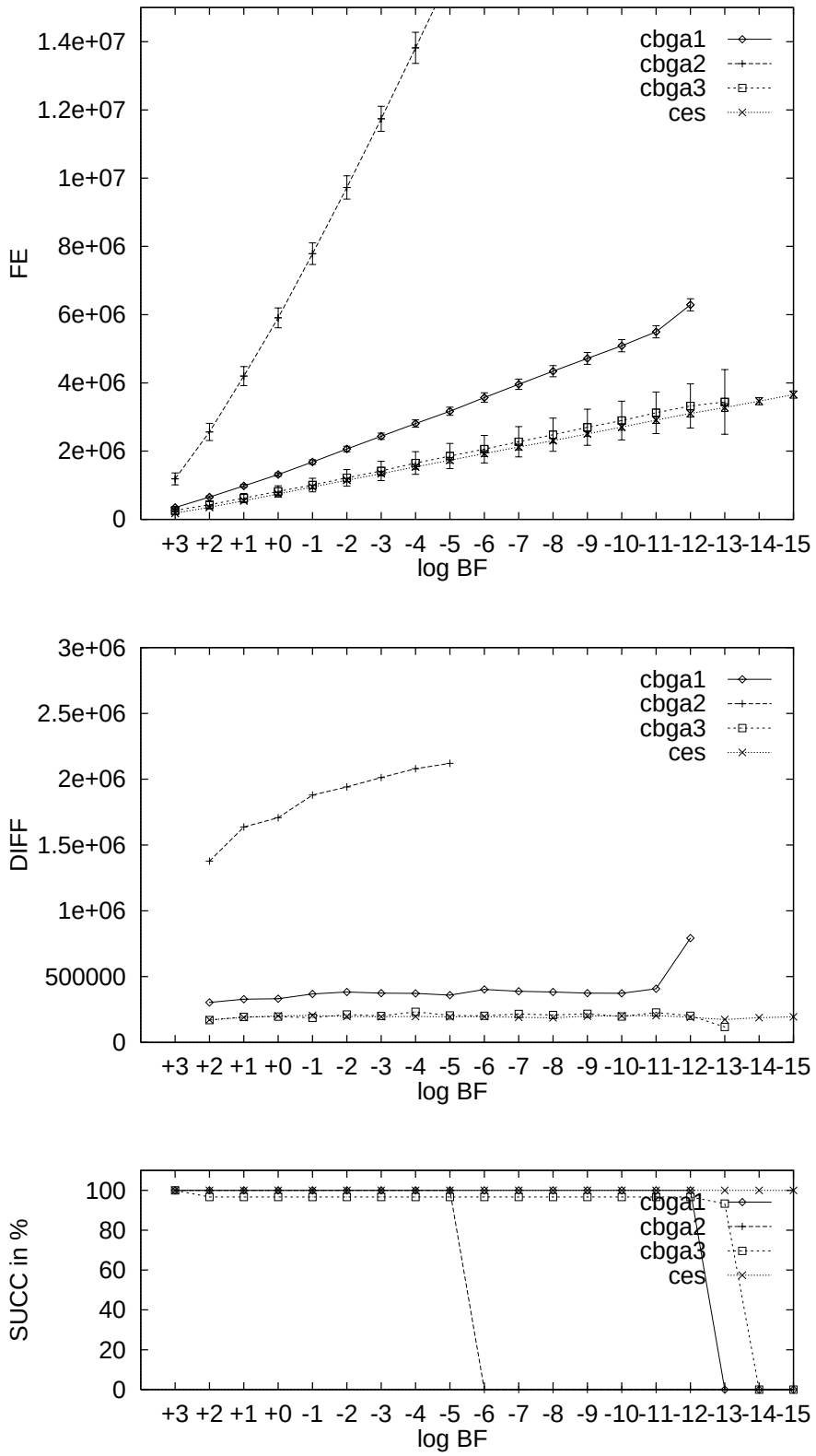

Abbildung 6.25: **Schwefels Funktion** (F3, n=100): Konvergenzverhalten von Wettbewerbsmodellen.

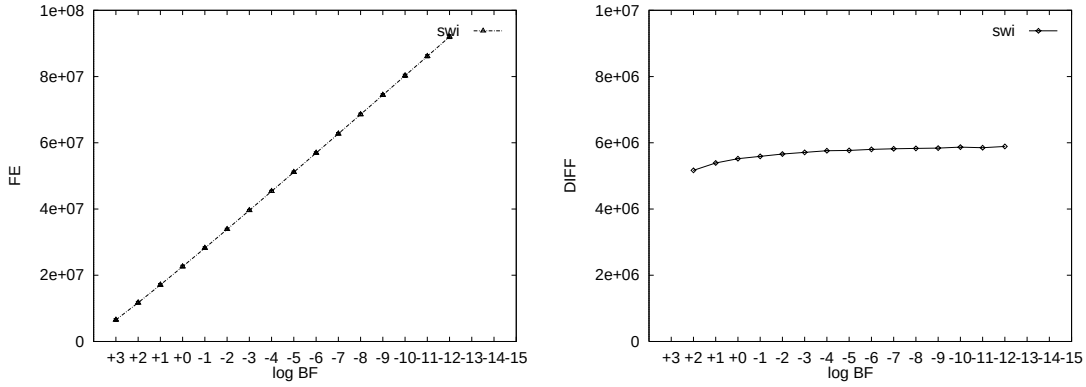

Abbildung 6.26: **Schwefels Funktion** (F3, n=100): Konvergenzverhalten des Verfahrens von Solis und Wets.

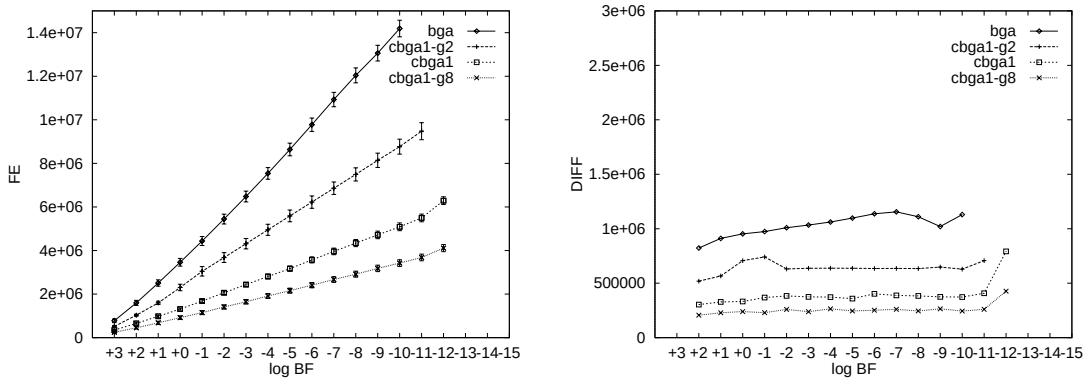

Abbildung 6.27: **Schwefels Funktion** (F3, n=100): Konvergenzverhalten des `cbga1` für verschiedene Anzahlen von konkurrierenden Populationen. `bga` bezeichnet den aus einer Population bestehenden Standard-BGA. `cbga1` ist der aus vier Populationen bestehende cBGA mit unterschiedlichen Schrittweitenparameter. `cbga1-g2` und `cbga1-g8` sind die entsprechenden cBGA mit zwei beziehungsweise acht Populationen.

Rosenbrocks Funktion (F4)

Das Konverenzverhalten der Suchverfahren für die 30-dimensionale Funktion von Rosenbrock ist in den Abbildungen 6.28 bis 6.31 auf den Seiten 173 bis 176 dargestellt. Bei den Simulationsergebnissen fällt auf, daß in der Anfangsphase alle Verfahren einen sehr großen Aufwand betreiben müssen, um die Fitness von 10.0 zu unterschreiten. Dieser Effekt ist auf den Suchbereich um den Punkt $\vec{x}^T = (1, 0, \ldots, 0)$ mit dem Funktionswert 3.987 zurückzuführen, aus dem nur ein sehr schmales gekrümmtes Tal herausführt. Das Verlassen dieses Bereichs gelingt vielen Verfahren nicht immer oder nur unter sehr großem Aufwand.

Das beste Konvergenzverhalten erzielen bei dieser Funktion das Powell-Verfahren und das Dynamic-Hillclimbing-Verfahren. Während das Dynamic-Hillclimbing-Verfahren bis zum Zielfunktionswert 10^{-14} mit einem DIFF-Wert von 13000 linear skaliert, erreicht das Powell-Verfahren bei einer höheren Konvergenzgeschwindigkeit nur den Zielfunktionswert 10^{-10}. Das Konvergenzverhalten dieser beiden Verfahren ist in Abbildung 6.33 auf Seite 177 noch einmal separat dargestellt. Das Verfahren von Solis und Wets konvergiert linear. Mit einem DIFF-Wert von 130000 beträgt seine Konvergenzgeschwindigkeit nur eine Zehntel der Konvergenzgeschwindigkeit des Dynamic-Hillclimbing-Verfahrens. Das Quasi-Newton-Verfahren konvergiert sublinear und erreicht in keinem Lauf einen Fitnesswert, der unter 10^{-7} liegt. Das Downhill-Simplex-Verfahren versagt bei dieser Funktion vollkommen. Es erreicht in keinem Lauf einen Fitnesswert, der unter 1.0 liegt.

Der Evolutionsstrategie es1 mit nur einem Strategieparameter gelingt es in 20 % der Läufe nicht, das zweite Attraktorgebiet zu verlassen. In den erfolgreichen Läufen zeigt sie ein unregelmäßiges Konvergenzverhalten mit einer niedrigen Konvergenzgeschwindigkeit, die hohen Schwankungen unterworfen ist. Zudem terminiert sie mit Fitnesswerten, die größer als 10^{-8} sind. Demgegenüber weist die Evolutionsstrategie es2 ein erheblich robusteres und effizienteres Konvergenzverhalten auf. In fast allen Läufen erreicht sie den Fitnesswert 10^{-15}. Sie skaliert linear mit einem DIFF-Wert von 150000. Durch das Wettbewerbsmodell ces kann die Konvergenzgeschwindigkeit der Evolutionsstrategie es2 noch einmal erheblich gesteigert werden. Ihr durchschnittlicher DIFF-Wert liegt bei bei 90000. Der Genetische Algorithmus (ga) terminiert aufgrund seiner zu geringen Konvergenzgeschwindigkeit mit einem durchschnittlichen Fitnesswert von 304.1.

Die BGA zeichnen sich gegenüber den lokalen Suchverfahren durch ihre größere Robustheit aus. Sie erreichen in allen Simulationsläufen Fitnesswerte bis 10^{-10}. Der Standard-BGA erzielt bei dieser Funktion nur ein sublineares Konvergenzverhalten. Abbildung 6.30 auf Seite 175 zeigt, daß es dem BGA mit Hauptachsentransformation (bga-pca) nur schwer gelingt, das Basissystem an die gekrümmte Fitnesslandschaft anzupassen. Nach einer aufwendigen Startphase konvergiert er dann aber sogar superlinear. Die DIFF-Werte bleiben allerdings auf hohem Niveau. Mit Hilfe der Wettbewerbsmodelle kann die Effizienz des Standard-BGA wieder erheblich gesteigert werden. Die größte Verbesserung erzielt der cBGA mit Strategiemodifikation (cbga3). Er weist ein lineares Konvergenzverhalten mit einem DIFF-Wert von ca. 200000 auf. Die Effizienzsteigerung geht allerdings zu Lasten der Robustheit. 20 Prozent der Läufe terminieren im zweiten Attraktorgebiet.

Der cbga2 erzielt aufgrund der großen rekombinationsbasierten Population, die die 15-fache Größe der üblichen BGA annehmen kann, eine nur geringe Konvergenzgeschwindigkeit. Im Gegensatz zu den beiden anderen Wettbewerbsmodellen gelingt es diesem

Verfahren jedoch, das globale Attraktorgebiet in allen Läufen zu erreichen.

Rosenbrocks Funktion hat in dieser Form die Eigenschaft, daß benachbarte Objektvariablen stark korreliert sind. Die Stärke der Korrelation sinkt mit zunehmendem Abstand. Falls bekannt ist, daß eine Zielfunktion diese Eigenschaft hat, kann dies durch einen speziellen Mutationsoperator, der die Korrelation benachbarter Objektvariablen berücksichtigt, ausgenutzt werden. Ein solcher Mutationsoperator ist die in Abschnitt 3.5.5 eingeführte Mehrachsen-BGA-Mutation. Die Simulationsergebnisse eines BGA mit Mehrachsen-BGA-Mutation (bga-bma) sind in Abbildung 6.32 auf Seite 177 dargestellt. Um die hohe Konvergenzgeschwindigkeit dieses Mutationsoperators darzustellen, sind die Funktionsauswertungen auch für Fitnesswerte eingezeichnet, die von weniger als den geforderten 80 % der Läufe erreicht werden. Kombiniert man diesen Mutationsoperator, analog zu Wettbewerbsmodell cbga2, mit einer rekombinationsgetriebenen Breitensuche, ergibt sich ein Wettbewerbsmodell, das die Robustheit der Breitensuche und die Effizienz dieses speziellen Mutationsoperators vereinigt. Die Simulationsergebnisse dieses Verfahrens sind ebenfalls in Abbildung 6.32 auf Seite 177 dargestellt. Bedingt durch die anfängliche aufwendige Breitensuche liegt die Effizienz etwas unter der des einfachen BGA. Sobald die Varianz aus der rekombinationsgetriebenen Population verschwunden ist und die Population mit der Mehrachen-BGA-Mutation die Suche dominiert, nähert sich die Konvergenzgeschwindigkeit an die des einfachen BGA an. Der durchschnittliche DIFF-Wert dieses Wettbewerbsmodells ist 30000.

Das Wettbewerbsmodell mit Mehrachsen-BGA-Mutation (cbga2-bma) macht deutlich, wie verschiedene, hoch spezialisierte Suchstrategien miteinander kombiniert werden können, so daß ein breites Spektrum von Zielfunktionen effizient optimiert werden kann. Die Kosten für eine nicht benötigte Suchstrategie entsprechen der minimalen Größe der dazugehörigen Population.

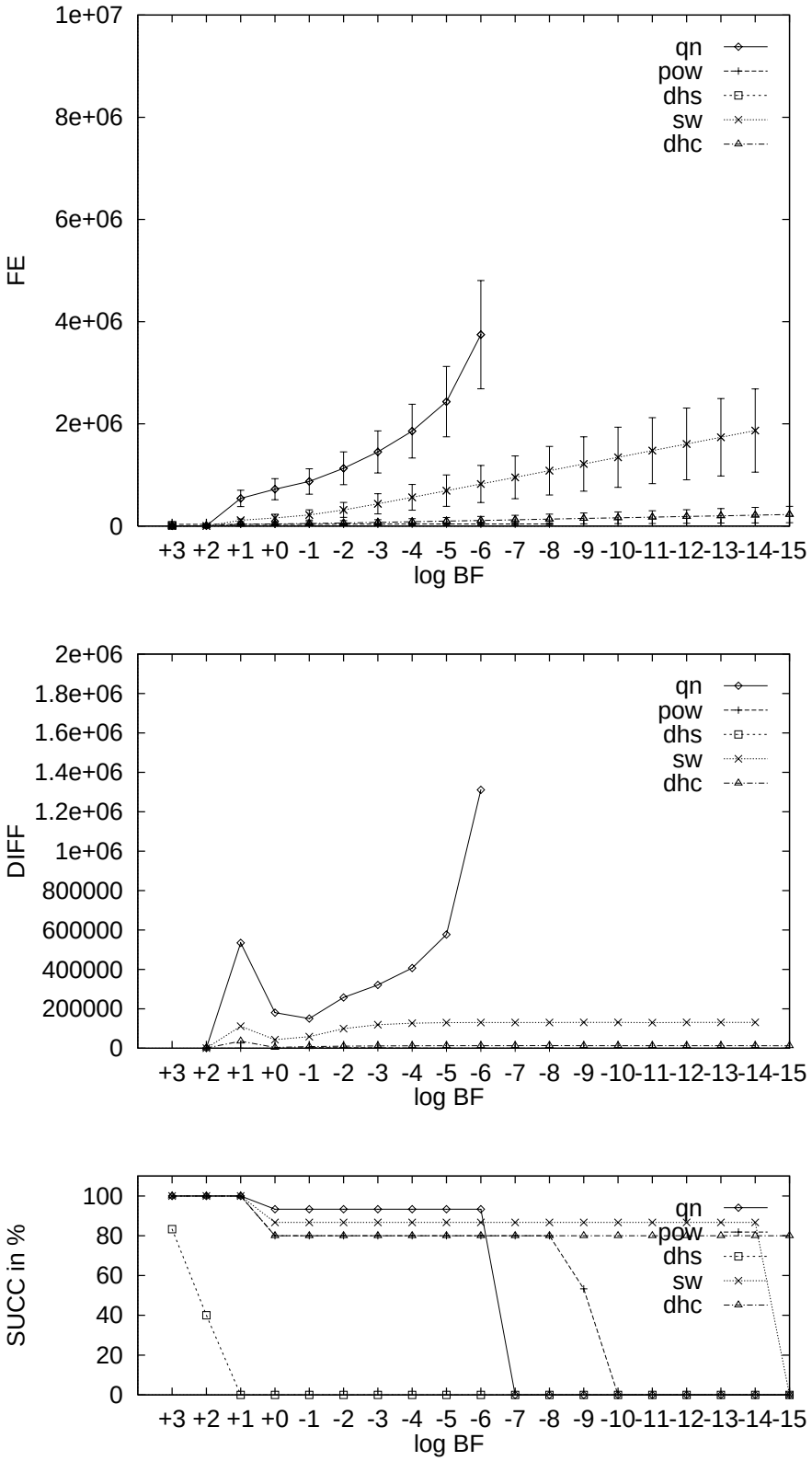

Abbildung 6.28: **Rosenbrocks Funktion** (F4, n=30): Konvergenzverhalten von lokalen Suchverfahren.

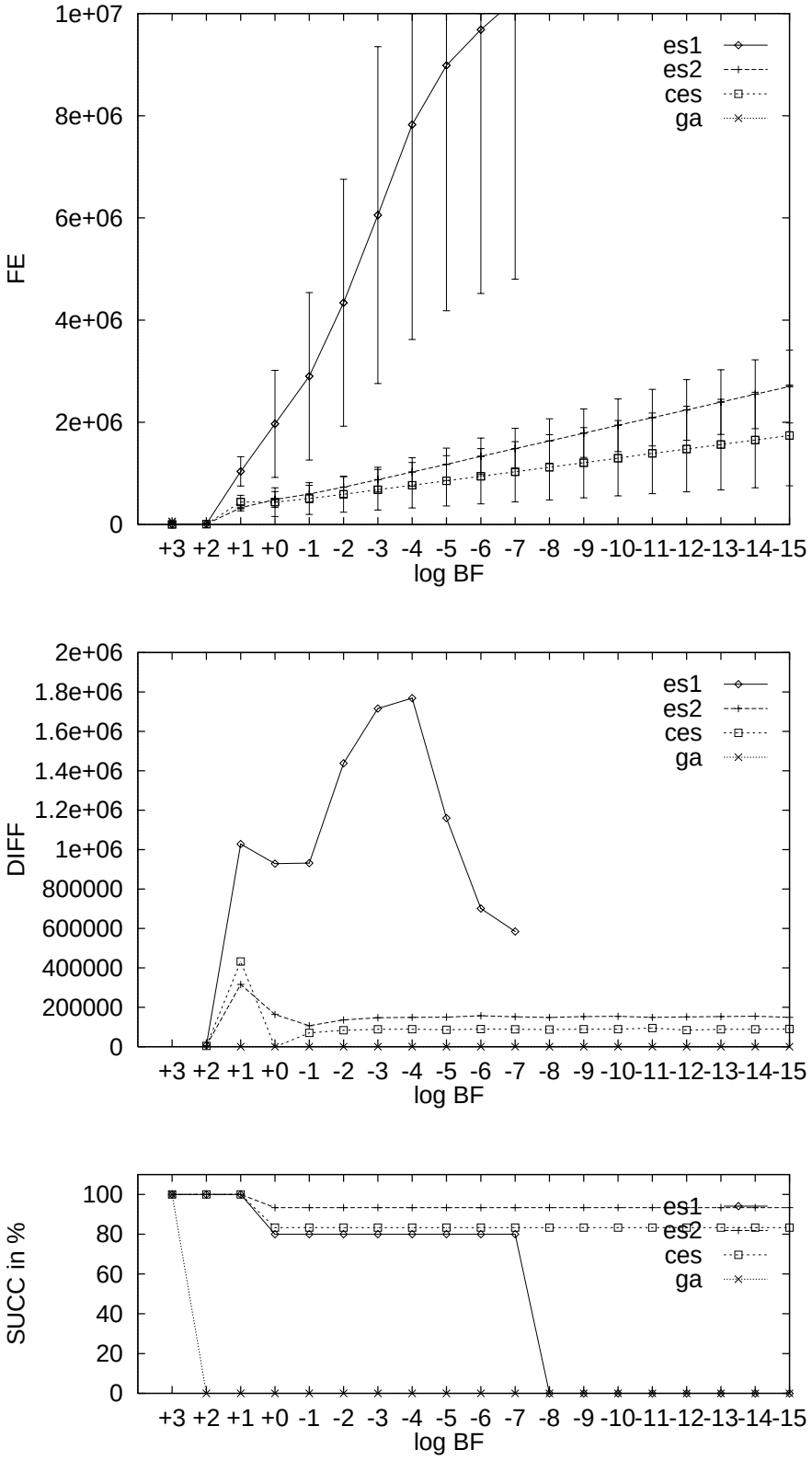

Abbildung 6.29: **Rosenbrocks Funktion** (F4, n=30): Konvergenzverhalten von Evolutionsstrategien und Genetischem Algorithmus.

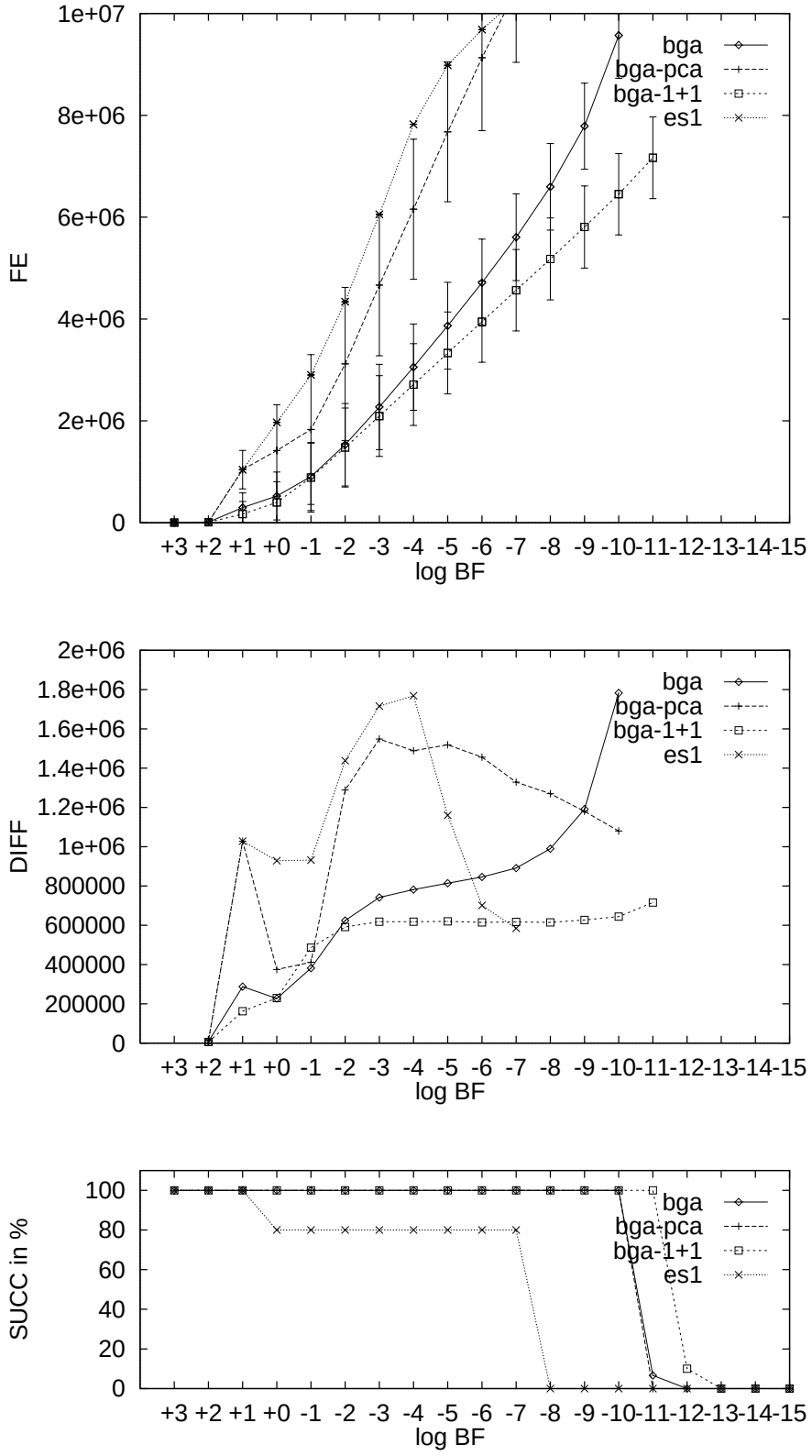

Abbildung 6.30: **Rosenbrocks Funktion** (F4, n=30): Konvergenzverhalten von BGA-Varianten und Evolutionsstrategie es1.

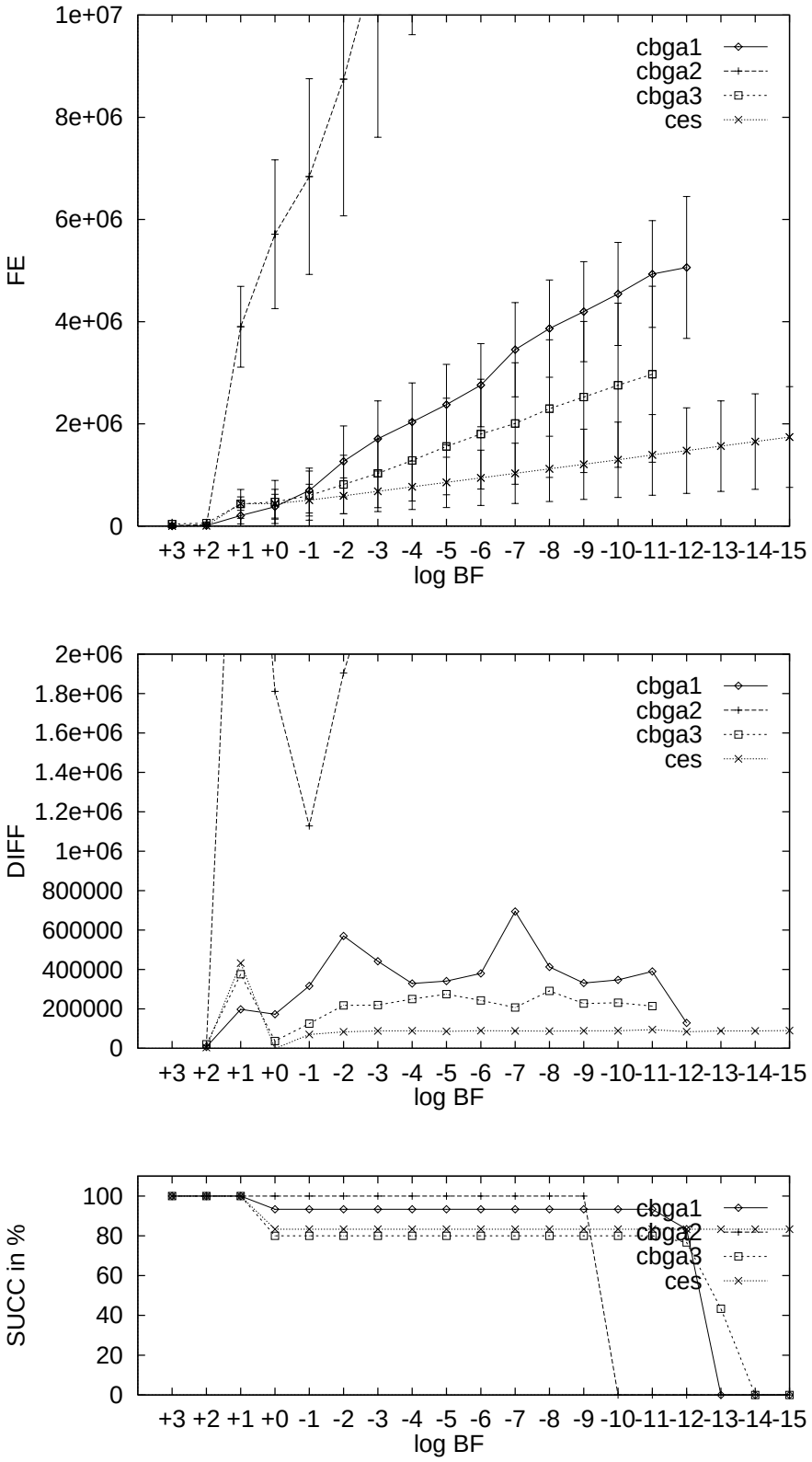

Abbildung 6.31: **Rosenbrocks Funktion** (F4, n=30): Konvergenzverhalten von Wettbewerbsmodellen.

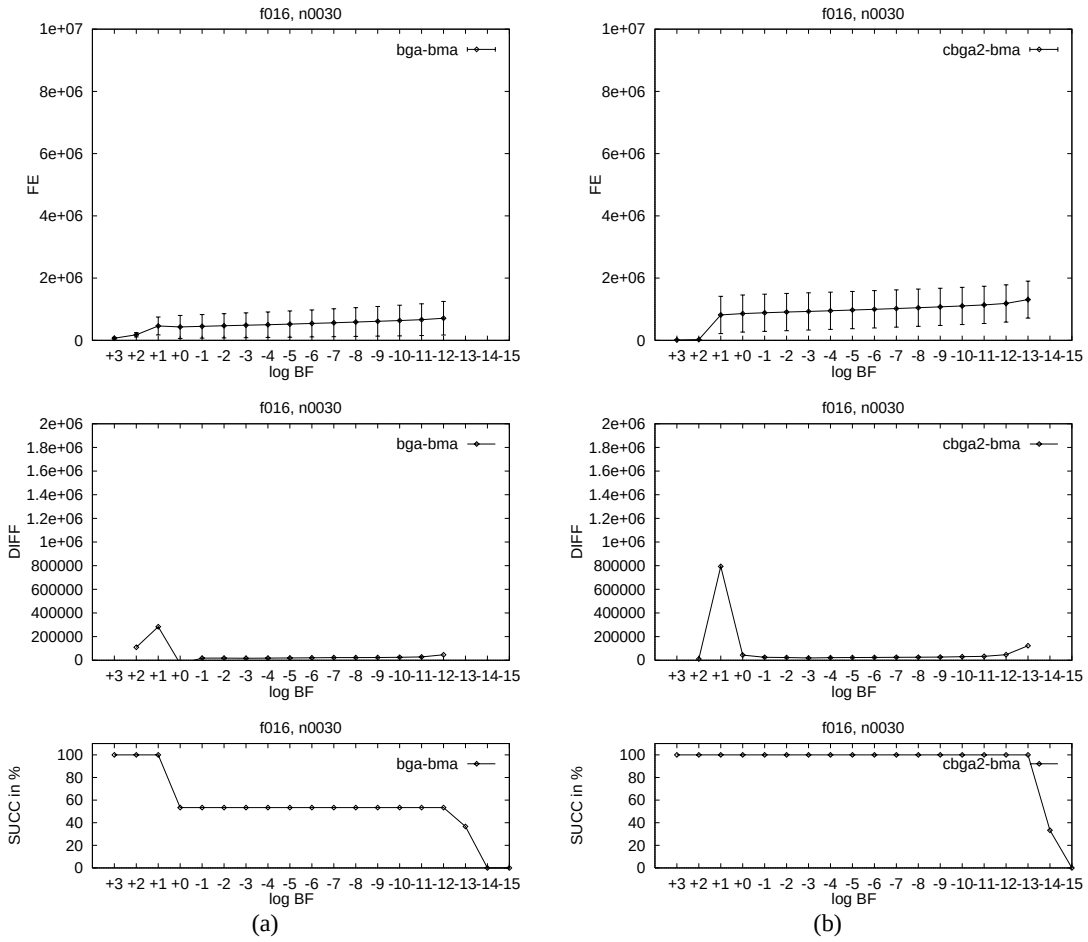

Abbildung 6.32: **Rosenbrocks Funktion** (F4, n=30): Konvergenzverhalten von BGA mit Mehrachsen-BGA-Mutation (bga-bma) und cBGA mit Mehrachsen-BGA-Mutation (cbga2-bma).

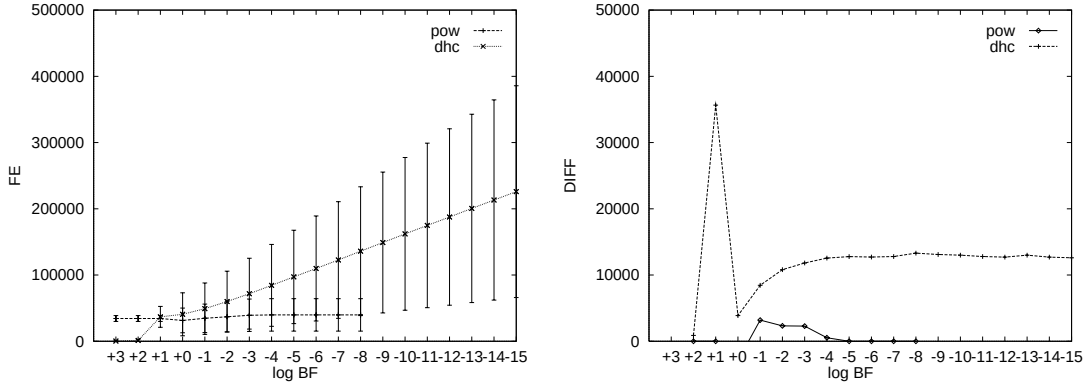

Abbildung 6.33: **Rosenbrocks Funktion** (F4, n=30): Konvergenzverhalten des Powell-Verfahrens und des Dynamic-Hillclimbing-Verfahrens.

Ackleys Funktion (F5)

Das Konvergenzverhalten der Suchverfahren angewandt auf Ackleys Funktion ist in den Abbildungen 6.34 bis 6.37 dargestellt. An dieser Funktion zeigen sich die Schwierigkeiten, die lokale Suchverfahren mit einfachen multimodalen Funktionen haben. Obwohl die Makrostruktur von Ackleys Funktion unimodal und kugelsymmetrisch ist, gelingt es keinem lokalen Suchverfahren, in die Nähe des Optimums zu gelangen. Alle werden Opfer der überlagerten Mikrostruktur. Wie in Tabelle 6.4 zu sehen ist, terminieren alle lokalen Suchverfahren mit Fitnesswerten zwischen 8 und 20.

ALG	$\bar{f}^*$	f^*_{sd}	$f^*_{\min}$	$f^*_{\max}$
qn	19.4	0.1	19.1	19.6
pow	18.1	0.4	17.2	18.8
dhs	8.5	3.9	2.8	17.2
sw	21.2	0.1	21.0	21.4
dhc	8.5	3.9	2.8	17.2
es1	1.5E-14	7.1E-16	8.0E-15	1.5E-14
es2	0.3	0.5	1.5E-14	1.4
ces	1.9	3.5	8.0E-15	13.7
ga	14.0	0.9	12.6	15.6
bga	1.4E-7	2.5E-8	1.0E-7	2.2E-7
bga-pca	13.9	1.3	12.5	16.0
bga-1+1	2.0E-8	7.9E-10	1.8E-8	2.2E-8
cbga1	1.4E-8	1.2E-9	1.1E-8	1.8E-8
cbga2	9.0E-15	0.0	9.0E-15	9.0E-15
cbga3	1.4	3.1	1.15E-14	9.7

Tabelle 6.4: Erreichte Fitnesswerte der Suchverfahren bei Ackleys Funktion (F5, n=100).

Die beiden ES-Varianten liefern recht unterschiedliche Resultate. Die Evolutionsstrategie es1, die sich durch die überlagerten Schwingungen kaum beeinflussen läßt, skaliert linear mit einem DIFF-Wert von 9400, der ungefähr doppelt so groß ist wie beim Kugelmodell. Mit Standardabweichungen, die weniger als 1/20 der Mittelwerte betragen, ist ihr Konvergenzverhalten äußerst robust. Ganz anders ist das Verhalten der Evolutionsstrategie es2, der es in 30 % der Läufe nicht gelingt, kleinere Fitnesswerte als 0.1 zu erreichen. Mit DIFF-Werten, die zwischen 60000 und 90000 liegen, weist sie zudem eine sehr geringe Konvergenzgeschwindigkeit auf.

Das Wettbewerbsmodell ces führt zwar zu einer Verbesserung der Konvergenzgeschwindigkeit, erreicht aber in nur 80 % der Läufe Fitnesswerte, die kleiner als 1.0 sind. Da auch die Läufe starken Schwankungen unterworfen sind, hat sie gegenüber der einfachen Evolutionsstrategie es1 an Robustheit eingebüßt.

Der Standard-BGA (bga) und das Mutations-Selektions-Verfahren (bga-1+1) terminieren mit Fitnesswerten zwischen 10^{-7} und 10^{-8}. Effizienz und Robustheit der beiden BGA sind in diesem Bereich mit der Evolutionsstrategie es1 vergleichbar. Die Hauptachsentransformation des bga-pca führt zu einer wesentlichen Verschlechterung des Konvergenzver-

haltens. Der bga-pca terminiert mit einer durchschnittlichen Fitness von 13.9. Der cbga1 erreicht gegenüber dem Standard-BGA nur eine geringfügig bessere Konvergenzgeschwindigkeit. Seine DIFF-Werte liegen zwischen 8000 und 14000. Das Wettbewerbsmodell cbga2, das die rekombinationsbasierte Breitensuche mit der mutationsbasierten Richtungssuche kombiniert, nähert sich in allen Läufen bis auf 10^{-14} an das Optimum an. Das Konvergenzverhalten dieses Wettbewerbsmodell zeigt sich mit seinen geringen Standardabweichungen als äußerst stabil. Es konvergiert linear mit einem DIFF-Wert von 28000. Der cBGA mit Strategiemodifikation (cbga3), der bis zum Zielfunktionswert 10^{-13} mit einem durchschnittlichen DIFF-Wert von 9800 linear konvergiert, erweist sich damit als der effizienteste BGA in 80 % der Läufe. Wie in Abbildung 6.37 auf Seite 183 zu sehen ist, sind seine Läufe allerdings hohen Schwankungen unterworfen.

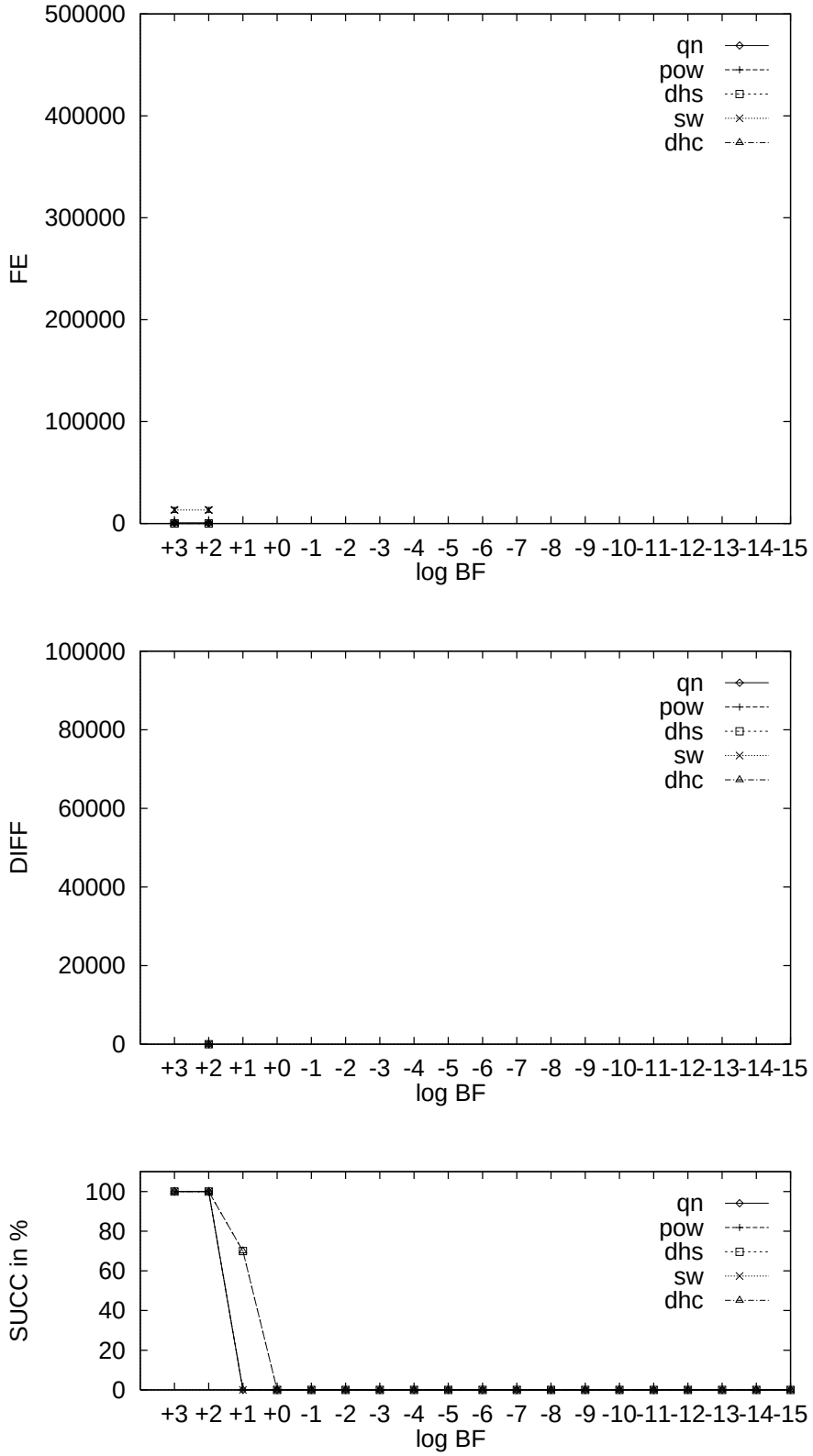

Abbildung 6.34: **Ackleys Funktion** (F5 , n=100): Konvergenzverhalten von lokalen Suchverfahren.

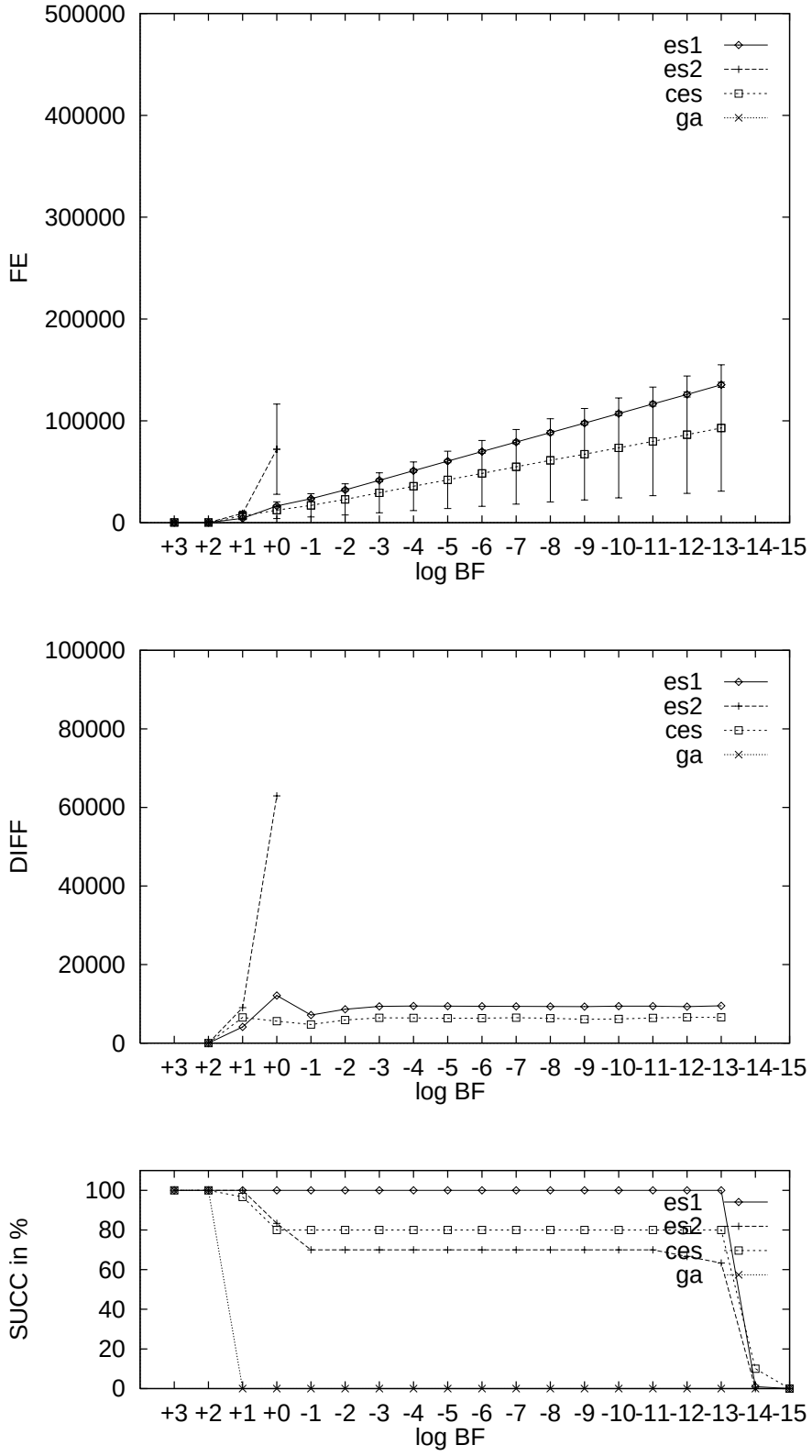

Abbildung 6.35: **Ackleys Funktion** (F5 , n=100): Konvergenzverhalten von Evolutionsstrategien und Genetischem Algorithmus.

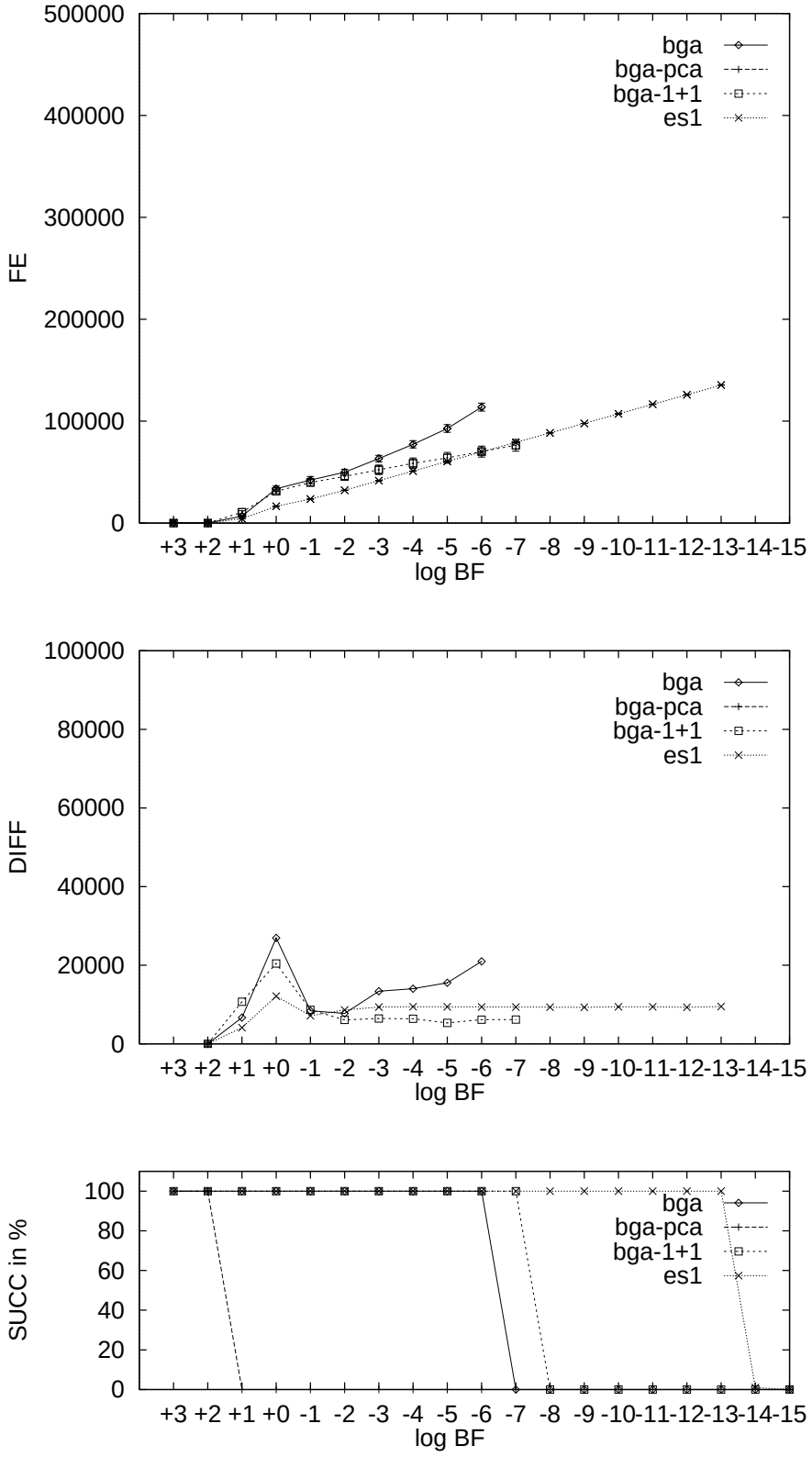

Abbildung 6.36: **Ackleys Funktion** (F5 , n=100): Konvergenzverhalten von BGA-Varianten und Evolutionsstrategie es1.

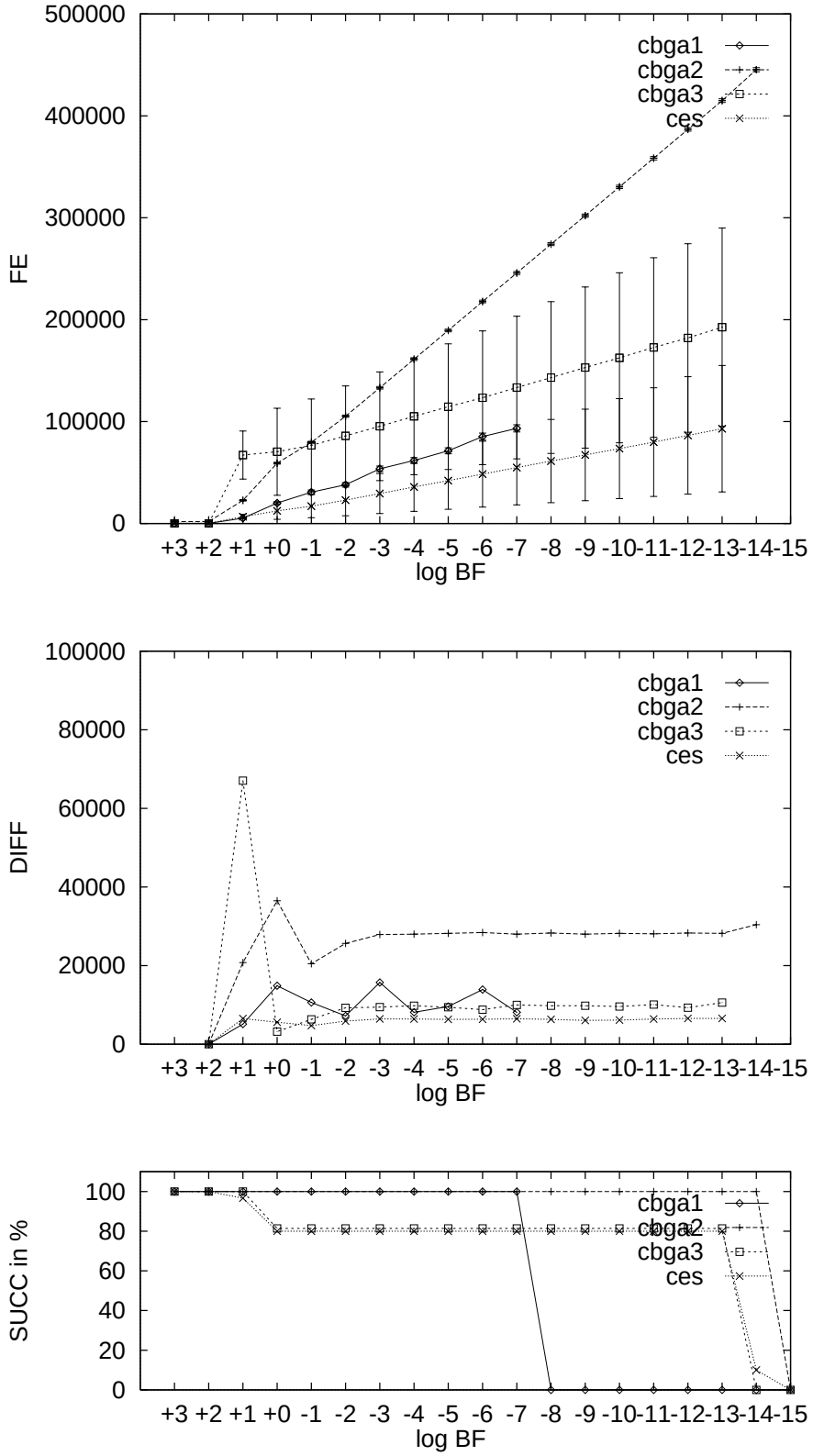

Abbildung 6.37: **Ackleys Funktion** (F5 , n=100): Konvergenzverhalten von Wettbewerbsmodellen.

Flechter und Powells Funktion (F6)

Da die Funktion von Flechter und Powell eine multimodale Funktion mit unregelmäßig angeordneten lokalen Minima ist, rückt hier der Robustheitsaspekt in den Vordergrund. Die Suchverfahren werden ausschließlich bezüglich ihrer Fähigkeit, gute Lösungen zu finden, miteinander verglichen.

Die Robustheit der Suchverfahren bezüglich der Zielfunktionsannäherung ist in Abbildung 6.38 dargestellt. Die Darstellung entspricht der in Abbildung 6.8 eingeführten Robustheitsdarstellung. Wie in Abschnitt 6.5.2 beschrieben, wird aus 100 Simulationsläufen die relative Häufigkeit der erreichten Fitnesswerte bestimmt und zur besseren Darstellung akkumuliert. Zur Berechnung der relativen Häufigkeit wird das Fitnessintervall $[0, 20000]$ in 100 Teilstücke zerlegt, so daß die Genauigkeit, mit der die Fitness angegeben wird, 200.0 beträgt. Die durchschnittlich erreichten Fitnesswerte sind in Tabelle 6.5 eingetragen. Als weiteres Robustheitsmaß enthält die Tabelle die Fläche unter der Kurve der akkumulierten Häufigkeitsverteilung der Fitnesswerte (siehe Gleichung 6.23).

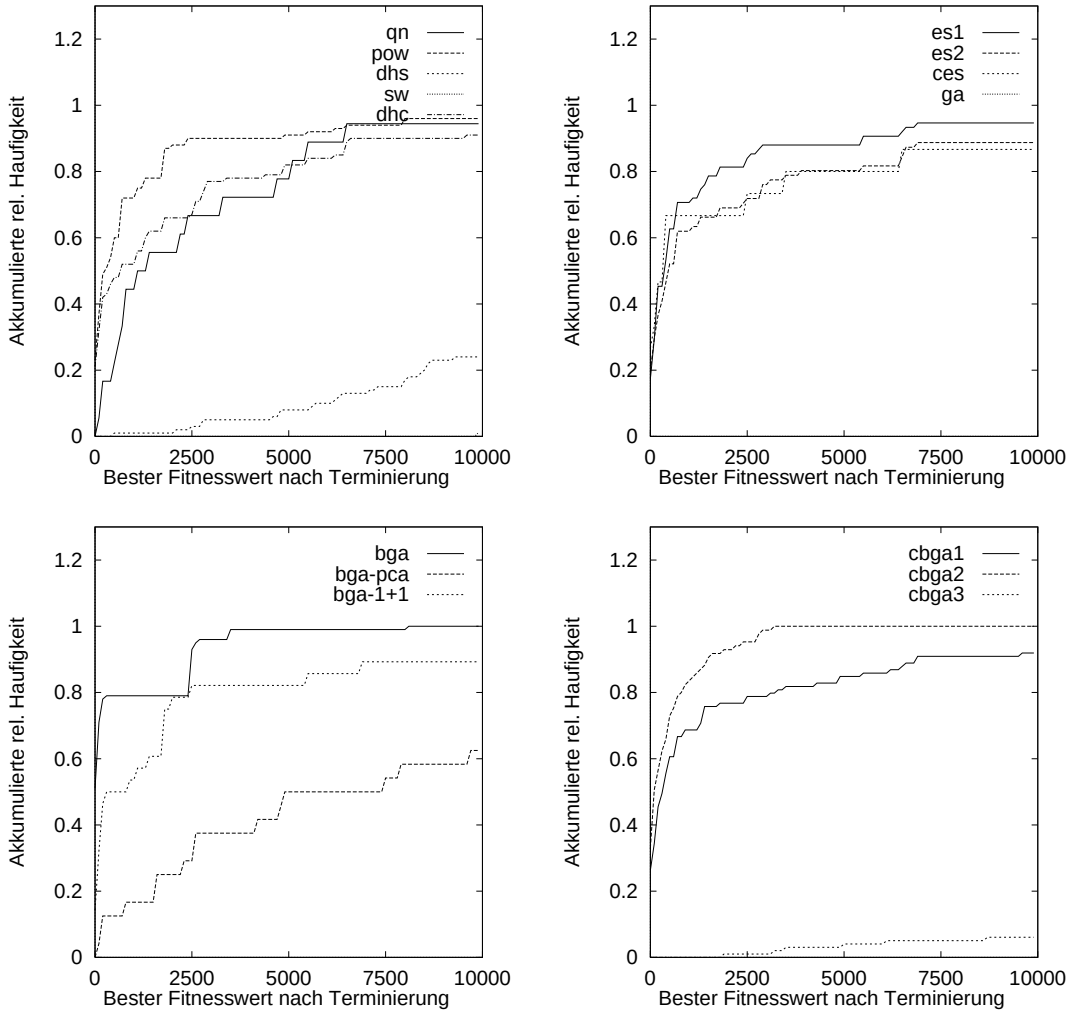

Abbildung 6.38: **Flechter und Powells Funktion** (F6 , n=30):

ALG	$\bar{f}^*$	f^*_{sd}	$f^*_{\min}$	$f^*_{\max}$	$A(c)$
qn	3394.5	5566.9	104.9	24127.1	7488.9
pow	1363.1	2652.5	0.1	12323.1	8780.0
dhs	27090.9	25045.2	554.4	143356.0	954.0
sw	222721.0	196504.0	9972.7	832399.0	5.3
dhc	2717.9	4378.9	1.0	21960.4	7740.0
es1	1626.1	3070.9	0.0	16529.4	8541.3
es2	26711.5	120056.0	0.0	942075.0	7731.0
ces	2707.4	4644.8	0.0	15328.7	7720.0
ga	123632.0	83924.5	23808.9	469050.0	0.0
bga	687.3	1303.9	1.0	8146.2	9388.0
bga-pca	12057.4	13602.7	191.8	54487.1	4150.0
bga-1+1	3066.6	5883.3	0.0	24127.1	7928.6
cbga1	2246.1	4284.6	0.0	24127.1	8183.8
cbga2	551.5	761.2	5.2	3269.9	9495.3
cbga3	243211.0	138266.0	1985.4	516928.0	323.2

Tabelle 6.5: Mittlere erreichte Zielfunktionswerte der Suchverfahren bei der Funktion von Flechter und Powell (F6, n=30). $\bar{f}^*$ und f^*_{sd} geben den Mittelwert und die Standardabweichung von 100 Simulationsläufen an. $f^*_{\min}$ und $f^*_{\max}$ bezeichnen den minimalen beziehungsweise maximalen Zielfunktionswert, den das jeweilige Suchverfahren in den 100 Läufen erzielte. $A(c)$ ist das in Gleichung 6.23 eingeführte Robustheitsmaß mit $c = 10000$. Je näher dieser Wert an c liegt, umso robuster ist das Suchverfahren in Bezug auf das Erreichen guter Zielfunktionswerte.

Bezüglich des Robustheitsmaßes $A(10000)$, das die Fläche unter den in Abbildung 6.38 dargestellten akkumulierten Häufigkeitsverteilungen der erreichten Zielfunktionswerte im Intervall 0 bis 10000 angibt, stellt sich das Powell-Verfahren (pow) als das robusteste der lokalen Suchverfahren heraus. Das Quasi-Newton-Verfahren (qn) und das Dynamic-Hillclimbing-Verfahren (dhc) zeigen ein geringfügig schlechteres Konvergenzverhalten. Das Verfahren von Solis und Wets (sw), dessen bester erreichter Zielfunktionswert mit 9972.7 nur knapp in das betrachtete Intervall fällt, erzielt bei dieser multimodalen Zielfunktion die schlechtesten Resultate aller hier betrachteten Suchverfahren.

Das Verhalten der einfachen Evolutionären Algorithmen ist mit dem der iterierten lokalen Suchverfahren vergleichbar. Während die Resultate der Evolutionsstrategie es1 etwas besser als die des Powell-Verfahrens sind, zeigen die Evolutionsstrategie es2 und das Wettbewerbsmodell ces ein deutlich schlechteres Verhalten. Das Wettbewerbsmodell wird offenbar stark durch die Evolutionsstrategie es2 beeinflußt. Da der beste Fitnesswert, den der Genetische Algorithmus (ga) erreicht, weit über 10000 liegt, sind dessen relative Häufigkeiten in dem betrachteten Intervall Null.

Der Vorteil populationsbasierter Suchverfahren wird am Verhalten der BGA deutlich. Der Standard-BGA bga erzielt wesentlich bessere Resultate als das aus einem Individuum bestehende Mutations-Selektions-Verfahren bga-1+1. Die Hauptachsentransformation führt bei dieser Funktion zu einer erheblichen Verschlechterung. Wie erwartet erzielt das Wettbewerbsmodell cbga2 aufgrund der großen rekombinationsbasierten Population ein sehr gutes Ergebnis. Die beiden anderen cBGA laufen dagegen sehr schnell in ein na-

hegelegenes lokales Minimum. Der cbga3 ist für eine derartige multimodale Zielfunktion ungeeignet, da er die Schrittweitenparameter zu schnell an das Attraktorgebiet des nächstgelegenen lokalen Minimums anpaßt. Diese Funktion macht die Problematik, die die Modifikation von Strategien während der Suche mit sich bringt, deutlich.

6.6.2 Skalierung

Das Skalierungsverhalten der Suchverfahren wird an den unimodalen Funktionen Kugelmodell, gewichtetes Kugelmodell und Schwefels Funktion untersucht. Als Skalierungsmaße werden die in Abschnitt 6.5.3 eingeführten Modelle und ihre Koeffizienten verwendet.

Da das Skalierungsverhalten eines Algorithmus für den gesamten Dimensionsbereich nur selten durch ein einziges Modell charakterisiert werden kann, wird im folgenden zwischen einem niedrigen und einem hohen Dimensionsbereich unterschieden. Die Grundlage für die Berechnung der Skalierungsmodelle im niedrigen Dimensionsbereich bilden die Simulationsergebnisse für die Dimensionen $10, 30, 50$ und 100. Der hohe Dimensionsbereich wird durch die Dimensionen $100, 200, 500$ und 1000 repräsentiert. Für jeden Bereich wird ein eigenes Modell berechnet. In den folgenden Tabellen enthält die erste Zeile jedes Suchverfahrens das Modell für den niedrigen Dimensionsbereich und die zweite Zeile das Modell für den hohen Dimensionsbereich.

Das Skalierungsverhalten wird bezüglich der beiden Zielfunktionswerte 10^{-6} und 10^{-12} untersucht. Falls ein Suchverfahren nicht in der Lage ist, den jeweiligen Zielfunktionswert in allen Dimension zu erreichen, wird dies in der Tabelle vermerkt. n_f gibt die niedrigste Dimension an, für die weniger als $80\,\%$ der Läufe erfolgreich sind. Die in Abschnitt 6.5.3 eingeführten Modelle sind im folgenden noch einmal angegeben:

$$\begin{aligned}
\text{M1} &: f_1(x) = a + b \cdot x^{0.75} \\
\text{M2} &: f_2(x) = a + b \cdot x \\
\text{M3} &: f_3(x) = a + b \cdot x \ln x \\
\text{M4} &: f_4(x) = a + b \cdot x^2 \\
\text{M5} &: f_5(x) = a + b \cdot x^{2.5} \\
\text{M6} &: f_6(x) = a + b \cdot x^3
\end{aligned} \tag{6.27}$$

Kugelmodell (F1)

In Tabelle 6.6 ist das Skalierungsverhalten der Suchverfahren für das Kugelmodell (F1) eingetragen.

Die Mehrzahl der Suchverfahren skaliert bei dem Kugelmodell annähernd linear. Für den Fitnesswert 10^{-6} weisen das Quasi-Newton-Verfahren (qn) und das Powell-Verfahren (pow) das mit Abstand beste Skalierungsverhalten auf. Beide skalieren in beiden Dimensionsbereichen linear mit einer Steigung unter 10.0. Für den Fitnesswert 10^{-12} scheiterten jedoch beide Verfahren an den Dimensionen 30 beziehungsweise 50. Das Dynamic-Hillclimbing-Verfahren (dhc) ist das einzige lokale Suchverfahren, das bereits im niedrigen Dimensionsbereich quadratisch skaliert.

Kugelmodell (F1)											
ALG	n	**BF = 10^{-6}**					**BF = 10^{-12}**				
		M	a	b	n_f	χ^2	**M**	a	b	n_f	χ^2
qn	L	**2**	8.3	**6.3**		0.058				30	
	H	**2**	49.1	**5.9**		0.116					
pow	L	**2**	2.5	**9.6**		0.004				50	
	H	**2**	4.7	**9.6**		0.073					
dhs	L	**1**	-217247.0	**39065.8**		0.161	**1**	-234556.0	**42174.6**		0.151
	H	**4**	982824.0	**2.8**		2.423	**4**	979468.0	**3.3**		0.194
sw	L	**2**	-155.2	**92.1**		0.532	**2**	-227.3	**155.0**		0.253
	H	**2**	-1437.9	**103.9**		3.498	**2**	-1506.7	**166.3**		3.064
dhc	L	**4**	350.1	**3.3**		0.039				100	
	H				200						
es1	L	**2**	2389.6	**422.9**		1.762	**2**	4935.7	**701.8**		3.034
	H	**2**	7882.5	**361.1**		2.594	**2**	17390.4	**565.7**		2.975
es2	L	**4**	5380.5	**21.2**		0.648	**4**	9747.8	**38.7**		1.501
	H				500					500	
ces	L	**2**	2424.0	**386.3**		0.766	**2**	4516.6	**650.3**		1.474
	H										
ga	L				30					10	
	H										
bga	L	**2**	-1862.2	**533.2**		0.602	**2**	-2683.7	**986.7**		0.514
	H	**2**	-13086.3	**629.0**		4.504	**2**	-20979.4	**1125.1**		11.075
bga-pca	L	**3**	1072.1	**113.8**		0.130	**3**	2757.1	**201.4**		0.063
	H										
bga-1+1	L	**2**	-953.9	**360.7**		0.137	**2**	-1277.0	**543.4**		0.152
	H	**2**	-8200.0	**433.0**		0.419	**2**	-8926.4	**627.2**		0.500
cbga1	L	**2**	2418.3	**442.4**		1.863	**2**	7969.1	**752.7**		0.093
	H	**2**	384.4	**457.6**		1.615	**2**	10013.6	**715.3**		0.532
cbga2	L	**1**	4650.2	**3832.0**		1.803	**1**	9634.6	**6370.5**		5.605
	H	**1**	-12084.0	**4230.5**		27.093	**2**	74058.6	**1369.3**		6.164
cbga3	L	**2**	15091.5	**726.5**		0.056	**2**	44348.9	**397.5**		0.251
	H	**2**	14895.0	**1060.1**		0.452	**2**	59503.0	**646.5**		0.094

Tabelle 6.6: Skalierung der Suchverfahren beim Kugelmodell (F1). n gibt den Dimensionsbereich an, der dem Skalierungsmodell zugrunde liegt ($L : n = 10 - 100$, $H : n = 100 - 1000$). M bezeichnet das Modell mit dem geringsten χ^2-Fehler. a und b sind die Koeffizienten des Modells. n_f kennzeichnet die niedrigste Dimension, für die weniger als 80% der Läufe erfolgreich waren.

Die Mehrzahl der EA skaliert linear. Eine Ausnahme bildet die Evolutionsstrategie es2, die quadratisch skaliert [6]. Die Anpassung der n Strategieparameter wird mit zunehmender Dimension aufwendiger. Der Genetische Algorithmus (ga) scheitert für den Zielfunktionswert 10^{-6} bereits an der Dimension 30.

Das Skalierungverhalten der Evolutionären Algorithmen kann durch die Wettbewerbsmodelle in den meisten Fällen verbessert werden. Bei den Wettbewerbsmodellen ces und cbga1 wird eine Reduzierung der Koeffizienten erreicht. Das superlineare Skalierungsverhalten des cbga2 ist auf dessen große Populationsgröße zurückzuführen, die erst in den hohen Dimensionen ausgenutzt wird.

In [MSV93b] wurde für den bga-1+1 eine Skalierung von $O(n \ln n)$ theoretisch abgeleitet. In Tabelle 6.6 ist eine Skalierung von $O(n)$ ermittelt worden. Der Grund für diese Differenz liegt darin, daß in dieser Arbeit der Startpunkt zufällig gewählt wurde, während in [MSV93b] der Startpunkt fest am Rande angenommen wurde. Damit wächst der Abstand δ_0 vom Optimum mit $\sqrt(n)$. Aus Satz 12 ersieht man, daß der Abstand δ_0 wie $\ln \delta_0$ in die Zahl der benötigten Funktionsauswertungen eingeht. Daher sind die Ergebnisse dieser Arbeit im Einklang mit der Theorie.

Gewichtetes Kugelmodell (F2)

Tabelle 6.7 enthält die Ergebnisse der Skalierungsuntersuchung für das gewichtete Kugelmodell (F2). Die unterschiedliche Gewichtung der Objektvariablen führt bei vielen Suchverfahren nicht nur zu einem veränderten Konvergenzverhalten, sondern ebenfalls zu einem anderen Skalierungsverhalten.

Während das Quasi-Newton-Verfahren (qn) und das Verfahren von Solis und Wets (sw) wesentlich schlechter als beim Kugelmodell skalieren, wird das Powell-Verfahren(pow) von den unterschiedlichen Gewichtungen nicht beeinflußt. Die Skalierungsmodelle des Powell-Verfahrens für Kugelmodell und gewichtetes Kugelmodell stimmen exakt überein.

Die Evolutionsstrategie es1, die beim Kugelmodell linear skaliert, weist hier ein quadratisches Skalierungsverhalten auf. Das Skalierungsverhalten der Evolutionsstrategie es2 entspricht exakt dem des Kugelmodells. Da die Anpassung der n Strategieparameter sowohl im Kugelmodell als auch im gewichteten Kugelmodell erfolgen muß, skaliert sie bei beiden Funktionen quadratisch. Das Wettbewerbsmodell ces führt zu einer erheblichen Verbesserung des Skalierungsverhaltens. In beiden Dimensionsbereichen skaliert das Wettbewerbsmodell wie $O(n \ln n)$.

Die BGA weisen aufgrund ihrer robusten Mutationsverteilung ein ähnliches Skalierungsverhalten wie beim Kugelmodell auf. Sie skalieren für alle Dimensionsbereiche und Fitnesswerte annähernd linear. Eine Ausnahme bildet der BGA mit Hauptachsentransformation (bga-pca), der ein quadratisches Skalierungsverhalten zeigt.

Schwefels Funktion (F3)

Tabelle 6.8 enthält Ergebnisse der Skalierungsuntersuchung für Schwefels Funktion (F3). Das Skalierungsverhalten von Schwefels Funktion wird aufgrund ihres Schwierigkeitgrads

[6] Bei Untersuchungen von Schwefel zeigte die es2 bis zur Dimension 435 ein lineares Verhalten ([Sch75]). Der Grund für die unterschiedlichen Skalierungsverhalten liegt vermutlich in der unterschiedlichen Parametrisierung dieser Evolutionsstrategie.

Gewichtetes Kugelmodell (F2)											
ALG	n	BF = 10^{-6}					BF = 10^{-12}				
		M	a	b	n_f	χ^2	M	a	b	n_f	χ^2
qn	L	**2**	-19944.4	**2634.2**		0.592				10	
	H				1000						
pow	L	**2**	2.9	**9.6**		0.002				30	
	H	**2**	2.0	**9.6**		0.061					
dhs	L	**1**	-234805.0	**42118.6**		0.017	**3**	-88750.3	**3966.4**		0.005
	H	**4**	1015360.0	**3.6**		0.061	**4**	1529580.0	**4.5**		0.932
sw	L	**4**	20.1	**21.1**		0.049	**4**	-53.1	**39.9**		0.021
	H	**4**	-10170.8	**21.9**		0.487	**4**	-10596.9	**40.9**		0.492
dhc	L	**4**	337.1	**1.4**	100	0.653				50	
	H										
es1	L	**4**	-3512.5	**111.1**		1.066	**4**	-5412.6	**231.1**		0.680
	H				500					500	
es2	L	**4**	5819.1	**24.9**		1.556	**4**	10238.3	**41.3**		2.763
	H				500					200	
ces	L	**3**	646.3	**266.9**		0.421	**3**	1646.1	**431.1**		0.584
	H				1000					500	
ga	L				30					10	
	H										
bga	L	**3**	1176.3	**139.1**		1.888	**3**	2456.9	**266.0**		0.535
	H	**3**	11622.9	**117.6**		3.338				500	
bga-pca	L	**4**	3344.7	**13.4**		5.503	**3**	-440.5	**366.0**		13.785
	H										
bga-1+1	L	**2**	-949.1	**393.9**		0.770	**2**	-1032.7	**560.2**		0.609
	H	**2**	-13913.8	**536.9**		0.445	**2**	-17759.3	**745.9**		0.152
cbga1	L	**2**	1964.9	**555.6**		0.241	**2**	6967.4	**821.1**		0.231
	H	**2**	1309.5	**552.5**		0.273	**2**	4482.5	**834.4**		0.340
cbga2	L	**1**	1689.4	**4645.3**		0.301	**1**	6960.2	**7177.8**		1.743
	H	**3**	60792.3	**186.3**		10.707	**3**	103281.0	**277.4**		8.667
cbga3	L	**2**	14726.8	**1023.2**		0.014	**2**	18324.7	**1373.8**		0.571
	H				500					500	

Tabelle 6.7: Skalierung der Suchverfahren beim gewichteten Kugelmodell (F2). Die Symbole entsprechen den in Tabelle 6.6 verwendeten.

und der damit verbundenen hohen Ausführungszeiten nur für den niedrigen Dimensionsbereich untersucht. Aufgrund des mit steigender Dimension zunehmenden Schwierigkeitsgrades skalieren hier alle Suchverfahren mindestens quadratisch. Die lokalen Suchverfahren zeigen bei dieser Funktion besondere Schwächen. Während das Powell-Verfahren und das Dynamic-Hillclimbing-Verfahren bei Dimension 100 nicht immer die vorgebene Genauigkeit erreichen, skalieren das Quasi-Newton-Verfahren und das Verfahren von Solis und Wets kubisch. Das beste Skalierungsverhalten der iterierten lokalen Suchverfahren weist das Downhill-Simplex-Verfahren mit $O(n^{2.5})$ auf. Der Grund für das schlechte Verhalten der lokalen Suchverfahren liegt in deren numerischer Instabilität bei schlechter Kondition dieses quadratischen Problems.

Gegenüber den lokalen Suchverfahren zeigen die Evolutionären Algorithmen bei dieser Funktion ein robusteres Suchverhalten. Bis auf die Evolutionsstrategie es2 und das Wettbewerbsmodells cbga2 sind alle in der Lage, bis zur Dimension 100 den Fitnesswert von 10^{-6} zu erreichen. Besonders eindrucksvoll sind bei dieser Funktion die Verbesserungen, die durch die Wettbewerbsmodelle erzielt werden. Obwohl die Evolutionsstrategie es1 wie

Schwefels Funktion (F3)										
ALG	BF = 10^{-6}					BF = 10^{-12}				
	M	a	b	n_f	χ^2	M	a	b	n_f	χ^2
qn	6	19943.9	**4.4**		0.595				10	
pow				100					10	
dhs	**5**	-1437.5	**14.6**		0.093	**5**	-878.4	**14.5**		0.089
sw	6	-1624.6	**56.8**		0.438	6	1536.3	**92.0**		0.230
dhc				100					100	
es1	**5**	4494.4	**35.1**		2.558	**5**	6524.5	**61.3**		7.948
es2				50					50	
ces	4	-2113.6	**192.6**		0.168	4	-4048.8	**309.6**		0.259
ga				30					10	
bga	4	-897.6	**965.4**		0.409				50	
bga-pca	4	2742.6	**318.7**		0.242				30	
bga-1+1	4	-19425.5	**816.7**		2.146				10	
cbga1	4	8995.1	**357.4**		0.139	4	12828.8	**613.5**		1.575
cbga2				100					30	
cbga3	4	53335.7	**220.4**		0.723	4	58490.2	**360.2**		0.456

Tabelle 6.8: Skalierung der Suchverfahren bei Schwefels Funktion (F3).

$O(n^{2.5})$ skaliert und die Evolutionsstrategie es2 schon ab Dimension 50 divergiert, skaliert das Wettbewerbsmodell ces, das beide Evolutionsstrategien kombiniert, quadratisch und liefert sogar die beste Skalierung aller Suchverfahren.

Die BGA skalieren durchweg quadratisch. Ihr Verhalten unterscheidet sich lediglich im Wert ihrer Koeffizienten. Wie bei den Evolutionsstrategien lassen sich auch bei den BGA durch die Wettbewerbsmodelle signifikante Verbesserung der Skalierungseigenschaften erzielen. Besonders hervorzuheben ist das Verhalten des Wettbewerbsmodells cbga3, das die Schrittweitenparameter während der Suche anpaßt. Er verhält sich annähernd so effizient wie das Wettbewerbsmodell ces.

Bei dieser Funktion wird gezeigt, wie sich die Effizienz des Wettbewerbsmodells cbga1 durch die Hinzunahme weiterer Populationen steigern läßt. In Tabelle 6.9 ist das Skalierungsverhalten des cBGA in Abhängigkeit der Anzahl der Populationen dargestellt. bga bezeichnet den Standard-BGA mit einer Population und cbga1 das Wettbewerbsmodell mit vier Populationen. Die cBGA cbga1-g2 und cbga1-g8, die dem Wettbewerbsmodell mit 2 bzw. 8 Populationen entsprechen, werden wie der cbga1 von der cBGA-Klasse cBGA aus Spezifikation 24 abgeleitet. Mit steigender Populationsanzahl kann der Präzisionsparameter der BGA-Mutation gesenkt werden, wodurch bei geeignetem Schrittweitenparameter die Fortschrittsgeschwindigkeit erhöht wird.

Mit zunehmender Anzahl von konkurrierenden Populationen kann der Koeffizient des quadratischen Terms erheblich reduziert werden. Der Wert des Koeffizienten des cBGA mit 8 Populationen beträgt nur ein Viertel des Koeffizienten des Standard-BGA. In Abbildung 6.39 sind die Skalierungsmodelle und die zugrunde liegenden Daten für den Fitnesswert 10^{-6} dargestellt.

Zusammenfassende Bewertung der Skalierungsergebnisse

Für die Berechnung der Skalierung wird die Anzahl der durchschnittlichen Funktionsauswertungen verwendet, die benötigt werden, um den minimalen Funktionswert auf $\epsilon = 10^{-6}$ und $\epsilon = 10^{-12}$ genau zu erreichen. Die Koeffizienten der Skalierungsmodelle für

Schwefels Funktion (F3)										
ALG	BF = 10^{-6}					BF = 10^{-12}				
	M	a	b	n_f	χ^2	M	a	b	n_f	χ^2
bga	4	-897.6	**965.4**		0.409				50	
cbga1-g2	4	-282.8	**616.9**		0.456	4	6453.0	**1010.2**	100	0.163
cbga1	4	8995.1	**357.4**		0.139	4	12828.8	**613.5**		1.575
cbga1-g8	4	10882.4	**251.6**		3.628	4	17932.6	**423.5**		2.735

Tabelle 6.9: Skalierungsverhalten des cBGA für verschiedenene Populationsanzahlen.

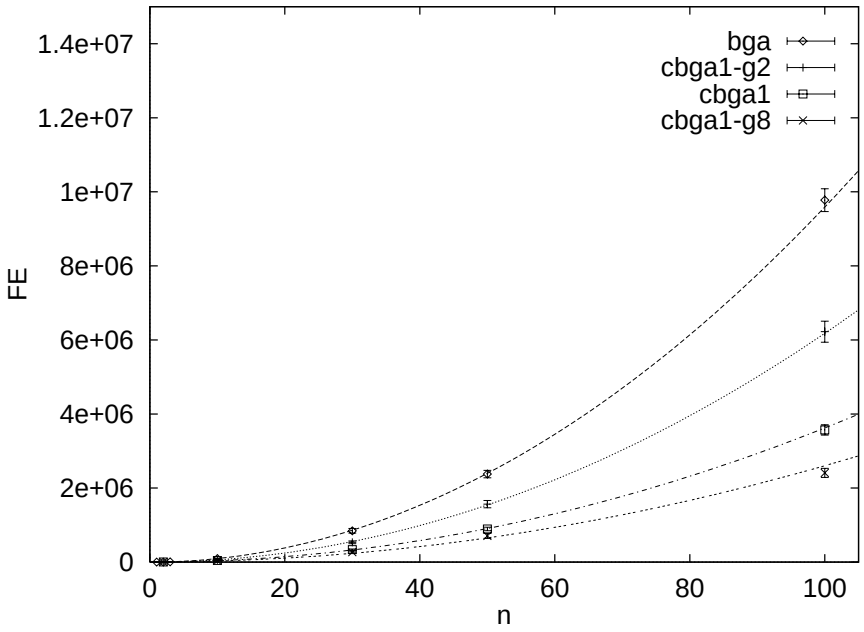

Abbildung 6.39: Skalierungsverhalten des cBGA für verschiedenene Populationsanzahlen bei Schwefels Funktion (F3, BF= 10^{-6}).

beide Genauigkeiten hängen im allgemeinen voneinander ab. Wir zeigen dies unter der Voraussetzung der linearen Konvergenz. Aus Satz 12 ergibt sich für die Anzahl der erforderlichen Funktionsauswertungen

$$t(n, 10^{-6}) \approx \Leftrightarrow \frac{1}{\ln a(n)} \ln(\delta_0 \cdot 10^6) \tag{6.28}$$

$$t(n, 10^{-12}) \approx \Leftrightarrow \frac{1}{\ln a(n)} \ln(\delta_0 \cdot 10^{12}) \tag{6.29}$$

wobei $a(n)$ die Ratenkonstante für die Dimension n und δ_0 der Abstand des Startpunkts vom Optimum ist. Im Falle des Kugelmodells liegt der durchschnittliche Fehler δ_0 zwischen 10^3 und 10^4. Unter diesen Umständen ergibt sich als Funktionsauswertungen

$$\frac{t(n, 10^{-6})}{t(n, 10^{-12})} \approx 0.6 \tag{6.30}$$

Für die Skalierung wird nun der Ansatz gemacht

$$t(n, 10^{-6}) \approx a_1 + b_1 f(n) \tag{6.31}$$

$$t(n, 10^{-12}) \approx a_2 + b_2 f(n) \tag{6.32}$$

wobei $f(n)$ die Basisfunktion des Modells ist.

Für das Kugelmodell ergibt sich daher

$$\frac{a_1 + b_1 f(n)}{a_2 + b_2 f(n)} \approx 0.6 \tag{6.33}$$

Mit $f(n) \to \infty$, erhält man

$$\frac{b_1}{b_2} \approx 0.6 \tag{6.34}$$

Aus den Tabellen 6.6 und 6.7 kann man entnehmen, daß obige Relation relativ gut approximiert wird. Dies ist eine Bestätigung, daß die numerisch ermittelte Skalierung sinnvoll ist.

Während das einfache Mutations-Selektions-Verfahren bga-1+1 sowohl für das Kugelmodell als auch für das gewichtete Kugelmodell linear skaliert, skalieren die Evolutionsstrategien es1 und es2 für das gewichtete Kugelmodell quadratisch. Interessant ist, daß die Evolutionsstrategie es2 mit n Strategieparametern schon für das Kugelmodell quadratisch skaliert [7]. Dies deutet an, daß die Anpassung von n Strategieparametern eine Mindestanzahl von Funktionsauswertungen benötigt. Das Wettbewerbsmodell ces, das beide Evolutionsstrategien miteinander kombiniert, erzielt aufgrund von Synergieeffekten, die sich durch die Kombinbination der beiden Evolutionsstrategien einstellen, mit $O(n \ln(n))$ ein erheblich besseres Skalierungsverhalten.

Für Schwefels Funktion skalieren alle Verfahren zumindest quadratisch. Der Grund liegt darin, daß der Schwierigkeitsgrad dieser Funktion mit zunehmender Dimension quadratisch wächst. Wie in Abschnitt 6.2.3 angegeben, ist die Matrix-Notation dieser Funktion

$$f(\vec{x}) = \vec{x}^T \mathbf{A} \vec{x} \tag{6.35}$$

wobei A eine symmetrische und positiv definite Matrix ist. Die Konditionszahl von A, die ein Maß für die numerische Schwierigkeit des Problems ist, wächst mit zunehmender Dimension quadratisch.

Besonders hervorzuheben ist, daß die iterierten lokalen Suchverfahren sogar nur kubisch skalieren. Dies ist ein Indiz dafür, daß bei schwierigen Optimierungsproblemen, die zu numerischen Instabilitäten führen, die lokalen Suchverfahren besonders anfällig sind.

Anzumerken ist, daß die Evolutionsstrategie es2 mit n Strategieparametern bei dieser Funktion ab Dimension $n = 50$ in allen Läufen divergiert, während die einfache Evolutionsstrategie es1 immer konvergiert. Kursawe erzielte mit der rein panmiktischen Rekombination $\mathbb{R}_{DI\_}$ ähnliche Resultate [Kur95].

Bei Schwefels Funktion ist das Wettbewerbsmodell ces besonders effektiv. Während die Evolutionsstrategie es1 mit $O(n^{2.5})$ skaliert und die Evolutionsstrategie es2 ab $n = 50$ nicht konvergiert, skaliert das Wettbewerbsmodell, das beide Evolutionsstrategien kombiniert, mit $O(n^2)$ und ist insgesamt sogar das effektivste Suchverfahren. Dieses Beispiel verdeutlicht, daß durch das Wettbewerbsmodell sehr starke Synergieeffekte möglich sind.

[7] Zu beachten ist, daß sich die hier eingesetzte Rekombination $\mathbb{R}^{\mathrm{ES}}_{di\_}$ der es2 in der Rekombination der Strategieparameter von der empfohlenen Standardrekombination $\mathbb{R}^{\mathrm{ES}}_{dI\_}$ unterscheidet.

Kapitel 7

Mustererkennung als reale Anwendung

7.1 Grundlagen der Mustererkennung

Die Mustererkennung im weiteren Sinne befaßt sich mit den mathematisch-technischen Aspekten der automatischen Verarbeitung und Auswertung von Mustern. Ein Teilbereich der Mustererkennung ist neben der Analyse komplexer Muster die Klassifikation einfacher Muster. Zu den wichtigsten Anwendungen der Mustererkennung zählen die Schriftzeichen- und die Spracherkennung. Ein neues immer wichtiger werdendes Anwendungsfeld der Mustererkennung ist das sogenannte Scoring, also das Bewerten von Klienten aufgrund ihrer Profile.

In diesem Kapitel wird das Problem der Musterkennung auf ein Problem der kontinuierlichen Parameteroptimierung abgebildet, das durch die in dieser Arbeit vorgestellten Suchverfahren gelöst werden kann. An einem praktischen Problem der Schriftzeichenerkennung (engl.: *optical character recognition, OCR*) wird die Stärke des Wettbewerbsmodells zur Lösung einer hochdimensionalen Fitnessfunktion demonstriert. Eine ausführliche Behandlung der Schriftzeichenerkennung ist in [Nie83] zu finden.

Bei der Schriftzeichenerkennung wird jedes Muster als Ganzes betrachtet. Das Klassifikationsproblem besteht darin, ein Muster f aus dem Problemkreis Ω genau einer Klasse aus einer endlichen Menge zuzuordnen. Falls die Zahl der möglichen Klassen κ ist, kann das Klassifikationsproblem als die Abbildung

$$f \to k \in \{0, \ldots, \kappa\} \tag{7.1}$$

der Muster f aus dem Musterraum auf den Index k der zugehörigen Klasse im Entscheidungsraum betrachtet werden. Die Klasse Ω_0 ist dabei für die Zurückweisung von Mustern, die nicht mit der gewünschten Sicherheit einer Klasse zugeordnet werden können, vorgesehen.

Der Problemkreis Ω wird also in $\kappa + 1$ Teilmengen mit folgenden Beziehungen zerlegt:

$$\begin{aligned} \Omega_k &\neq \emptyset & k &= 0, \ldots, \kappa \\ \Omega_k \cap \Omega_l &= \emptyset & l &\neq k \\ \bigcup_{k=0}^{\kappa} \Omega_k &= \Omega \end{aligned} \tag{7.2}$$

Ein Klassifikationssystem besteht, wie in Abbildung 7.1 dargestellt, aus verschiedenen Teilaufgaben, die weitgehend unabhängig voneinander behandelt werden können.

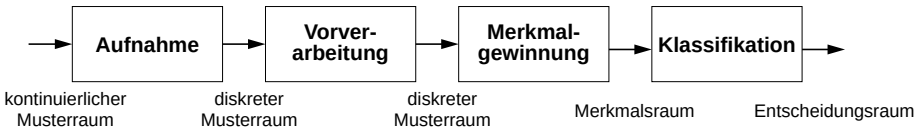

Abbildung 7.1: Schematischer Aufbau eines Klassifikationsystems.

Zunächst werden die zu erkennenden Muster durch ein Aufnahmegerät in einen diskreten Musterraum abgebildet. Anschließend erfolgt durch die Vorverarbeitung eine Bereinigung und Normalisierung der eingelesenen Muster. In der Merkmalgewinnung werden aus den vorverarbeiteten Mustern die wesentlichen Informationen extrahiert, die eine Unterscheidung von Mustern verschiedener Klassen ermöglichen. Ziel der Merkmalgewinnung ist die Komprimierung der Muster mit einem möglichst geringen Informationsverlust. Schließlich werden die Muster durch die Klassifikation aufgrund ihrer Merkmalsvektoren bestimmten Klassen zugeordnet. Die Güte eines Klassifikationssystems hängt von dem Zusammenspiel aller Phasen ab, wobei Merkmalgewinnung und Klassifikation im Vordergrund stehen.

Die Modularisierung eines Klassifikationssystems erlaubt es, einzelne Komponenten auszutauschen und so deren Eignung für das restliche System zu ermitteln. Da der Schwerpunkt dieses Kapitels in der Klassifikation liegt, werden Vorverarbeitung und Merkmalgewinnung im folgenden festgehalten. Auf diese Weise können verschiedene Klassifikationsverfahren miteinander verglichen werden.

Praktische Anwendungen von verschiedenen Mustererkennungssystemen sind in [Mai93] zu finden.

7.2 Mustermengen

Das amerikanische *National Institute of Standards and Technology* (NIST) veranstaltete mit der amerikanischen Volkszählungsbörde *Bureau of Census* 1992 in Gaithersburg (USA) die Konferenz *The First Census Optical Character Recognition System Conference (COCR)* [Nat92]. Ziel dieser Konferenz war es, den Entwicklungsstand im Bereich der automatischen Mustererkennung festzustellen. Zu diesem Zweck wurde eine umfangreiche, aus Ziffern, Groß- und Kleinbuchstaben bestehende Mustermenge zusammengestellt.

Die Mustermenge wurde unter dem Namen *NIST Special Database 3* (SD3) [GW92] als offizielle Lernmenge zur oben genannten OCR-Konferenz veröffentlicht. Die Datenbasis SD3 enthält ganzseitige Binärbilder von 2100 sogenannten handgeschriebenen Musterformularen (engl.: *handwritting sample form*, hsf). Die Mustermenge setzt sich aus den NIST-Partitionen hsf0, hsf1, hsf2 und hsf3 zusammen. Jedes Formular wurde von je einem Schreiber ausgefüllt. Die Schreiber rekrutierten sich aus Mitarbeitern der Census-Behörde. Die gesamte Mustermenge besteht aus insgesamt 313389 segmentierten und referenzierten Zeichen. Diese Menge setzt sich wiederum aus 223125 Ziffern, 44951 Großbuchstaben und 45313 Kleinbuchstaben zusammen.

Als Testmenge für die COCR wurde die Mustermenge *NIST Special Database 7* (SD7)[1] verwendet, die aus 500 Formularen der NIST-Partition hsf4 erstellt wurde. Im Gegensatz zur SD3 wurde diese Menge nicht von Census-Mitarbeitern, sondern von Schülern einer

[1] Während der Konferenz wurde diese Mustermenge als TD1 bezeichnet.

High School geschrieben. Durch den unterschiedlichen Personenkreis der Schreiber, die diese beiden Mustermengen erstellten, ergaben sich recht unterschiedliche Mustermengen. Grother hat mittels der PNN-Methode (s. Abschnitt 7.6.2) eine Crossvalidierung der Mustermengen SD3 und SD7 durchgeführt ([Nat92]) und dabei gezeigt, daß SD7 eine allgemeinere Datenmenge als SD3 ist.

Weitere 500 Formulare stehen in der NIST-Partition hsf6 zur Verfügung. Die Schreiber dieser Muster stammen von Angestellten der Census-Niederlassung in Suitland, Maryland. Die Tabelle 7.1 gibt einen Überblick über Umfang und Herkunft der zur Verfügung stehenden Mustermengen. Eine noch umfangreichere Datenbasis, die die hier vorgestellten umfaßt, wurde von NIST mit der *Special Database 19* (SD19) zusammengestellt ([Gro95]).

Partition	Anzahl der Formulare	Ziffern	Buchstaben		Herkunft der Schreiber	Datenbasis
			Groß-	Klein-		
hsf0	500	53449	10790	10968	Census Field	SD3
hsf1	500	53312	10662	10784	Census Field	SD3
hsf2	500	52467	10863	10883	Census Field	SD3
hsf3	600	63896	12636	12678	Census Field	SD3
hsf4	500	58646	11941	12000	High School	SD7
hsf6	499	61094	61094	12205	Census MD	

Tabelle 7.1: Überblick über die NIST-Datenbasen

Die Formulare wurden mit 300×300 dpi eingescannt. Die in den Formularen enthaltenen Zeichen wurden extrahiert und einzeln als 128×128 große Pixelbilder abgelegt. Die Segmentierung wurde überprüft und manuell korrigiert, so daß die Fehlerrate der segmentierten Zeichen bezüglich ihrer Klassifikation weniger als 0.1 % beträgt.

Zur Bearbeitung von Mustererkennungsproblemen sind zwei disjunkte Mustermengen notwendig, eine Lernmenge und eine Testmenge. Die Zusammenstellung dieser Mustermengen hat erheblichen Einfluß auf das Ergebnis der Klassifikation. Die Lernmenge sollte in Bezug auf die Anwendung möglichst repräsentativ sein.

Die hier verwendeten Mustermengen sind aus Ziffern der oben beschriebenen Datenbasen zusammengestellt worden. Um sicherzustellen, daß die Muster von möglichst vielen verschiedenen Schreibern stammen, wurde von jedem Formular nur ein Satz von Ziffern gewählt. Sofern nichts anderes angegeben ist, wurden in allen Fällen die Ziffern aufgenommen, die als erste ihrer Klasse auftraten. So wurden aus der SD3-Mustermenge zwei disjunkte Mengen gebildet. Die Menge D_1 besteht aus 10575 Ziffern und wird als Lernmenge verwendet. Die Mustermenge D_2 enthält 9494 Ziffern, die von 946 verschiedenen Schreibern stammen. Die Mustermengen D_3 und D_4 wurden aus der Datenbasis SD7 bzw. aus der NIST-Partition hsf6 gewonnen. Die Mustermengen $\mathsf{D}_1, ..., \mathsf{D}_4$ sind paarweise disjunkt.

In Tabelle 7.2 ist für die verschiedenen Mustermengen die Anzahl der Muster jeder Klasse angegeben.

Ω	hsf	$\sum_k \lvert\Omega_{[k]}\rvert$	$\lvert\Omega_{[k]}\rvert$									
			0	1	2	3	4	5	6	7	8	9
D_1	0,1,2	10575	1058	1058	1057	1057	1058	1056	1058	1058	1058	1057
D_2	2,3	9454	945	946	946	946	946	944	945	945	945	946
D_3	4	4999	500	500	500	500	500	499	500	500	500	500
D_4	6	4990	499	499	499	499	499	499	499	499	499	499

Tabelle 7.2: Umfang der Mustermengen für Ziffern.

7.3 Vorverarbeitung

Unter Vorverarbeitung ist eine Transformation zu verstehen, die ein vorgegebenes Muster in ein anderes überführt, das sich für die nachfolgenden Schritte des Mustererkennungsprozesses besser eignet. Der Nutzen eines Verfahrens zur Vorverarbeitung kann letztlich nur bezüglich des gesamten Klassifikationssystems beurteilt werden.

Dennoch gibt es einige Grundtechniken der Vorverarbeitung, die zum Ziel haben, die Variabilität zwischen den Mustern einer Klasse zu reduzieren und damit den Klassifikationsprozeß zu vereinfachen. Zu diesen Grundtechniken zählt die *Verbesserung* und die *Normierung* von Mustern. Unter Verbesserung oder Säuberung ist die Beseitigung fehlerhafter oder unnötiger Funktionswerte des Musters zu verstehen. Die Normierung gleicht die Werte einiger Parameter an Normalwerte oder Normalwertebereiche an.

Für die hier gestellte Klassifikationsaufgabe, die in der Erkennung handgeschriebener Ziffern besteht, werden die von NIST zur Verfügung gestellten, digitalisierten Ziffern durch ein Standardverfahren größennormiert und gesäubert ([SGH+89]). Das Verfahren normiert die Muster derart, daß sie ein gegebenes Raster der Größe 16×16 mindestens in einer Richtung vollständig ausfüllen.

In Abbildung 7.2 ist eine Auswahl digitalisierter Ziffern, wie sie von NIST zur Verfügung gestellt wurden, zu sehen. Das Ergebnis der Vorverarbeitung ist in Abbildung 7.3 dargestellt. Durch die Vorverarbeitung werden die 128×128 Pixel großen Muster in 16×16 Pixel große Muster transformiert. Die Anzahl ihrer Pixel wird damit um den Faktor 64 reduziert.

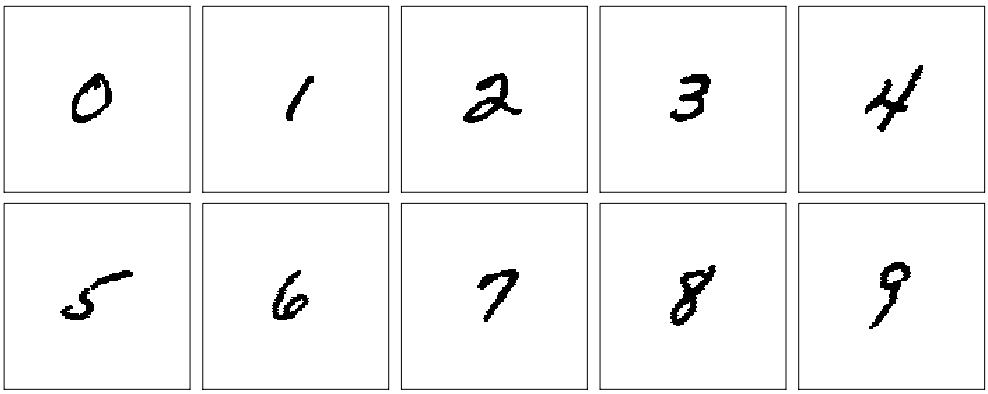

Abbildung 7.2: Digitalisierte Ziffern aus der NIST-Mustermenge bestehend aus 128×128 Pixel.

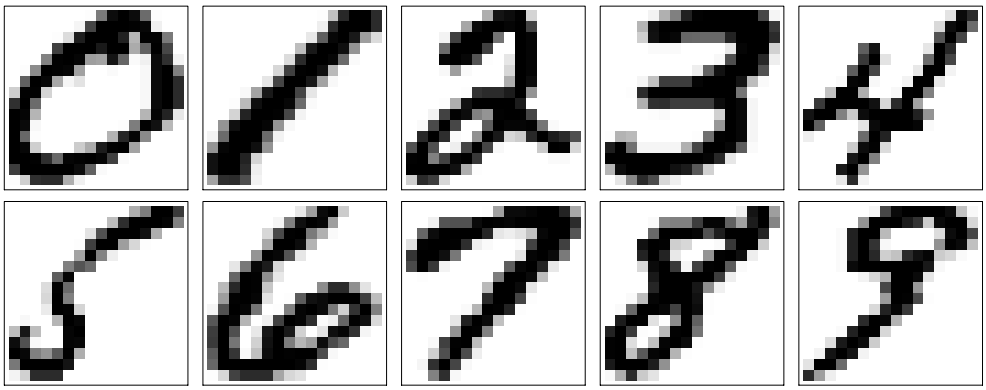

Abbildung 7.3: Vorverarbeitete Ziffern bestehend aus 16×16 Pixel.

Die vorverarbeiteten Muster bestehen aus 256 Werten im Intervall $[0, 1]$ und aus einer zweidimensionalen Nachbarschaftsrelation der Werte. Die in den zweidimensionalen Mustern enthaltene Information wird weiter komprimiert, indem ausschließlich die 256 Grauwerte verwendet werden. Das Resultat der Vorverarbeitung ist damit ein aus 256 Werten bestehender Grauwertvektor. Die Grauwerte der einzelnen Pixel der vorverarbeiteten Ziffern aus Abbildung 7.3 sind in Abbildung 7.4 als Balkendiagramm dargestellt.

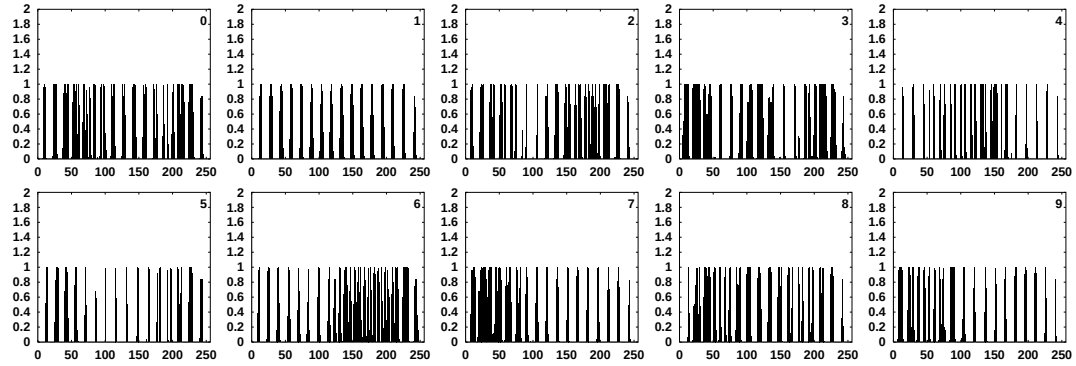

Abbildung 7.4: Balkendiagramm der Grauwerte der einzelnen Pixel der vorverarbeiteten Ziffern aus Abbildung 7.3.

7.4 Merkmalsgewinnung

Ein vorverarbeitetes Muster der Größe 16×16 besteht aus 256 Werten, die als Merkmale verwendet werden können. Da sich die Konstruktion von Klassifikatoren mit zunehmender Dimension des Merkmalsvektors erschwert, wird zur Merkmalsgewinnung eine Methode verwendet, die diese 256 Werte auf wenige signifikante Merkmale reduziert.

Eine Methode zur Merkmalsreduktion, die sich in der Klassifikation bestens bewährt hat, ist die *Hauptachsentransformation*, die auch als *Karhonen-Loève-Transformation* bezeichnet wird ([Gro92], [Nie83], [VBH+91]). Diese Koordinatentransformation besteht aus einer Translation, die den Koordinatenursprung in den Schwerpunkt der Masseverteilung legt, und einer Rotation des Koordinatensystems in die Richtung der Hauptachsen der

Masseverteilung. Diese Transformation ist dieselbe, die bei der BGA-Mutation $\mathbb{M}^{\mathrm{BGA,PCA}}$ verwendet wird, um das Koordinatensystem an die Fitnesslandschaft anzupassen (s. Abschnitt 3.5.5). Die Hauptachsentransformation ist damit ein analytisches, problemabhängiges Verfahren, das aus einem Mustervektor $\vec{f}$ einen Merkmalsvektor $\vec{x}$ durch Transformation nach einem orthonormalen Basissystem Φ' generiert:

$$\vec{x} = \Phi \vec{f} \tag{7.3}$$

Die Transformation Φ wird durch die Bestimmung des Eigensystems der Kovarianzmatrix des mustererzeugenden Prozesses berechnet. Φ setzt sich aus den n Eigenvektoren mit den größten Eigenwerten zusammen. Auf diese Weise wird der Rekonstruktionsfehler, der der Summe der Eigenwerte der nicht verwendeten Eigenvektoren entspricht, minimiert.

Die Merkmalsvektoren, die durch die Hauptachsentransformation aus den in Abbildung 7.4 dargestellten Grauwertvektoren entstehen, sind in Abbildung 7.5 dargestellt. Aufgrund dieser Merkmalsvektoren erfolgt die Klassifikation der Muster.

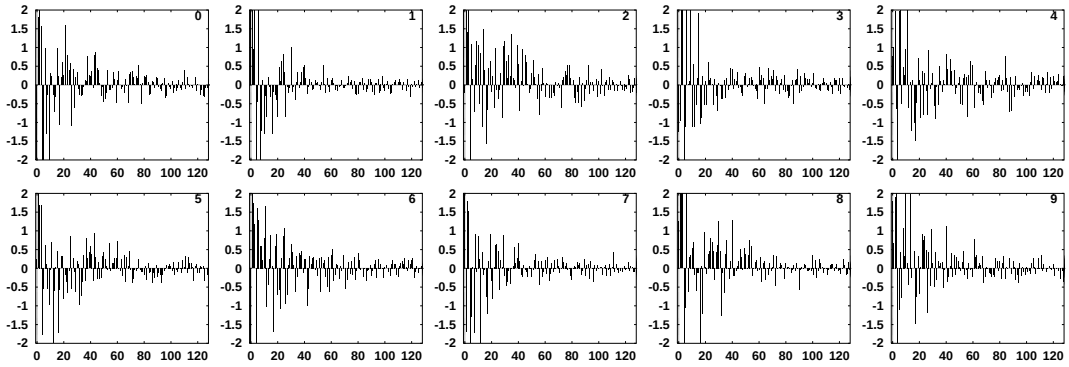

Abbildung 7.5: Die ersten 128 Merkmale, die aus den in Abbildung 7.4 dargestellten Grauwertvektoren erzeugt werden.

Die Rücktransformation eines Merkmalsvektors in einen Eingangsvektor macht den Informationsverlust in Abhängigkeit der Anzahl der verwendeten Merkmale deutlich. In Abbildung 7.6 ist die Rücktransformation der Ziffer 2 für unterschiedliche Merkmalslängen dargestellt.

7.5 Klassifikationsansatz

7.5.1 Allgemeiner Polynomansatz

Die Aufgabe der Klassifikation besteht darin, Muster aufgrund ihrer Merkmale bestimmten Klassen zuzuordnen. Einen Eindruck des Klassifikationsproblems liefert Abbildung 7.7, in der für die zehn Ziffernklassen die Werte der ersten beiden Merkmale gegeneinander eingetragen sind. Die Aufgabe eines Klassifikators besteht in der Schätzung der Klassenzugehörigkeit aufgrund des Punktes im Merkmalsraum. In der Abbildung sind deutlich markante Häufungspunkte zu erkennen. Aufgrund der Überlappungsbereiche der verschiedenen Klassen reichen zwei Merkmale für dieses Problem offenbar nicht aus.

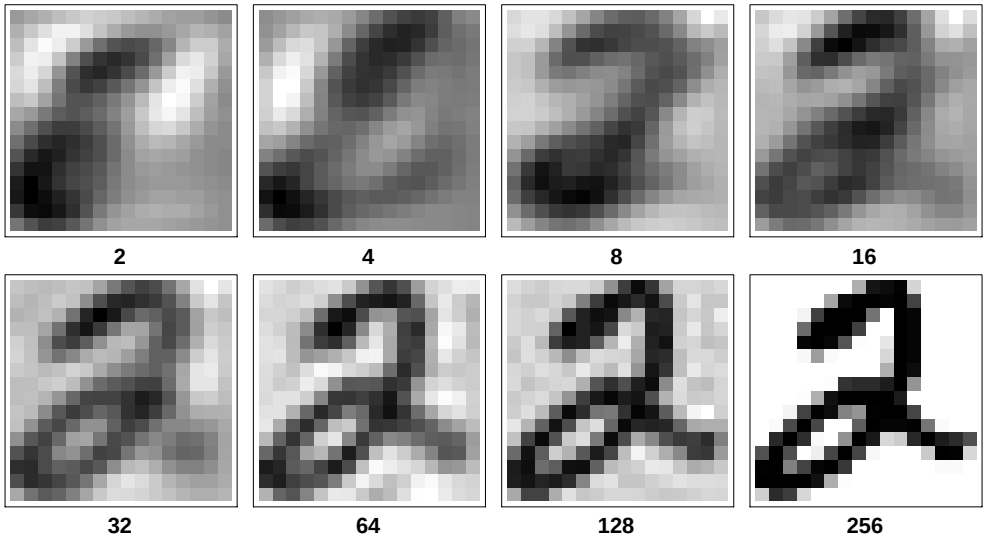

Abbildung 7.6: Rücktransformation der Ziffer 2 in Abbildung 7.3 aus Merkmalsvektoren unterschiedlicher Länge.

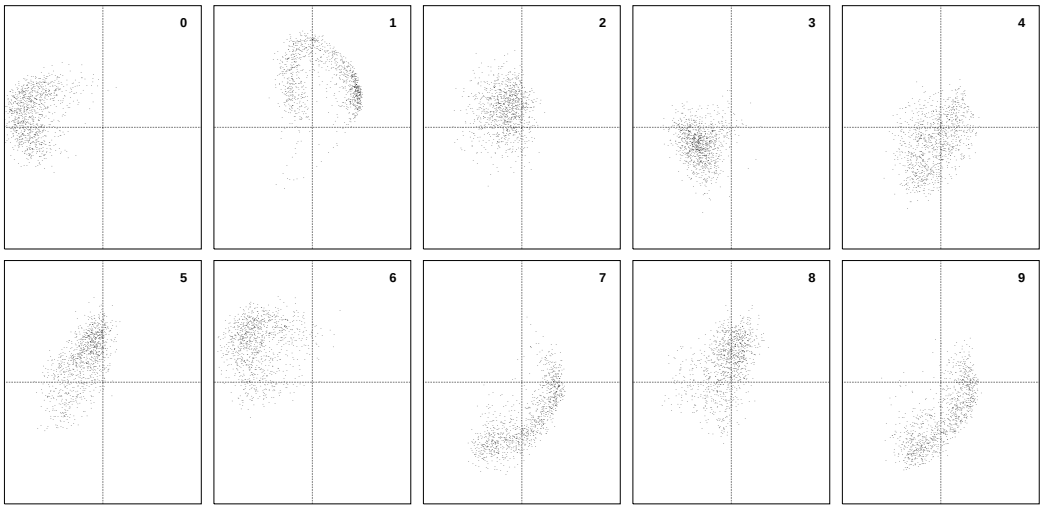

Abbildung 7.7: Die beiden wichtigsten Merkmale aller Muster der Lernmenge $\mathsf{D}_1^{\mathrm{L}}$.

Die Klassifikation entspricht einer Abbildung aus dem subsymbolischen Merkmalsraum in den symbolischen Entscheidungsraum. Diese Abbildung wird auch als Unterscheidungsfunktion bezeichnet.

Der Entwurf eines Klassifikators besteht also in der Konstruktion einer geeigneten Unterscheidungsfunktion. Typischerweise schränkt man die Funktionenklasse, aus deren Funktionen die Unterscheidungsfunktion zusammengesetzt wird, ein und legt sie durch charakteristische Konstruktionsregeln fest. Innerhalb dieser Grundkonstruktion verfügt jede Funktionenklasse über eine Anzahl freier Parameter, durch die die Funktionen der jeweiligen Funktionenklasse frei zu formen sind.

Eine häufig verwendete Grundkonstruktion, die sich auf die Funktionenklasse der Polynome beschränkt, wird Polynomansatz genannt ([Sch92]). In diesem Fall besteht eine

Unterscheidungsfunktion aus einer Linearkombinationen von Produkttermen. Die Unterscheidungsfunktion der Klasse k hat folgendes Aussehen:

$$\begin{aligned} d_k(\vec{x}) &= a_{0,k}\ p_0(\vec{x}) + a_{1,k}\ p_1(\vec{x}) + \ldots \\ &= \vec{a}_k^T \vec{p}(\vec{x}) \end{aligned} \tag{7.4}$$

Die Abbildung $\vec{p}(\vec{x})$ beschreibt die Art der Produktterme und damit die *Polynomstruktur*. Durch die Zusammenfassung der κ Unterscheidungsfunktionen zu dem Schätzvektor $\vec{d}$ ergibt sich:

$$\vec{d}(\vec{x}) = \mathbf{A}^T \cdot \vec{p}(\vec{x}) \tag{7.5}$$

Während die Wahl der Polynomstruktur eine Frage des geschickten Entwurfs ist, bildet die Bestimmung der Koeffizientenmatrix $\mathbf{A}$ eine Aufgabe der kontinuierlichen Parameteroptimierung, die durch die in dieser Arbeit vorgestellten Suchverfahren gelöst werden kann.

Als Polynomstruktur liegt die Wahl eines vollständigen Polynoms eines bestimmten Grades C nahe. In diesem Fall berechnet sich die Anzahl der Koeffizienten bei F Merkmalen durch:

$$n = \binom{F+C}{C} \tag{7.6}$$

Mit $C = 2$ und $F = 3$ ergibt sich für die k-te Klasse folgende quadratische Unterscheidungsfunktion:

$$d_k(\vec{x}) = a_{0,k} + a_{1,k}x_1 + a_{2,k}x_1^2 + a_{3,k}x_1x_2 + a_{4,k}x_2 + a_{5,k}x_2^2 \tag{7.7}$$

Nachdem die κ Unterscheidungsfunktionen in der sogenannten Lernphase durch eine Optimierung ermittelt sind, kann die eigentliche Entscheidung erfolgen. Eine einfache Möglichkeit besteht darin, ein Muster der Klasse zuzuweisen, deren Unterscheidungsfunktion den größten Wert liefert.

Die Berechnung der Koeffizientenmatrix $\mathbf{A}$ stellt eine Optimierungsaufgabe dar, die unter Umständen analytisch oder durch numerische Optimierung gelöst werden kann. Eines der bekanntesten analytischen Verfahren ist der in Abschnitt 7.6.1 beschriebene Quadratmittelansatz von Schürmann ([Sch92]).

7.5.2 Polynomansatz mit erweitertem Optimierungsziel

Der in dieser Arbeit entwickelte Klassifikator basiert auf einem Polynomansatz mit erweitertem Optimierungskriterium, das neben der quadratischen Abweichung von Schätz- und Zielwert auch den Klassifikationsfehler selbst einbezieht. Aufgrund des komplexen Optimierungsziels ist eine analytische Bestimmung der Koeffizientenmatrix nicht möglich. Die Bestimmung der Koeffizientenmatrix erfolgt daher durch ein Optimierungsverfahren. Da sich das Wettbewerbsmodell `cbga1`, in dem vier BGA mit unterschiedlichen Schrittweitenparametern konkurrieren, besonders bei hochdimensionalen Funktionen als robustes und effizientes Suchverfahren erwiesen hat, wird es für die Optimierung dieser Problemstellung eingesetzt.

Zur Vereinfachung der Optimierung werden die Unterscheidungsfunktionen separat nach dem festgelegten Optimierungsziel ermittelt. Jede Unterscheidungsfunktion entscheidet, ob ein Muster zur entsprechenden Klasse gehört oder nicht. Damit wird das κ-Klassenproblem in κ Zweiklassenprobleme transformiert.

Der Zielwert der Unterscheidungsfunktion $f_{\mathrm{p}}^{k}(\vec{x})$ ist Eins, falls das Muster $\vec{x}$ zur Klasse k gehört, sonst Null. Der Fehler setzt sich aus den gewichteten quadratischen Abweichungen aller Muster der Lernmenge zusammen. Durch die Gewichtungen wurden die Relevanz der Fehler und das Größenverhältnis der Klassen berücksichtigt. Da eine Abweichung vom Zielwert für die Klassifikation schwerwiegender ist, falls sich der Schätzwert dem Zielwert der jeweils anderen Klasse nähert, werden die Fehler zusätzlich aufgrund ihrer Richtung gewichtet.

$$
\begin{aligned}
f_{\mathrm{p}}^{k}(\vec{x}) \;=\; & \frac{1}{|\Omega_k|} \sum_{\vec{x}\in\Omega_k} \left((1 \Leftrightarrow d_k(\vec{x}))^2 \cdot \begin{cases} 1 & : \quad d_k(\vec{x}) \geq 1 \\ \gamma & : \quad d_k(\vec{x}) < 1 \end{cases} \right) + \\
& \frac{1}{|\Omega_{\overline{k}}|} \sum_{\vec{x}\notin\Omega_k} \left((0 \Leftrightarrow d_k(\vec{x}))^2 \cdot \begin{cases} 1 & : \quad d_k(\vec{x}) \leq 0 \\ \gamma & : \quad d_k(\vec{x}) > 0 \end{cases} \right)
\end{aligned}
\tag{7.8}
$$

mit γ als Gewichtungsfaktor. $\Omega_{\overline{k}}$ bezeichnet die Menge aller Muster, die nicht zur Klasse k gehören.

Der Fehlerterm, der das Klassifikationsergebnis des Zweiklassenproblems widerspiegelt, setzt sich aus der Anzahl der durch die Unterscheidungsfunktion $d(\vec{x})$ falsch klassifizierten Muster der eigenen Klasse (err_k) und der falsch klassifizierten Muster der anderen Klassen ($err_{\overline{k}}$) zusammen.

$$
f_{\mathrm{c}}^{k}(\vec{x}) = \frac{1}{|\Omega_k|} err_k + \frac{1}{|\Omega_{\overline{k}}|} err_{\overline{k}} \tag{7.9}
$$

Aus den beiden Fehlertermen $f_{\mathrm{p}}^{k}(\vec{x})$ und $f_{\mathrm{c}}^{k}(\vec{x})$ wird die Zielfunktion für die k-te Klasse zusammengesetzt:

$$
\boxed{f^{k}(\vec{x}) = \gamma_{\mathrm{p}} f_{\mathrm{p}}^{k}(\vec{x}) + \gamma_{\mathrm{c}} f_{\mathrm{c}}^{k}(\vec{x})} \tag{7.10}
$$

mit den Gewichtungsfaktoren γ_{p} und γ_{c}. Die Gewichtungsfaktoren müssen so gewählt werden, daß der Fehlerterm $f_{\mathrm{c}}^{k}(\vec{x})$ einen geringeren Einfluß auf das Gesamtergebnis hat als der erste Fehlerterm. In der folgenden Anwendung werden beide Faktoren auf 1.0 gesetzt.

Als Polynomstruktur wird ein vollständiges Polynom zweiten Grades gewählt. Bei 32 Merkmalen sind damit pro Unterscheidungsfunktion 561 Koeffizienten einzustellen. Die Bestimmung der Koeffizienten erfolgt durch das Wettbewerbsmodell `cbga1`. In Tabelle 7.3 ist das Klassifikationsergebnis dieses Ansatzes dargestellt.

7.6 Vergleich zu anderen Verfahren

7.6.1 Quadratmittelansatz

Der Quadratmittelansatz (QMA) von Schürmann basiert, wie auch das oben beschriebene Verfahren, auf einem Polynomansatz. Hier ist das Optimierungsziel die Minimierung des

$\Omega_{(k)}$	d_k										$\sum$	err	
	0	1	2	3	4	5	6	7	8	9		#	%
0	1055	0	0	0	0	0	3	0	0	0	1058	3	0.28
1	0	1057	0	0	0	0	0	0	0	1	1058	1	0.09
2	0	0	1052	0	1	0	0	2	2	0	1057	5	0.47
3	0	0	1	1051	0	1	0	0	4	0	1057	6	0.57
4	0	0	0	0	1057	0	0	0	0	1	1058	1	0.09
5	0	0	0	3	0	1049	2	0	0	2	1056	7	0.66
6	0	1	0	0	1	0	1055	0	0	1	1058	3	0.28
7	0	1	0	0	0	0	0	1055	1	1	1058	3	0.28
8	0	3	1	5	0	4	0	1	1044	0	1058	14	1.32
9	0	0	0	0	2	0	0	3	2	1050	1057	7	0.66
$\sum$	1055	1062	1054	1059	1061	1054	1060	1061	1053	1056	10575	50	0.47

Tabelle 7.3: Klassifikationsergebnisse der durch den Polynomansatz mit erweitertem Optimierungsziel gewonnenen Unterscheidungsfunktionen auf die Lernmenge $\mathsf{D}_1^{\mathrm{L}}$. Für jede Unterscheidungsfunktion ist pro Klasse die Anzahl der Entscheidungen für die jeweilige Klasse eingetragen.

Ω_k	d_k										$\sum$	err	
	0	1	2	3	4	5	6	7	8	9		#	%
0	1047	2	0	1	1	3	3	0	1	0	1058	11	1.04
1	0	1051	3	0	0	0	0	3	0	1	1058	7	0.66
2	1	0	1046	0	3	0	0	3	4	0	1057	11	1.04
3	0	0	3	1041	0	3	0	3	7	0	1057	16	1.51
4	0	0	0	0	1053	0	1	0	0	4	1058	5	0.47
5	0	0	0	9	0	1038	3	0	1	5	1056	18	1.70
6	1	3	0	0	3	1	1050	0	0	0	1058	8	0.76
7	0	8	0	0	6	1	0	1040	1	2	1058	18	1.70
8	2	3	2	8	2	7	0	1	1031	2	1058	27	2.55
9	1	1	0	6	7	3	0	5	5	1029	1057	28	2.65
$\sum$	1052	1068	1054	1065	1075	1056	1057	1055	1050	1043	10575	149	1.41

Tabelle 7.4: Klassifikationsergebnisse der durch den Quadratmittelansatz (QMA) gewonnenen Unterscheidungsfunktionen für die Lernmenge $\mathsf{D}_1^{\mathrm{L}}$. Für jede Unterscheidungsfunktion ist pro Klasse die Anzahl der Entscheidungen für die jeweilige Klasse eingetragen.

Betragsquadrates des Fehlervektors, der die Differenz zwischen Schätzvektor und Zielvektor beschreibt. Da die Optimierung der Koeffizientenmatrix $\mathbf{A}$ nach dem Quadratmittelkriterium auf ein lineares Gleichungssystem bezüglich der Koeffizienten führt, kann die Bestimmung der Koeffizienten sehr schnell durch ein entsprechendes Standardverfahren gelöst werden. Eine ausführliche Beschreibung dieses Verfahrens ist in [Sch92] zu finden.

Die Klassifikation durch den Quadratmittelansatz arbeitet, wie der oben eingeführte Polynomansatz mit erweitertem Optimierungsziel, mit einem Polynom zweiten Grades und 32 Merkmalen. Bei 10 Klassen sind also 5610 Koeffizienten zu ermitteln. Da die zugrunde liegende Polynomstruktur der im Polynomansatz mit erweitertem Optimierungsziel verwendeten entspricht, ist ein direkter Vergleich zwischen den beiden Verfahren möglich. Unterschiede im Klassifikationsergebnis sind also nur durch die Einstellung der Koeffizienten zurückzuführen.

Das Klassifikationsergebnis, das der Quadratmittelansatz für die Lernmenge $\mathsf{D}_1^{\mathrm{L}}$ erzielt, ist in Tabelle 7.4 eingetragen. Mit einem Gesamtfehler von 1.41% liegt er deutlich über dem Ergebnis des Polynomansatzes mit erweitertem Optimierungsziel.

7.6.2 Nearest-Neighbor-Klassifikatoren

Nearest-Neigbor-Klassifikatoren sind seit Jahrzehnten Forschungsgegenstand im Bereich der Klassifikation [Das91]. Die NN-Klassifikatoren zählen zu den parameterfreien Klassifikatoren, die keine Annahme über die Verteilung der Muster voraussetzen. NN-Klassifikatoren gehen davon aus, daß im Merkmalsraum eng beieinander liegende Muster mit hoher Wahrscheinlichkeit derselben Klasse angehören. Ein zu klassifizierendes Muster wird daher der Klasse zugeordnet, der auch sein nächster bekannter Nachbar im Merkmalsraum angehört.

Im folgenden werden zwei einfache aber effektive Beispiele dieser Verfahren vorgestellt.

k-Nearest-Neighbor (k-NN)

Für den Fall, daß $k = 1$ ist, ist das Verfahren eine Ausprägung des Minimum-Abstands-Klassifikators. Der Unterschied zu dieser Methode besteht ausschließlich darin, daß die gesamte Lernstichprobe als Referenzmenge benutzt wird. Cover and Hart haben in [CH67] gezeigt, daß dieses Verfahren ein gutes asymptotisches Verhalten aufweist: Für große Referenzmengen ist seine Fehlerrate durch den doppelten Wert der Bayesschen (d.h. der minimalen) Fehlerrate beschränkt.

Die Diskriminanz-Funktion des 1-NN hat folgende Form:

$$d_i(\vec{x}) \quad = \quad \Leftrightarrow \min_{1 \le j \le M_i} d^2\left(\vec{x}, \vec{x}_j^{(i)}\right) \tag{7.11}$$

Für den allgemeineren Fall ($k > 1$) nimmt man die Klasse als Hypothese, die unter den k Nachbarn am häufigsten vertreten ist. Falls $\mathcal{S}_{\vec{x}}$ die Menge der k ähnlichsten Prototypen zu $\vec{x}$ ist, ist sie die Vereinigung der Mengen $\mathcal{S}_{\vec{x}}^{(i)}$, die ausschließlich Prototypen der Klasse i beinhalten. Damit sieht die Diskriminanzfunktion des k-NN folgendermaßen aus:

$$d_i(\vec{x}) = |\mathcal{S}_{\vec{x}}^{(i)}|. \tag{7.12}$$

Weighted Several Nearest-Neighbor (WSNN)

Diese Klassifikationsmethode, die eine Verfeinerung des Nearest Neighbor Verfahrens darstellt, ist in [GC93] ausführlich beschrieben. Sie unterscheidet sich in zwei Punkten von dem oben beschriebenen k-NN-Verfahren. Zum einen ist die Anzahl der Nachbarn, die über die Klassenzugehörigkeit eines Musters entscheiden, eine Zufallsvariable. Die Anzahl der Nachbarn ist also variabel. Zum anderen werden die Stimmen der Nachbarn mit deren Abstand zum unbekannten Muster gewichtet.

Die Methode geht wie folgt vor: Zunächst wird der Prototyp bestimmt, der den kleinsten quadratischen euklidischen Abstand zum unbekannten Muster aufweist. Der Abstand dieses nächsten Nachbarn von dem unbekannten Muster legt die Größe der Nachbarschaft fest. Alle Prototypen, deren Abstand zum unbekannten Muster kleiner ist als der α-fache Abstand des nächsten Nachbarn, werden als benachbarte Prototypen bezeichnet. Der konstante Faktor α legt die Größe der Nachbarschaft fest. Die Entscheidung wird aufgrund der Klassenzugehörigkeit der benachbarten Prototypen getroffen. Die Stimmenanzahl der Klasse i wird durch die Wurzel der Summe der quadratischen Abstände der Nachbarn aus Klasse i von der Unbekannten geteilt.

Seien

$$\mathcal{S}_{\vec{x}}^{(i)} = \text{die Menge der Indizes der Merkmale der Klasse i aus der } \alpha\text{-Nachbarschaft der Unbekannten } \vec{x} \tag{7.13}$$

$$V_{\vec{x}}^{(i)} = |\mathcal{S}_{\vec{x}}^{(i)}| \tag{7.14}$$

Dann lassen sich die Unterscheidungsfunktionen schreiben als:

$$d_i(\vec{x}) = \begin{cases} V_{\vec{x}}^{(i)} \left(\sum_{j \in \mathcal{S}_{\vec{x}}^{(i)}} d^2(\vec{x}, \vec{x_j}^i) \right)^{-0.5} & : \text{ falls } V_{\vec{x}}^{(i)} > 0 \\ 0 & : \text{ sonst} \end{cases} \tag{7.15}$$

Probabilistic Neural Net (PNN)

Der von Specht entwickelte Klassifikator ist trotz des Namens eng mit den Nearest-Neighbor-Verfahren verwandt ([Spe90]). Hierbei definiert jedes Muster der Lernmenge den Mittelpunkt einer Funktion, die mit zunehmender Entfernung von dem Muster monoton abnimmt. Die Klassifikation des unbekannten Musters $\vec{x}$ wird aufgrund der Funktionswerte aller Muster aus der Lernmenge an der Stelle $\vec{x}$ vorgenommen. Dazu werden zunächst die Funktionswerte der Muster jeder Klasse an der Stelle $\vec{x}$ summiert. Das Muster wird schließlich der Klasse zugeordnet, die an der Stelle den höchsten Wert aufweist.

Für eine kreissymmetrische Gauss-Funktion hat die Unterscheidungsfunktion folgende Form:

$$d_i(\vec{x}) = \sum_{j=1}^{|\Omega_{(i)}|} \exp\left(\Leftrightarrow \frac{1}{2\sigma^2} d^2\left(\vec{x}, \vec{x}_j^{(i)}\right) \right) \tag{7.16}$$

wobei der Parameter σ ein externer Parameter ist, der an das Problem anzupassen ist. Die im nächsten Abschnitt dargestellten Klassifikationsergebnisse wurden mit $\sigma = 1.0$ erzielt.

7.6.3 Multilayer-Perceptron (MLP)

Die bekannteste Form eines neuronalen Netzes ist das *Multilayer-Perceptron (MLP)*. Aufgrund des hohen Bekanntheitsgrades dieses Klassifikationsansatzes wird auf eine detaillierte Beschreibung verzichtet.

Die Approximation der Unterscheidungsfunktionen erfolgt durch in Schichten angeordnete Neuronen. Ein Neuron ist definiert als:

$$y = f\left(\sum_j w_j x_j \right) \tag{7.17}$$

wobei f in der Regel eine Sigmoidfunktion, $\vec{x}$ der Eingangsvektor und y der Ausgang des Neurons ist.

Ein MLP wird durch folgende Größen spezifiziert:

$$N^{(i)} = \text{Größe der } i\text{-ten Schicht} \tag{7.18}$$

$$f(x) = \text{Sigmoidfunktion} \tag{7.19}$$

Während die Größen der Eingangs- und Ausgangsschicht durch die Problemstellung vorgegeben sind, sind die Größen der sogenannten versteckten Schichten frei wählbar.

In [GC93] ist die Unterscheidungsfunktion eines aus drei Schichten bestehenden Multilayer-Perceptrons angegeben:

$$d_i(\vec{x}) = f\left(b_i^{(2)} + \sum_{j=1}^{N^{(1)}} w_{ij}^{(2)} \quad f\left(b_j^{(1)} + \sum_{k=1}^{N^{(0)}} w_{jk}^{(2)} x_k\right)\right) \tag{7.20}$$

wobei $w_{ij}^{(k)}$ das Gewicht, das den i-ten Knoten der k-ten Schicht mit dem j-ten Knoten der $(k \Leftrightarrow 1)$-ten Schicht verbindet und $b_i^{(k)}$ das Bias-Gewicht des i-ten Knotens der k-ten Schicht bezeichnet.

Die Lösung des MLP besteht darin, die Gewichte so einzustellen, daß für die Lernmenge der mittlere quadratische Fehler zwischen Schätz- und Zielwerten minimiert wird. Da für das Multilayer-Perceptron keine analytische Lösung existiert, wird für die Bestimmung der Gewichte ein iteratives Gradientenverfahren verwendet. Dieses sogenannte *error backpropagation* steuert über die Parameter Lernrate und Momentum die Geschwindigkeit und Güte des Lernverhaltens. Das hier verwendete MLP, dessen Implementierung auf dem Programm *bpsim.c* von Josuah C. Hoskins basiert, ist folgendermaßen spezifiziert:

$$\begin{aligned} N^{(0)} &= 32 \\ N^{(1)} &= 100 \\ N^{(2)} &= 10 \\ f(x) &= \frac{1}{1+e^{-x}} \\ \text{Lernrate} &= 0.1 \\ \text{Momentum} &= 0.0 \end{aligned}$$

7.6.4 Klassifikationsergebnisse

Zur Beurteilung der Güte des hier eingeführten Klassifikationsansatzes werden dessen Klassifikationsergebnisse mit den anderen Verfahren verglichen. Dazu werden die verschiedenen Klassifikatoren auf vier Mustermengen, zu denen auch die Lernmenge zählt, angewendet. In Tabelle 7.5 sind die jeweiligen Fehlerraten und die Lern- und Klassifikationszeiten eingetragen. Eine Beschreibung der Mustermengen ist in Abschnitt 7.2 zu finden.

Auffallend ist der Einfluß der Testmenge auf das Klassifikationsergebnis. Da die Lernmenge $\mathsf{D}_1^{\mathrm{L}}$ und die Testmenge $\mathsf{D}_2^{\mathrm{T}}$ von einem einheitlichen Personenkreis stammen, nämlich von Mitarbeitern des Census, sind die Klassifikationsergebnisse für diese Testmenge wesentlich besser als die der beiden anderen. Die Testmenge $\mathsf{D}_3^{\mathrm{T}}$ wurde von Schülern einer Highschool und die Testmenge $\mathsf{D}_4^{\mathrm{T}}$ von Mitarbeitern einer anderen Census-Niederlassung erstellt.

Die besten Klassifikationsergebnisse erzielt der in dieser Arbeit entwickelte Polynomansatz mit erweitertem Optimierungsziel, dessen Koeffizienten durch einen wettbewerbsgesteuerten BGA bestimmt werden. Gegenüber dem Quadratmittelansatz ist jedoch eine erheblich höhere Lernzeit erforderlich. Als bestes Nearest-Neighbor-Verfahren erweist sich das WSNN-Verfahren, dessen Resultate bei allen Mustermengen über denen des Quadratmittelansatzes liegen. Die Klassifikationsergebnisse der anderen Nearest-Neighbor-Verfahren liegen im Bereich des Quadratmittelansatzes. Trotz langer Lernzeit liefert das Neuronale Netz (MLP) die mit Abstand schlechtesten Klassifikationsergebnisse.

Verfahren	Testmengen				Zeiten	
	$\mathsf{D}_1^{\mathrm{L}}$	$\mathsf{D}_2^{\mathrm{T}}$	$\mathsf{D}_3^{\mathrm{T}}$	$\mathsf{D}_4^{\mathrm{T}}$	T_L	cps
cBGA	0.47	1.52	7.62	2.46	$>$ 200:00	831
QMA	1.41	2.08	8.82	3.25	00:29	831
kNN(1)	0.00	1.85	8.03	2.89	00:43	3
kNN(7)	1.83	2.14	9.05	3.15	00:00	3
WSNN	0.00	1.76	7.69	2.69	00:00	3
PNN	0.02	1.76	9.00	2.91	00:00	3
MLP	2.68	4.29	14.16	5.83	72:08	622

Tabelle 7.5: Klassifikationsergebnisse auf der Lernmenge ($\mathsf{D}_1^{\mathrm{L}}$) und verschiedenen Testmengen ($\mathsf{D}_2^{\mathrm{T}}$, $\mathsf{D}_3^{\mathrm{T}}$, $\mathsf{D}_4^{\mathrm{T}}$). Mit 'cBGA' wird hier der Polynomansatz mit erweitertem Optimierungsziel bezeichnet. T_{L} (hh:mm) gibt die Zeit an, die für das Lernen der Lernmenge aufgewendet wurde. *cps* gibt die Anzahl der Zeichen an, die das System in einer Sekunde auf einer Workstation des Typs SUN20 verarbeitet.

Der Vorteil der Klassifikationverfahren, die auf dem Polynomansatz basieren, und des neuronalen Netzes gegenüber den Nearest-Neighbor-Verfahren besteht in der schnellen Klassifikationszeit. Die Polynomklassifikatoren erreichen im Vergleich zu den Nearest-Neighbor-Verfahren eine 300-mal schnellere Klassifikationszeit.

Welcher der beiden Polynomansätze zu bevorzugen ist, hängt von der konkreten Problemstellung ab. Da die Klassifikationszeit beider Verfahren identisch ist, muß zwischen Lernaufwand und Klassifikationsgüte gewichtet werden. Der Lernaufwand des Polynomansatzes mit erweitertem Optimierungsziel kann durch die Verwendung von leistungsfähigeren Rechnern reduziert werden. Insbesondere läßt sich, wie im nächsten Kapitel gezeigt wird, durch die Parallelisierung des Optimierungsverfahrens eine erhebliche Beschleunigung und damit Reduzierung des Lernaufwandes erzielen.

Kapitel 8

Parallelisierungsaspekte

8.1 Effizienzmaße der Parallelverarbeitung

Unter Parallelverarbeitung wird die simultane Manipulation von Daten durch mehrere Berechnungseinheiten zur Erfüllung einer gemeinsamen Aufgabe verstanden.

Zur Quantifizierung des Nutzens, den der Einsatz eines Parallelrechners bringt, werden spezielle Effizienzmaße verwendet ([BH93]). Da die Parallelisierung von Evolutionären Algorithmen im wesentlichen mit MIMD-Architekturen erfolgt, werden hier nur Effizienzmaße dieser Parallelrechnerklasse eingeführt.

Das Kernziel der Parallelverarbeitung besteht in der Verkürzung der Zeit, die für die Bearbeitung einer Aufgabe benötigt wird. Während man in Einprozessorsystemen üblicherweise die *CPU-Zeit* als Leistungsmaß verwendet, müssen Leistungsmaße von Parallelrechnern auf der wirklichen Zeit basieren. Die Summe oder die maximale *CPU-Zeit* aller Einzelprozessoren erfaßt keine Wartezeiten, die aufgrund einer ungleichen Lastverteilung entstehen.

Definition 21 (Beschleunigung) *Die* Beschleunigung *(engl.:* Speedup*) eines parallelen Algorithmus, der auf p Prozessoren ausgeführt wird, ist definiert als:*

$$\mathcal{S}(p) = \frac{\begin{array}{c}\textit{Laufzeit des seriellen Codes} \\ \textit{auf einem speziellen Parallelrechner}\end{array}}{\begin{array}{c}\textit{Laufzeit des parallelen Codes} \\ \textit{auf p Prozessoren desselben Rechners}\end{array}} \tag{8.1}$$

Durch die Parallelisierung wird in der Regel eine *lineare Beschleunigung* $\mathcal{S}(p) = p$, wobei p die Anzahl der Prozessoren angibt, angestrebt. Eine *superlineare Beschleunigung*, die größer ist als die Anzahl der eingesetzten Prozessoren, ist häufig algorithmisch oder technisch bedingt.

Definition 22 (Effizienz) *Die* Effizienz *eines parallelen Algorithmus, der auf p Prozessoren ausgeführt wird, ist:*

$$\mathcal{E}(p) = \frac{\mathcal{S}(p)}{p} \tag{8.2}$$

Die Effizienz ist die durch die Anzahl der Prozessoren normalisierte Beschleunigung. Aus der linearen Beschleunigung folgt damit eine Effizienz von 1.0.

8.2 Parallelitätsebenen in Evolutionären Algorithmen

8.2.1 Inselmodell

Die gröbste Form der Parallelisierung in Evolutionären Algorithmen stellt die parallele Aufführung der einzelnen Populationen eines Inselmodells dar. Da sich die Populationen eines Inselmodells, wie es in Abschnitt 3.6.1 vorgestellt wurde, im wesentlichen unabhängig voneinander entwickeln, ist diese Form der Parallelisierung sehr effektiv. Der Kommunikationsaufwand, der zwischen den Populationen zum Austausch von Individuen erfolgt, ist gering und stellt daher keine hohen Anforderungen an das Kommunikationssystem. Diese Art der Parallelisierung kann nur auf das Inselmodell angewendet werden. Der maximale Parallelisierungsgrad ist hier durch die Anzahl der Populationen vorgegeben.

8.2.2 Farming

Das Farmingkonzept (Master-Slave-Konzept) sieht einen ausgezeichneten Prozeß, den sogenannten *Master* vor, der die zu bearbeitenden Aufgaben an andere Prozesse, die sogenannten *Slaves*, verteilt. Dieses Konzept eignet sich für Aufgaben, die aus einer Vielzahl gleicher Teilaufgaben besteht, die parallel ausgeführt werden können. Im Bereich der Evolutionären Algorithmen macht das Farmingkonzept Sinn, sobald die Berechnung der Fitnessfunktion die Ausführung des EA dominiert. In diesem Fall wird der Rahmen des EA einschließlich der genetischen Operatoren vom Master-Prozeß ausgeführt, während die Berechnung der Fitnessfunktion der einzelnen Individuen auf die Slaves ausgelagert wird.

In den Algorithmen 8.1 und 8.2 sind Fragmente des Master- und des Slave-Prozesses beschrieben. Der Master-Prozeß schickt zunächst Aufträge an alle Slaves und stellt damit sicher, daß alle Slaves beschäftigt sind. Anschließend wartet er auf das Resultat eines beliebigen Slaves, empfängt es und belegt den Slave mit einer neuen Aufgabe. Dieses Vorgehen gewährleistet, daß alle Slaves ständig beschäftigt sind. Der Slave-Prozeß besteht aus einer Schleife, in der der Slave einen Auftrag vom Master empfängt, diesen bearbeitet und das Resultat an den Master zurücksendet.

Algorithmus 8.1 *Master*

```
MASTER()
 1  tid_snd ← 0; sid_snd ← 0; tasks_done ← 0
 2  while sid_snd < SLAVES  AND  tid_snd < TASKS
 3      do SENDTASKTOSLAVE(sid_snd, tid_snd)
 4         sid_snd ← sid_snd + 1
 5         tid_snd ← tid_snd + 1
 6  while task_done < TASKS
 7      do RECEIVERESULTFROMSLAVE(&sid_rcv, &tid_rcv)
 8         tasks_done ← tasks_done + 1
 9         sid_snd ← sid_rcv
10         if tid_snd < TASKS
11           then SENDTASKTOSLAVE(sid_snd, tid_snd)
12                tid_snd ← tid_snd + 1
13  return
```

Algorithmus 8.2 *Slave*

```
SLAVE()
1  while true
2     do RECEIVETASKFROMMASTER(&tid, &data)
3        result ← TASK(data);
4        SENDRESULTTOMASTER(tid, result)
5  return
```

8.3 Parallele Umgebung

8.3.1 Hardware

Typische parallele oder verteilte Rechnersysteme bestehen häufig aus durch Busse oder Datenleitungen verbundene Einzelprozessorsysteme. Im einfachsten Fall bildet bereits eine Menge von PCs oder Workstations, die durch Ethernet miteinander kommunizieren, ein verteiltes MIMD-Rechensystem.

Verteilte SUN-Workstations. Als SUN20/612MP wird eine SUN-Workstation des Typs *SPARCstation 20 Modell 612MP* bezeichnet. Sie besteht aus zwei 60 MHz SuperSPARC Prozessoren mit jeweils einem MByte SuperCache. Die beiden Prozessoren teilen sich 64 MB Hauptspeicher. Die Kommunikation zwischen den Prozessoren verschiedener Workstations erfolgt über Ethernet-Leitungen.

Parsytec EXPLORER. Der EXPLORER der Aachener Firmal Parsytec besteht aus Knoten. Jeder Knoten besteht aus einem Prozessor des Typs PowerPC 601 mit 16 MB Hauptspeicher und einem T805 Transputer, der die Kommunikation zwischen benachbarten Knoten übernimmt. Die Peak-Leistung eines PowerPC 601 beträgt nach Herstellerangaben 77 SPECint92 und 93 SPECfp92. Die Kommunikationsleistung des T805 beträgt für jeden seiner vier Links 1,5 MByte/s. Für die Messungen standen 2 EXPLORER, also 16 Knoten, zur Verfügung.

IBM SP2. Die SP2 der Firma IBM besteht aus 33 Power2-Prozessoren mit jeweils 32 MB Hauptspeicher. Die Peak-Leistung eines Prozessors beträgt nach Herstellerangaben 109 SPECint92 und 202 SPECfp92. Für die Kommunikation zwischen den Knoten steht ein spezieller *High Performance Switch* zur Verfügung.

8.3.2 Kommunikationssoftware

Die Kommunikation zwischen verschiedenen Prozessen erfolgt über die Kommunikationssoftware *PVM (Parallel Virtual Machine)*, die für die oben vorgestellten Plattformen zur Verfügung steht.

PVM ist eine integrierte Menge von Hilfsprogrammen und Bibliotheken, die eine allgemeine, flexible, heterogene Parallelrechnerumgebung auf miteinander vernetzten Rechnern unterschiedlicher Architektur emuliert.

Unter PVM werden verschiedenartige Rechner zu einem sogenannten *Host-Pool* zusammengefaßt. Die Rechner des Host-Pools werden von Anwendungsprogrammen als attributlose virtuelle Berechnungseinheiten oder als spezialisierte Hardware betrachtet. PVM arbeitet prozeßbasiert. Die Parallelitätseinheit ist ein Prozeß, der abwechselnd rechnet oder kommuniziert. Mehrere Prozesse können auf einen Prozessor abgebildet werden. Als Kommunikationsmodell verwendet PVM *message passing*. Für bestimmte Parallelrechner, wie die IBM SP2, existieren PVM-Versionen, die auf den eigenen Kommunikationsroutinen aufsetzen und die zugrundeliegende Hardware ausnutzen.

Eine ausführlichere Beschreibung von PVM ist in [GBD+94] zu finden.

8.4 Messungen

Die Grundlage der Zeitmessungen bildet das in Kapitel 7.1 eingeführte Optimierungsproblem der Mustererkennung. Damit werden für eine reale Anwendung Möglichkeiten und Nutzen der Parallelisierung, die sich durch die Verwendung von Evolutionären Algorithmen ergeben, demonstriert. Als Parallelisierungsmodell wird das in Abschnitt 8.2.2 vorgestellte Farmingkonzept verwendet.

Die Auswertung der Zielfunktion stellt in der Optimierung realer Probleme häufig die rechenaufwendigste Phase dar. Falls die Evaluierung der Population die restlichen Phasen eines Evolutionären Algorithmen bezüglich der Rechenzeit bei weitem übertrifft, besteht die einfachste und effektivste Parallelisierung in der parallelen Evaluierung der Individuen einer Population durch ein Farming-Modell.

Die Leistungsmaße der Parallelisierung eines aus 64 Individuen bestehenden cBGA, der das in Kapitel 7.1 beschriebene Mustererkennungsproblem mit 561 Objektvariablen löst, ist in Tabelle 8.1 angegeben. In den meisten Fällen wird eine annähernd lineare Beschleunigung erzielt. Mit zunehmender Anzahl von Prozessoren sinkt die Beschleunigung, da das geringer werdende Verhältnis von Individuen zu Prozessoren die gleichmäßige Belegung der Prozessoren erschwert. Effizienzwerte, die größer als Eins sind, sind auf Ungenauigkeiten in der Zeitnahme zurückzuführen. Der EXPLORER weist geringe Effizienzeinbrüche auf, falls der Master-Prozeß und einer der Slave-Prozesse auf einem Prozessor ausgeführt werden. Eine SP2 mit 33 Prozessoren erreicht relativ zu einem SUN20-Prozessor eine Beschleunigung von fast 200.

Die Zeiten, die für die einzelnen Phasen einer typischen BGA-Simulation benötigt werden, sind in Tabelle 8.2 angegegeben. Die erste Spalte (GEN) gibt die Zeit, die für eine vollständige Generation benötigt wird, an. Anhand dieser Werte wird deutlich, daß die Rechenzeit bei diesem Problem fast vollständig für die Evaluierung der Population verbraucht wird.

Maschine	p	S	FE/S	$\mathcal{S}$	$\mathcal{E}$	$\mathcal{E}_{\text{SUN20}}$
SUN20	1	1	0.640	1.000	1.000	1.000
	2	2	1.412	2.206	1.103	2.206
	4	4	2.667	4.167	1.042	4.167
EXPLORER	2	1	1.423	1.000	1.000	2.223
	3	2	2.987	2.099	1.050	4.667
	5	4	5.793	4.071	1.018	9.052
	8	8	9.409	6.612	0.825	10.331
	16	16	16.564	11.640	0.728	25.881
SP2	2	1	4.812	1.000	1.000	7.519
	3	2	9.552	1.985	0.993	14.925
	5	4	19.048	3.958	0.990	29.763
	9	8	37.142	7.719	0.965	58.034
	17	16	72.113	14.986	0.937	112.677
	33	32	123.349	25.634	0.801	192.733

Tabelle 8.1: Leistungsmaße der Parallelisierung eines BGA, der ein Mustererkennungsproblem löst ($n = 561$). **p**: Anzahl der Prozessoren, **S**: Anzahl der Slave-Prozesse, **FE/s**: Funktionsauswertungen pro Sekunde, $\mathcal{S}$: Beschleunigung, $\mathcal{E}$: Effizienz, $\mathcal{E}_{SUN20}$: Geschwindigkeitsverhältnis zur SUN20.

Maschine	GEN	EVAL	GO	MAT	SEL
SUN20	83.52	83.51	0.0674	0.00070	0.00046
EXPLORER	45.15	45.02	0.0865	0.00021	0.00056
SP2	13.28	13.33	0.0254	0.00012	0.00018

Tabelle 8.2: Zeiten (in ms) der einzelnen Phasen des BGA unter PeGAsuS für die Mustererkennung ($\lambda = 64, n = 561$). GEN - Generationsschleife, EVAL - Evaluierung, GO - Genetische Operatoren, MAT - Paarbildung, SEL - Selektion

Kapitel 9

Zusammenfassung

In dieser Arbeit wurden Wettbewerbsinteraktionen zwischen Populationen, wie sie in der natürlichen Evolution beobachtet werden, auf Evolutionäre Algorithmen übertragen. Analog zum in der Natur vorkommenden Wettbewerb zwischen verschiedenen Arten konkurrieren in dem hier entwickelten Wettbewerbskonzept Evolutionäre Algorithmen mit unterschiedlichen Suchstrategien um eine beschränkte Ressource. Dieses Wettbewerbskonzept ermöglicht die Integration ausgetesteter, sich ergänzender Suchstrategien in einen aus mehreren Populationen bestehenden Evolutionären Algorithmus. Durch die Kopplung der Suchstrategien stellen sich Synergieeffekte ein, die sowohl zu Effizienzsteigerungen als auch zu einer Erhöhung der Robustheit gegenüber den einzelnen Suchverfahren führen. Im Gegensatz zu dem Inselmodell, das aus mehreren Populationen fester Größe besteht, werden die einzelnen Populationsgrößen im Wettbewerbsmodell dynamisch an die aktuelle Suchsituation angepaßt. Auf diese Weise wird versucht, die zur Verfügung stehende Rechenzeit optimal auf die verwendeten Suchstrategien aufzuteilen.

Zur Steuerung der Populationsgröße wurden Konzepte aus der Populationsdynamik verwendet. Die Anpassung der Populationsgrößen erfolgt über eine konstante Ressource, die während der Suche dynamisch auf die Populationen verteilt wird. Die Aufteilung erfolgt selbstorganisierend mit Hilfe eines Gütekriteriums, das den Beitrag der Suchstrategie der entsprechenden Population bewertet. Die Größe einer Population ergibt sich dann aus dem ihr zur Verfügung stehendem Anteil an der Ressource.

Aus diesem Wettbewerbskonzept wurde zunächst ein Modell entwickelt, das mit einer konstanten Gesamtpopulationsgröße und festen Suchstrategien arbeitet. Dieses Basis-Wettbewerbsmodell wurde dann durch die Einführung von Verbrauchsfaktoren erweitert, die den Pro-Kopf-Verbrauch der Ressource modellieren. Dieser Verbrauchsfaktor ist für alle Individuen einer Population gleich und legt den funktionalen Zusammenhang zwischen Ressource und Populationsgröße fest. Mit den Verbrauchsfaktoren kann ein Wettbewerb zwischen Populationen, deren Suchstrategien unterschiedliche Anforderungen an die Populationsgröße stellen, modelliert werden. Eine Umverteilung der Ressource während der Suche kann zu einer Veränderung der Gesamtpopulationsgröße führen. Auf diese Weise wird die Gesamtpopulationsgröße an die Erfordernisse der Suche angepaßt. Die zweite Erweiterung des Basis-Wettbewerbsmodells besteht in der dynamischen Anpassung der Suchstrategien. Hierbei wird die Entwicklung der konkurrierenden Populationen beobachtet, und ihre Strategien bei Bedarf gezielt modifiziert.

Für die Umsetzung der Wettbewerbsmodelle in Optimierungsverfahren wurden spezielle Gütekriterien zur Bewertung der Populationen sowie Verfahren zur Ressourcenauf-

teilung und Populationsgrößenanpassung entwickelt. Ein konkreter wettbewerbsgesteuerter Evolutionärer Algorithmus wird durch die Wahl des Wettbewerbsmodells und der Suchstrategien für die einzelnen Populationen spezifiziert. Für die Untersuchung der drei Wettbewerbsmodelle wurden als Suchstrategien die Evolutionsstrategien und der BGA ausgewählt.

Die Entwicklung des Breeder Genetic Algorithm (BGA) ist ein weiterer Schwerpunkt dieser Arbeit. Der BGA unterscheidet sich von anderen Evolutionären Algorithmen im wesentlichen durch seinen Mutationsoperator. Gegenüber Mutationsoperatoren, die mit normal- oder gleichverteilten Verteilungen arbeiten, verwendet der BGA eine Mutationsverteilung, bei der auf eine Regelung des Schrittweitenparameters verzichtet werden kann. Da die BGA-Mutation nur einen Objektparameter verändert, werden ausschließlich Schritte entlang der Achsen des zugrunde liegenden Koordinatensystems gemacht. Um Zielfunktionen mit stark korrelierten Objektparametern effizient bearbeiten zu können, wurde die BGA-Mutation um eine Hauptachsentransformation erweitert. Diese erweiterte BGA-Mutation mutiert entlang der Achsen eines Koordinatensystems, das durch eine Hauptachsentransformation an die Lage der selektierten Individuen angepaßt wird. Ähnlich zu den Strategieparametern der Evolutionsstrategien wird mit diesem Verfahren ein internes Modell der Zielfunktion gebildet. Im Unterschied zu den Evolutionsstrategien wird das interne Modell jedoch analytisch und nicht durch die Evolution selbst bestimmt.

Die Leistungsfähigkeit des Wettbewerbskonzepts wurde empirisch durch Simulationsläufe nachgewiesen. Zur Einordnung der Resultate wurden Evolutionäre Algorithmen und Standardverfahren aus der Literatur auf eine repräsentative Menge von Testfunktionen angewendet. Ein Vergleich der Suchverfahren bezüglich ihres Konvergenz- und Skalierungsverhaltens zeigt, daß die Wettbewerbsmodelle in fast allen Fällen eine Steigerung der Effizienz und Robustheit gegenüber einem Evolutionären Algorithmus mit nur einer Suchstrategie erreichten. Bei den komplexeren hochdimensionalen und multimodalen Benchmarkfunktionen erwiesen sich die Evolutionären Algorithmen gegenüber den lokalen Standardsuchverfahren als überlegen. Besonders bei hochdimensionalen Zielfunktionen traten bei den analytischen Suchverfahren numerische Instabilitäten auf.

Der Einsatzmöglichkeit des Wettbewerbskonzepts für reale Anwendungen wurde an einem Beispiel aus der Mustererkennung demonstriert. Die Erkennung handgeschriebener Zeichen wurde auf eine aus mehreren hundert Objektparametern bestehende Zielfunktion abgebildet und mit einem BGA mit Wettbewerbsmodell bearbeitet. Dieser Ansatz übertraf hinsichtlich des Klassifikationsergebnis alle zum Vergleich herangezogenen Verfahren wie den Quadratmittel-Ansatz, verschiedene Nearest-Neigbor-Verfahren und ein Neuronales Netz. An diesem Mustererkennungsproblem mit seiner rechenaufwendigen Fitnessfunktion wurde außerdem gezeigt, wie populationsbasierte Suchverfahren durch die parallele Auswertung der Individuen beschleunigt werden können. Bei Simulationsläufen auf verschiedenen Parallelrechnern konnte eine fast lineare Beschleunigung erzielt werden.

Anhang A

Daten für Fletcher und Powells Funktion

Die in Abschnitt 6.2.6 definierte Funktion von Fletcher und Powell wird durch die beiden $n \times n$ Matrizen **A** und **B** und den Vektor $\vec{\alpha}$ beschrieben.

Im folgenden sind die Werte, die Bäck zur Untersuchung Evolutionärer Algorithmen verwendete, aufgeführt [[Bäc96]]. Mit diesen Werten ist die Funktion bis zur Dimension $n = 30$ skalierbar. Für $n \leq 30$ wird die obere linke Teilmatrix verwendet. Der Vektor $\vec{\alpha}$ ist gegeben als:

$$
\begin{array}{lll}
\alpha & = & (\\
 & & \\
 & & \\
 & & \\
 & & \\
 & & \\
 & & \\
 & &)
\end{array}
\begin{array}{rrrrr}
0.4359340 & 0.5505950 & -1.2834100 & -0.0734284 & -2.6051900 \\
2.1041000 & 1.8675400 & -3.0127500 & 0.8628350 & 0.0666833 \\
2.3361100 & -0.6581490 & -3.1124300 & -3.0755600 & 0.8418540 \\
-0.6925490 & 3.0628400 & -0.9173990 & 0.2111350 & -1.4526100 \\
2.4824400 & 2.0083400 & 0.9061900 & -0.1087510 & 0.6348730 \\
1.4588100 & 1.2409200 & 2.3031100 & -2.3116000 & -2.1476100
\end{array}
$$

Die Matrizen **A** und **B** sind auf den nachfolgenden Seiten aufgeführt. Aus Platzgründen werden die aus 30 Elementen bestehenden Zeilen der Matrizen in jeweils zwei aus 15 Einträgen bestehendene aufeinanderfolgenden Zeilen dargestellt.

$$a_{ij} = \begin{pmatrix}
-78 & 28 & 53 & -9 & 75 & -55 & -62 & 20 & -88 & 95 & -71 & -70 & 36 & -4 & -76 \\
38 & 13 & -30 & 77 & 61 & -25 & -65 & 37 & 22 & -88 & 24 & -45 & -61 & 30 & 0 \\
-13 & -50 & -98 & 20 & -40 & -62 & -68 & -6 & 16 & 58 & -98 & 2 & 10 & 67 & 0 \\
-75 & 0 & -22 & -60 & -88 & -90 & -18 & 59 & 61 & 2 & -95 & 69 & -15 & -28 & 20 \\
27 & 73 & 63 & 81 & 15 & 92 & 0 & 59 & 63 & 62 & 97 & -22 & -49 & 66 & -93 \\
-43 & 71 & -97 & -81 & 74 & 37 & 80 & -51 & -93 & -85 & 99 & -30 & -8 & 76 & 14 \\
59 & -23 & 39 & 98 & 11 & 26 & -25 & -49 & -77 & -36 & 66 & -43 & 30 & 26 & -52 \\
80 & -60 & 96 & 47 & 96 & -87 & -5 & -28 & 43 & 24 & -28 & 81 & 19 & -68 & 69 \\
-44 & -58 & -49 & -9 & -22 & 5 & -27 & 35 & 95 & -70 & 60 & 71 & -83 & 23 & 77 \\
78 & 13 & 68 & -8 & 71 & 39 & -76 & -5 & 25 & 1 & 6 & -76 & -74 & -5 & 92 \\
0 & 81 & 30 & 1 & -52 & -32 & -56 & -70 & 31 & -53 & 75 & -12 & 41 & 14 & -51 \\
10 & 22 & -84 & 10 & 17 & 46 & 90 & 99 & -15 & -37 & 89 & -67 & 64 & -55 & -23 \\
16 & 49 & -39 & -66 & 42 & -63 & 47 & 80 & -58 & 17 & -56 & 26 & -92 & -54 & -25 \\
-8 & -52 & -66 & 52 & -73 & 62 & -69 & -31 & 5 & 73 & 30 & 17 & -43 & -17 & -94 \\
90 & -50 & -17 & -11 & 69 & 92 & -49 & 2 & -2 & 35 & 64 & -38 & -6 & 92 & -98 \\
20 & 31 & -80 & 95 & 52 & 91 & 76 & -67 & 53 & 32 & 76 & -83 & 47 & -74 & 66 \\
-6 & 2 & -32 & 1 & 51 & -17 & 2 & 86 & -12 & -77 & 95 & 37 & -68 & 23 & -57 \\
99 & -97 & -98 & 69 & 94 & 40 & 62 & 25 & 66 & 27 & 39 & 24 & -37 & -5 & -97 \\
65 & -7 & 19 & 95 & 9 & -44 & 10 & 96 & -85 & 13 & -13 & -17 & -25 & 95 & 37 \\
68 & -40 & -76 & 31 & -84 & 69 & -45 & -62 & -38 & 60 & 24 & -35 & -2 & 83 & 0 \\
-22 & -97 & -89 & 81 & 4 & 45 & -6 & -6 & 52 & 68 & -34 & 21 & 47 & 55 & 52 \\
92 & 92 & 65 & -56 & -90 & -93 & -25 & -53 & 21 & -10 & 38 & -90 & -69 & -74 & -78 \\
54 & 59 & 61 & -54 & -74 & 51 & 67 & 77 & -58 & -61 & 29 & 36 & -1 & -52 & 52 \\
-31 & -34 & -61 & -60 & -52 & 47 & -77 & -88 & 70 & 89 & -26 & 66 & 28 & 7 & 57 \\
50 & -42 & -71 & -4 & 73 & 66 & -93 & -82 & 15 & 13 & -95 & -86 & 4 & -48 & -49 \\
-10 & 41 & -41 & 10 & 2 & -4 & -73 & -18 & -51 & 21 & 69 & 12 & -91 & -30 & -5 \\
-21 & 11 & -45 & -76 & 68 & 96 & -63 & 15 & 70 & -35 & 39 & -1 & -94 & 44 & -91 \\
27 & 32 & 20 & -9 & 11 & -77 & -43 & -63 & 77 & 98 & 67 & -81 & 37 & -81 & -6 \\
-67 & 6 & 74 & -2 & -42 & 36 & -92 & 84 & 91 & -43 & -23 & -3 & 78 & -4 & -32 \\
-48 & 30 & 38 & -29 & -7 & -20 & 64 & 51 & 6 & 56 & -73 & 22 & 86 & -67 & -43 \\
-30 & -68 & 92 & 2 & 76 & -22 & -72 & -97 & 86 & 50 & -11 & 80 & -82 & 54 & 26 \\
68 & -94 & 90 & -8 & -48 & -87 & 20 & 80 & 84 & 70 & -78 & 95 & -23 & -30 & 60 \\
-67 & -45 & 68 & -20 & 80 & -25 & -26 & -21 & -35 & 60 & 86 & 99 & -17 & 30 & -55 \\
-77 & 50 & -91 & 31 & -49 & -52 & -42 & -70 & -2 & -65 & -7 & -43 & 45 & -53 & 89 \\
-50 & 8 & -31 & -62 & -7 & -5 & -60 & -26 & -91 & 42 & 46 & 90 & -37 & -49 & -2 \\
-14 & 50 & -56 & 93 & -43 & -98 & -17 & -69 & -76 & -61 & 47 & 36 & -76 & -20 & 87 \\
-99 & 21 & 29 & -59 & -28 & 2 & -80 & 38 & -98 & -41 & 4 & -49 & 0 & -52 & 74 \\
-29 & 99 & 53 & -4 & 56 & -87 & 43 & -44 & -30 & -7 & -97 & -39 & -93 & 18 & -5 \\
97 & 96 & 12 & -62 & 89 & 34 & 68 & -76 & -24 & 0 & -95 & -85 & -2 & 54 & -34 \\
94 & 20 & 17 & 21 & 71 & 13 & 4 & 10 & 98 & 29 & 16 & -88 & -47 & 22 & 22 \\
-76 & 29 & -76 & 21 & 87 & -41 & 64 & -40 & -76 & -93 & -58 & 5 & 87 & -31 & 48 \\
-57 & 78 & -56 & -17 & 93 & -93 & -69 & 1 & -4 & -15 & -95 & 56 & 50 & -30 & -64 \\
-26 & 34 & 72 & 75 & -86 & 30 & -17 & 4 & -28 & -69 & 54 & -92 & -89 & 2 & 65 \\
-93 & 71 & -36 & -66 & 38 & -65 & 57 & -51 & 55 & -96 & -83 & 84 & -37 & -73 & 21 \\
2 & -69 & 99 & -64 & -10 & -95 & -29 & 18 & 89 & -17 & 11 & -14 & -96 & 62 & -9 \\
-53 & 4 & 67 & -45 & 59 & -13 & -8 & -88 & 85 & 64 & 91 & 43 & 55 & -99 & -96 \\
-44 & 15 & 12 & -2 & 54 & -61 & 52 & -36 & -27 & -61 & 32 & 60 & -88 & 54 & -47 \\
-10 & -93 & -42 & -49 & -23 & 90 & -33 & -40 & -43 & 56 & 48 & 74 & -50 & 26 & 52 \\
-69 & 92 & -77 & 27 & -14 & -91 & -74 & 80 & -66 & -33 & 30 & -44 & 69 & -87 & -53 \\
3 & 31 & -61 & -66 & 84 & 90 & 99 & 6 & -93 & -74 & -54 & -78 & 4 & -77 & -53 \\
93 & 44 & -14 & 81 & 78 & 75 & -60 & -2 & 54 & 66 & 60 & -54 & 86 & 77 & -43 \\
-22 & -51 & 3 & 5 & -6 & 1 & 62 & -68 & -78 & 42 & 47 & 34 & -30 & 88 & -31 \\
5 & 70 & 11 & -26 & -30 & 28 & 76 & 91 & -10 & -89 & 90 & 27 & 42 & 49 & -95 \\
63 & 83 & 45 & -90 & -7 & 42 & -72 & 13 & 92 & 89 & -92 & -31 & 64 & 64 & 82 \\
29 & 18 & 88 & -66 & -20 & -54 & 59 & -23 & 7 & -18 & -30 & -15 & 82 & -60 & -59 \\
-6 & -35 & 11 & 71 & -19 & 93 & 59 & -67 & 45 & -33 & 76 & -67 & -23 & -25 & -26 \\
76 & -48 & 42 & 27 & -15 & 24 & -44 & -3 & 12 & 80 & 26 & 14 & -35 & 97 & -71 \\
26 & -9 & -54 & -36 & 87 & -87 & 14 & -6 & -87 & 89 & -14 & -12 & 62 & 39 & 93 \\
53 & -39 & 38 & -54 & 84 & 56 & -51 & -37 & 98 & -18 & -3 & 69 & 50 & -3 & -38 \\
-20 & 48 & -33 & -40 & 83 & 38 & -30 & -95 & 0 & 38 & 57 & 34 & -63 & -46 & -57
\end{pmatrix}$$

$$b_{ij} = \begin{pmatrix}
97 & -25 & -78 & -27 & 85 & 0 & -55 & 68 & 61 & -57 & 88 & 2 & -9 & -44 & -52 \\
-11 & -72 & 10 & -33 & -19 & 56 & 68 & -55 & -54 & -10 & 1 & -65 & 14 & -19 & 64 \\
30 & 25 & -32 & -1 & 15 & -5 & 28 & 92 & 99 & 42 & -83 & 21 & 10 & 82 & -45 \\
76 & 75 & 46 & 58 & 74 & 75 & 12 & -86 & -11 & 0 & -29 & -17 & 15 & -58 & 59 \\
87 & -31 & -92 & -47 & 25 & -68 & 76 & -43 & -87 & 92 & -58 & -25 & -88 & -45 & 89 \\
82 & -40 & 85 & 96 & 78 & 43 & -61 & -99 & -22 & -57 & -37 & 65 & -26 & -65 & -20 \\
31 & -37 & 6 & 27 & -1 & 3 & 76 & -70 & 73 & -15 & 0 & -97 & 71 & -66 & 7 \\
-43 & 63 & -62 & -44 & -82 & -62 & 38 & -39 & 88 & 60 & 67 & 90 & 97 & -96 & 51 \\
13 & -19 & 85 & -59 & -18 & 63 & -62 & 39 & 77 & 5 & 6 & -38 & -88 & 3 & -63 \\
0 & -6 & 66 & -32 & -91 & -92 & -56 & -24 & 48 & -34 & -10 & 95 & 57 & 91 & 64 \\
64 & 82 & -5 & 5 & -25 & -69 & 70 & 21 & 16 & 27 & 40 & -37 & -31 & 45 & -28 \\
35 & 58 & 23 & 50 & 3 & 83 & -41 & 90 & -63 & 36 & -51 & 6 & -66 & 95 & -15 \\
41 & -61 & 60 & 69 & -16 & 24 & 22 & 43 & 2 & 54 & 5 & 54 & 74 & -88 & -99 \\
-26 & 12 & 85 & 45 & 88 & -64 & 79 & 8 & -44 & 9 & -96 & -95 & -82 & 2 & 56 \\
38 & 81 & 5 & 30 & -45 & 0 & 2 & -72 & -17 & 53 & -95 & -93 & 22 & 38 & -4 \\
50 & -39 & -30 & 43 & -65 & -33 & -72 & -99 & -78 & 45 & -82 & -22 & 40 & 68 & -41 \\
-57 & 77 & 83 & 2 & -61 & 27 & -3 & -3 & -76 & 54 & 86 & 73 & -25 & -65 & 41 \\
50 & -33 & -42 & 3 & -40 & 96 & -3 & 92 & 67 & 37 & 68 & -45 & 0 & 99 & 13 \\
13 & 31 & 78 & -12 & 17 & -69 & 67 & -70 & 26 & -23 & 29 & 90 & 40 & -69 & 53 \\
10 & 32 & 86 & 81 & -17 & -26 & -61 & -85 & -75 & 70 & -12 & 35 & 22 & -64 & -28 \\
72 & -18 & -52 & -66 & -73 & -85 & -23 & -50 & 64 & 70 & 17 & 38 & 5 & 87 & -44 \\
-43 & -37 & 51 & -40 & -15 & 80 & 93 & 60 & -52 & 13 & 0 & -63 & 98 & -42 & 39 \\
86 & -84 & -3 & 49 & 54 & -27 & 93 & 42 & -38 & 37 & 84 & -7 & -33 & 20 & -59 \\
41 & -98 & -49 & -23 & -89 & -2 & -44 & 25 & 95 & 19 & -49 & -43 & 54 & -3 & -3 \\
95 & 8 & 3 & -21 & 20 & -10 & -46 & -91 & 72 & 53 & -56 & 1 & 15 & -36 & 20 \\
93 & 29 & 32 & 7 & 67 & -89 & -64 & 93 & 88 & -8 & -4 & 9 & -78 & -96 & 30 \\
-25 & 1 & 41 & -7 & 28 & 60 & 1 & -82 & 33 & -47 & 79 & -42 & 10 & 63 & 99 \\
81 & 6 & 89 & 11 & -50 & 26 & 51 & 47 & 80 & 14 & -96 & -47 & -98 & 78 & -43 \\
9 & -39 & -93 & -90 & 4 & 23 & 31 & -29 & -9 & 93 & 0 & -86 & -61 & -27 & -86 \\
-64 & -96 & -61 & 43 & -76 & 10 & 65 & -89 & -85 & 95 & -6 & 96 & 2 & 56 & -38 \\
94 & -98 & 25 & 40 & -88 & -77 & -84 & -58 & -75 & -29 & -88 & -42 & 95 & 28 & 2 \\
31 & -50 & 66 & 79 & 47 & -55 & -60 & -6 & -30 & 36 & 97 & -43 & -59 & -39 & -46 \\
1 & 18 & -74 & 44 & -50 & -34 & -56 & -49 & 27 & -73 & -87 & -51 & -91 & 40 & -9 \\
31 & -78 & 66 & -63 & 25 & 69 & -18 & 46 & 87 & 69 & -70 & 20 & 40 & 73 & 72 \\
95 & -33 & -42 & -62 & -20 & -18 & -33 & -33 & -73 & 17 & 65 & 27 & -27 & 63 & 0 \\
60 & 5 & 42 & 47 & 75 & -12 & 90 & -68 & 10 & 33 & -34 & 7 & -70 & -43 & 74 \\
86 & 33 & 3 & -23 & 17 & -75 & 14 & -40 & -48 & 34 & 31 & -51 & -77 & -79 & -98 \\
94 & -44 & 30 & -47 & -28 & -28 & -79 & 56 & -47 & -32 & -33 & -93 & 14 & 28 & 83 \\
80 & -4 & -28 & -77 & 35 & -74 & -62 & -76 & -78 & -72 & 84 & -96 & -12 & -63 & 40 \\
62 & -67 & -35 & 22 & -72 & 9 & 14 & -57 & -71 & 43 & -55 & 55 & 9 & -18 & -93 \\
61 & 89 & -28 & 72 & 26 & 30 & 74 & -34 & -7 & 17 & 34 & 67 & -23 & 98 & -88 \\
-14 & -6 & 36 & -55 & -38 & 14 & 28 & -42 & -69 & 46 & 54 & -72 & 9 & 10 & -13 \\
-48 & 18 & -73 & 32 & -9 & -52 & -62 & 82 & -5 & -34 & -28 & 40 & 41 & -66 & -87 \\
-73 & 15 & -19 & -3 & 85 & -16 & 19 & 35 & 94 & -20 & -72 & -54 & 26 & 27 & -85 \\
-82 & 37 & -18 & 86 & 30 & -65 & -65 & -18 & 89 & -13 & -59 & 0 & 23 & -80 & -55 \\
-48 & -73 & 70 & -2 & -51 & -10 & -96 & -80 & 39 & -2 & 41 & 68 & -61 & -68 & 20 \\
-94 & 80 & -52 & 50 & 38 & 1 & 4 & 50 & -34 & 19 & -41 & -8 & -9 & -94 & 9 \\
-82 & 23 & -14 & 45 & 41 & 57 & 69 & 7 & 67 & 25 & -40 & -22 & 53 & 93 & 0 \\
-83 & -91 & 2 & -43 & 90 & -8 & -44 & -42 & 82 & 71 & 34 & -54 & 48 & -85 & -42 \\
-28 & 6 & -19 & 18 & -14 & 0 & -58 & -43 & 3 & 59 & -56 & 78 & 52 & -72 & -63 \\
5 & 23 & 84 & 46 & -65 & 47 & -37 & -95 & 28 & -59 & 10 & 13 & 85 & -48 & -95 \\
-98 & 9 & -67 & -19 & 29 & 35 & 62 & 73 & 30 & 51 & -21 & 22 & -53 & 90 & 47 \\
-52 & -42 & -81 & 7 & -85 & 1 & -30 & 75 & 95 & 24 & -13 & -27 & 15 & 99 & 79 \\
47 & -1 & 89 & 61 & 29 & 96 & -38 & -28 & -26 & -14 & -8 & -39 & 18 & -79 & 91 \\
11 & -53 & -82 & 77 & 96 & -72 & 58 & -82 & -93 & 26 & -63 & -72 & -29 & 56 & 84 \\
60 & 97 & -84 & -89 & -63 & 14 & -59 & -50 & 10 & -8 & 18 & 97 & 52 & 90 & 81 \\
-47 & 67 & -53 & -32 & -5 & 45 & -86 & -43 & 51 & -87 & -82 & 46 & 64 & -45 & 3 \\
-35 & 59 & 55 & 36 & -43 & -83 & 49 & -3 & -83 & 57 & 71 & 62 & 68 & 3 & -83 \\
80 & 67 & -14 & 42 & -70 & -10 & -66 & 73 & -7 & 92 & 93 & 93 & 78 & -62 & -74 \\
-51 & 16 & 11 & -12 & 15 & -95 & -46 & 13 & -74 & -82 & -38 & 19 & -75 & -39 & 33
\end{pmatrix}$$

Anhang B

Notation

Die in dieser Arbeit verwendeten mathematischen Symbole und Abkürzungen sind in den folgenden Tabellen zusammengefaßt.

Mathematische Symbole

Symbol	Bezeichnung	Seite
$\mathbb{A}$	Anpassung der Populationsgrößen	99
$\vec{a}$	Individuum aus I	16
$\hat{\vec{a}}$	situiertes Individuum	17
α	Verbrauchsfaktor in Wettbewerbsmodellen	103
$\vec{\alpha}$	Vektor von Rotationswinkeln in Evolutionsstrategien	32
$\mathbb{B}$	Binärraum {0,1}	
$\mathbf{B}(k,\rho)$	BGA-Mutationsverteilung	48
$\hat{\mathbf{B}}(k,\rho)$	symmetrische BGA-Mutationsverteilung	48
β	Mutationsparameter für die Rotationswinkel in Evolutionsstrategien	36
$\mathbf{C}(\sigma)$	Cauchyverteilung	
$\mathbb{C}$	Kodierung	15
$\gamma(.)$	Standard-Graycode-Konvertierung	39
$\mathbf{E}(\lambda)$	Exponentialverteilung	
ϵ_a	absolute minimale Standardabweichung in Evolutionsstrategien	37
ϵ_b	relative minimale Standardabweichung in Evolutionsstrategien	37
$\delta(\vec{x})$	Abstand des Suchpunktes vom Optimum	71

Tabelle B.1: Mathematische Symbole

Mathematische Symbole (Fortsetzung)

Symbol	Bezeichnung	Seite
η	Wettbewerbsintervall	99
$\mathbb{F}$	Fitnessfunktion	15
f^*	Funktionswert des globales Optimums	70
$\mathbb{G}$	Generationsschleife	15
Θ_x	Parameter des Operators x	15
θ	Migrationsintervall	105
ϑ	Remontierungsrate der Schnittselektion	22
$\mathbb{I}$	Initialisierung	15
I	Raum der Individuen	15
K	Kapazität	102
$\mathbb{K}$	Migrationsverfahren	19
κ	Anzahl der Populationen	19
k	Präzisionsparameter der BGA-Mutationsverteilung	48
$\mathbb{L}$	lokales Suchverfahren	20
λ	Anzahl der Nachkommen	15
$\mathbb{M}$	Mutation	15
μ	Anzahl der selektierten Individuen	15
$\mathbf{N}(0,1)$	Normalverteilung mit Mittelwert 0 und Standardabweichung 1	
n	Dimension des Suchraums	
n_σ	Anzahl der Standardabweichung in Evolutionsstrategien	36
n_α	Anzahl der Rotationswinkel in Evolutionsstrategien	36
φ	Fortschrittsgeschwindigkeit	75
$\mathbb{P}$	Paarbildung	15
$P(t)$	Population in Generation t	16
$\hat{P}(t)$	situierte Population in Generation t	17
$P^{\mathbb{S}}(t)$	Menge der selektierten Individuen in Generation t	16
$p_{\mathbb{S}}$	Selektionswahrscheinlichkeit	23
$\mathbb{Q}$	Qualität einer Population	99
R	Selektionserfolg	101
$\mathbb{R}$	Rekombination	15
$\mathbb{R}$	reelle Zahlen	
ρ	Anzahl der Eltern pro Individuum	15
ρ	Schrittweitenparameter der BGA-Mutationsverteilung	48

Tabelle B.2: Mathematische Symbole

Mathematische Symbole (Fortsetzung)

Symbol	Bezeichnung	Seite
ϱ	Anzahl der Eltern bei der Rekombination	35
s	Schätzwert der Standardabweichung	60
$s_{\bar{x}}$	Schätzwert des Standardfehlers	60
$\mathcal{S}$	Suchstrategie	87
$\vec{\sigma}$	Strategieparametervektor in Evolutionsstrategien	32
$\mathbf{T}(a)$	gleichschenklige Dreiecksverteilung	
$\mathbb{T}$	Terminierung	15
τ_0	globaler Mutationsparameter in Evolutionsstrategien	36
τ	individueller Mutationsparameter in Evolutionsstrategien	36
$\mathbf{U}([a,b])$	Gleichverteilung im Intervall $[a,b]$	
$\mathbf{U}(a,b)$	diskrete Gleichverteilung aus der Menge a,b	
$\mathbb{V}$	Vefahren zur Modifikation von Strategien	99
$\mathbb{W}$	Wettbewerbsverfahren	99
$\bar{x}$	Schätzwert des Erwartungswertes	59
$\vec{x}^*$	globales Optimum	70
$\mathbb{Z}$	ganze Zahlen	

Tabelle B.3: Mathematische Symbole

EA-Objekte		
EA-Objekt	Bezeichnung	Seite
Populationen		
$\mathsf{POP}^{\mathsf{LHC}}$	Iteriertes lokales Suchverfahren	133
$\mathsf{POP}^{\mathsf{ES1}}$	ES mit einem Strategieparameter	37
$\mathsf{POP}^{\mathsf{ES2}}$	ES mit n Strategieparametern	37
$\mathsf{POP}^{\mathsf{GA1}}$	GA mit rangbasierter Selektion	44
$\mathsf{POP}^{\mathsf{GA2}}$	GA mit fitnessbasierter Selektion	44
$\mathsf{POP}^{\mathsf{BGA}}$	BGA	54
$\mathsf{POP}^{\mathsf{BGA,BM}}$	BGA mit Mutation	54
$\mathsf{POP}^{\mathsf{BGA,PCA}}$	BGA mit Hauptachsentransformation	54
$\mathsf{POP}^{\mathsf{BGA,BLR}}$	BGA mit Line-Rekombination	54
$\mathsf{POP}^{\mathsf{BGA,FR}}$	BGA mit Fuzzy-Rekombination	55
Klassen		
$\mathsf{BGA}^{\mathsf{BM}}$	BGA mit Mutation	54
$\mathsf{BGA}^{\mathsf{PCA}}$	BGA mit Hauptachsentransformation	54
$\mathsf{BGA}^{\mathsf{BLR}}$	BGA mit Line-Rekombination	54
$\mathsf{BGA}^{\mathsf{FR}}$	BGA mit Fuzzy-Rekombination	55
ES1	ES mit einem Strategieparameter	37
ES2	ES mit n Strategieparametern	37
ES3	ES mit Rotationswinkeln	38
GA1	GA mit rangbasierter Selektion	44
Instanzen		
bga	Standard-BGA	135
bga-pca	BGA mit Hauptachsentransformation	135
bga-1+1	Mutations-Selektions-Verfahren mit BGA-Mutation	135
cbga1	cBGA mit unterschiedlichen Schrittweitenparametern	135
cbga2	cBGA mit unterschiedlichen Verbrauchsfaktoren	136
cbga3	cBGA mit Strategiemodifikation	136
ces	cES mit unterschiedlicher Anzahl von Strategieparametern	136
dhc	Iteriertes Dynamic-Hillclimbing-Verfahren	133
dhs	Iteriertes Downhill-Simplex-Verfahren	133
es1	ES mit einem Strategieparameter	134
es2	ES mit n Strategieparametern	134

Tabelle B.4: EA-Objekte

EA-Objekte (Fortsetzung)		
Symbol	Bezeichnung	Seite
ga	GA mit rangbasierter Selektion	134
pow	Iteriertes Powell-Verfahren	133
qn	Iteriertes Quasi-Newton-Verfahren	133
sw	Iteriertes Solis-and-Wets-Verfahren	133

Tabelle B.5: EA-Objekte

EA-Operatoren		
Symbol	Bezeichnung	Seite
Fitnessfunktion $\mathbb{F}$		
$\mathbb{F}^{EA}$	EA-Fitnessfunktion	22
$\mathbb{F}^{ES}$	ES-Fitnessfunktion	33
$\mathbb{F}^{GA}$	GA-Fitnessfunktion	38
$\mathbb{F}^{BGA}$	BGA-Fitnessfunktion	45
Initialisierung $\mathbb{I}$		
$\mathbb{I}^{EA,U}$	Gleichverteilte Initialisierung	30
$\mathbb{I}^{EA,P}$	Punktuelle Initialisierung	30
Kodierung $\mathbb{C}$		
$\mathbb{C}^{EA}$	EA-Kodierung	21
$\mathbb{C}^{ES}$	ES-Kodierung	32
$\mathbb{C}^{GA}$	GA-Kodierung	38
$\mathbb{C}^{BGA}$	BGA-Kodierung	45
Mutation $\mathbb{M}$		
$\mathbb{M}^{EA}$	Mutationsoperator für reelle Kodierungen	27
$\mathbb{M}^{ES}$	ES-Mutation	36
$\mathbb{M}^{GA}$	GA-Mutation	43
$\mathbb{M}^{BGA,STD}$	Standard-BGA-Mutation	49
$\mathbb{M}^{BGA,MA}$	Mehrachsen-BGA-Mutation	49
$\mathbb{M}^{BGA,PCA}$	BGA-Mutation mit Hauptachsentransformation	49

Tabelle B.6: EA-Operatoren

EA-Operatoren (Fortsetzung)		
Symbol	Bezeichnung	Seite
Paarbildung $\mathbb{P}$		
$\mathbb{P}^{EA,R}$	Paarbildung (Ziehen mit Zurücklegen)	23
$\mathbb{P}^{EA,R*}$	Paarbildung (Ziehen ohne Zurücklegen innerhalb eines Tupels)	23
$\mathbb{P}^{EA,S}$	Paarbildung (Ziehen ohne Zurücklegen)	24
$\mathbb{P}^{ES}$	ES-Paarbildung	34
$\mathbb{P}^{GA}$	GA-Paarbildung	42
$\mathbb{P}^{BGA}$	BGA-Paarbildung	46
Rekombination $\mathbb{R}$		
$\mathbb{R}^{EA}$	Rekombinationsoperator für reelle Kodierungen	25
$\mathbb{R}^{ES}$	ES-Rekombination	34
$\mathbb{R}^{GA,1P}$	GA-One-Point-Crossover	42
$\mathbb{R}^{GA,mP}$	GA-Multi-Point-Crossover	43
$\mathbb{R}^{GA,U}$	Uniform-Crossover	43
$\mathbb{R}^{BGA,F}$	BGA-Fuzzy-Rekombination	46
$\mathbb{R}^{BGA,L}$	BGA-Line-Rekombination	46
Selektion $\mathbb{S}$		
$\mathbb{S}^{EA,TR}$	Schnittselektion	22
$\mathbb{S}^{cEA,TR}$	Schnittselektion für Wettbewerbsmodelle	107
$\mathbb{S}^{EA,TO}$	Turnierselektion	22
$\mathbb{S}^{EA,PR}$	Proportionale Selektion	23
$\mathbb{S}^{ES,C}$	ES-Comma-Selektion	33
$\mathbb{S}^{ES,P}$	ES-Plus-Selektion	33
$\mathbb{S}^{BGA}$	BGA-Schnittselektion	45
$\mathbb{S}^{GA,R}$	Rangbasierte proportionale Selektion	42
$\mathbb{S}^{GA,F}$	Fitnessbasierte proportionale Selektion	40
Terminierung $\mathbb{T}$		
$\mathbb{T}^{EA,FIT}$	Terminierung	31
$\mathbb{T}^{EA,GEN}$	Terminierung	31
$\mathbb{T}^{EA,IMP}$	Terminierung	31

Tabelle B.7: EA-Operatoren

EA-Operatoren (Fortsetzung)		
Symbol	Bezeichnung	Seite
Wettbewerbsverfahren		
$\mathbb{W}^{\mathrm{WTA}}$	WTA-Wettbewerbsverfahren	108
$\mathbb{Q}^{\mathrm{TOP}}$	Qualitätskriterium	101
$\mathbb{A}^{\mathrm{WTA}}$	Ressourcenzuweisungsverfahren	103
$\mathbb{K}^{\mathrm{B2A}}$	Migrationsverfahren	105
$\mathbb{V}^{\mathrm{BGA,RHO}}$	Verfahren zur Strategiemodifikation	106

Tabelle B.8: EA-Operatoren

Abkürzungen		
Abkürzung	Bezeichnung	Seite
BGA	Breeder Genetic Algorithm	44
cBGA	Breeder Genetic Algorithm mit Wettbewerb	109
cEA	Evolutionärer Algorithmus mit Wettbewerb	98
cES	Evolutionsstrategie mit Wettbewerb	112
cGA	Genetischer Algorithmus mit Wettbewerb	113
EA	Evolutionärer Algorithmus	13
ES	Evolutionsstrategie	32
GA	Genetischer Algorithmus	38
GENESIS	Simulationsumgebung für GA	137
k-NN	k-Nearest-Neighbor-Verfahren	203
MLP	Multilayer-Perceptron	204
NIST	National Institute of Standards and Technology	194
OCR	Optical Character Recognition	193
OptimA	Simulationsumgebung für ES	37
PeGAsuS	Simulationsumgebung für EA	136
PGA	Parallel Genetic Algorithm	44
PNN	Probabilistic Neural Net	204
QMA	Quadratmittelansatz	201
WSNN	Weighted Several Nearest-Neighbor-Verfahren	203

Tabelle B.9: Abkürzungen

Literaturverzeichnis

[Ack87] D.H. Ackley. *A Connectionist Machine for Genetic Hillclimbing.* Kluwer, Boston, 1987.

[Bäc92] Th. Bäck. Self-adaptation in genetic algorithms. In *Proceedings of the First European Conference on Artificial Life*, Seiten 263–271, MIT Press, Paris, 1992.

[Bäc96] Th. Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York, 1996.

[Bak85] J.E. Baker. Adaptive selection methods for genetic algorithms. In Grefenstette [Gre85], Seiten 101–106.

[Bak87] J.E. Baker. Reducing bias and inefficiency in the selection algorithm. In Grefenstette [Gre87], Seiten 14–21.

[BB91] R.K. Belew und L. Booker, Hrsg. *Proceedings of the 4th International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, 1991.

[BGK+95] R.S. Barr, B.L. Golden, J.P. Kelly, M.G.C. Resende und W.R. Stewart. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1:9–32, 1995.

[BH93] R.S. Barr und B.L. Hickman. Reporting computational experiments with parallel algorithms: Issues, measures, and expert's opinions. *ORSA Journal on Computing*, 5(1):2–18, 1993.

[Bro70] C.G. Broyden. The convergence of a class of double rank minimization algorithms, Part I and II. *J. Inst. Maths. Comp.*, 6:76–90, 1970.

[BRS66] H.J. Bremermann, M. Rogson und S. Salaff. Global properties of evolution processes. In H.H. Pattee, Hrsg., *Natural Automata and Useful Simulations*, Seiten 3–42, Macmillan and Co., Washington, 1966.

[BRS93] Th. Bäck, G. Rudolph und H.-P. Schwefel. Evolutionary programming and evolution strategies: Similarities and differences. In D.B. Fogel und W. Atmar, Hrsg., *Proceedings of the 2nd Annual Conference on Evolutionary Programming*, Seiten 11–22, Evolutionary Programming Society, San Diego, CA, 1993.

[BS91] Th. Bäck und H.-P. Schwefel. A survey of evolution strategies. In Belew and Booker [BB91], Seiten 2–9.

[BS93] Th. Bäck und H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–24, 1993.

[BS95] Th. Bäck und H.-P. Schwefel. Evolution strategies i: Variants and their computational implementation. In G. Winter, J. Périaux, M. Galán und P. Cuesta, Hrsg., *Genetic Algorithms in Engineering and Computer Science*, Seiten 111–126, Wiley, New York, Proc. First Short Course EUROGEN-95, Las Palmas de Gran Canaria, Spain, 1995.

[CH67] T.M. Cover und P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Processing*, IT-13:21–27, 1967.

[Das91] B.V. Dasarathy, Hrsg. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

[DJ75] K.A. De Jong. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975. Diss. Abstr. Int. 36(10), 5140B, University Microfilms No. 76-9381.

[dlMY94] M. de la Maza und D. Yuret. Dynamic hill climbing. *AI Expert*, 9(3):26–31, 1994.

[DS94] D. Davidor und H.-P. Schwefel, Hrsg. *Parallel Problem Solving from Nature 3*, Band 866 der *Lecture Notes in Computer Science*, Springer, Jerusalem, 1994.

[Eml84] J.M. Emlen. *Population Biology : The Coevolution of Population Dynamics and Behavior*. Macmillan Publishing Company, New York, 1984.

[Esh95] L.J. Eshelman, Hrsg. *Proceedings of the 6th International Conference on Genetic Algorithms*, Morgan Kaufmann, San Francisco, CA, Juli 1995.

[Fal81] D.S. Falconer. *Introduction to Quantitative Genetics*. Longman, London, 1981.

[Fis73] M. Fisz. *Wahrscheinlichkeitsrechnung und mathematische Statistik*, Band 40 der *Hochschulbücher für Mathematik*. VEB Deutscher Verlag der Wissenschaften, Berlin, 1973.

[Fle70] R. Fletcher. A new approach to variable metric algorithms. *Computer*, 13:317–322, 1970.

[Fle87] R. Fletcher. *Practical Methods of Optimization*, 2 Auflage. Wiley, 1987.

[Fog92] D.B. Fogel. *Evolving Artificial Intelligence*. PhD thesis, University of California, San Diego, CA, 1992.

[For93] St. Forrest, Hrsg. *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann, San Francisco CA, 1993.

[FOW66] L.J. Fogel, A.J. Owens und M.J. Walsh. *Artificial Intelligence Through Simulated Evolution*. Wiley, New York, 1966.

[FP63] R. Fletcher und M.J.D. Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6:163–168, 1963.

[Gau32] G.F. Gause. Experimental studies on the struggle for existence, i. mixed populations of two species of yeast. *J. Exp. Biol.*, 9:389–402, 1932.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, et al. *PVM: Parallel Virtual Machine – A Users' Guide and Tutorial for Networked Parallel Computing.* Scientific and Engineering Computation. MIT Press, 1994.

[GC93] P.J. Grother und G.T. Candela. Comparison of handprinted digit classifiers. Technical report, Image Recognition Group, National Institute of Standards and Technology, 1993.

[Gol70] D. Goldfarb. A family of variable metric methods derived by variational means. *Maths. Comp.*, 24:23–26, 1970.

[Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addisson-Wesley, Reading, 1989.

[Gre85] J.J. Grefenstette, Hrsg. *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum, Hillsdale NJ, 1985.

[Gre86] J.J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-16(1):122–128, 1986.

[Gre87] J.J. Grefenstette, Hrsg. *Proceedings of the 2nd International Conference on Genetic Algorithms*, Lawrence Erlbaum, Hillsdale NJ, 1987.

[Gre90] J.J. Grefenstette. A user's guide to genesis, version 5.0. Technical report, Navy Center for Applied Research in Artificial Intelligence, Washington, D.C., 1990.

[Gro92] P.J. Grother. Karhunen loéve feature extraction for neural handwritten character recognition. In *Applications of Artificial Neural Networks III, Vol 1709*, Seiten 155–166, Orlando, SPIE, 1992.

[Gro95] P.J. Grother. Nist special database 19 – handprinted forms and character database. Technical report, National Institute of Standard and Technology, Gaithersburgh MD, 1995.

[GS90] M. Gorges-Schleuter. *Genetic Algorithms and Population Structures – A Massively Parallel Algorithm.* Dissertation, Universität Dortmund, 1990.

[GW92] M.D. Garris und R.A. Wilkinson. Nist special database 3 – handwritten segemented characters. Technical report, National Institute of Standards and Technology, Gaithersburgh MD, 1992.

[Her92] M. Herdy. Reproductive isolation as strategy parameter in hierarchical organized evolution strategies. In Männer and Manderick [MM92], Seiten 207–217.

[Hol75] J.H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Harbour, MI, 1975.

[Hoo94] J. Hooker. Needed: An empirical science of algorithms. *Operations Research*, 42(2):201–212, 1994.

[Hoo95] J. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1(1):33–42, 1995.

[Kap96] C. Kappler. Are evolutionary algorithms improved by large mutations? In Voigt et al. [VERS96], Seiten 346–355.

[Kur95] F. Kursawe. Towards self-adapting evolution strategies. In IEEE Press, Hrsg., *Proceedings of the Second IEEE International Conference on Evolutionary Computation (IEEE/ICEC'95)*, Seiten 283–288, Perth, Australia, 1995.

[Lot25] A.J. Lotka. Elements of physical biology. 1925. reprinted in 1956 by Dover, Publications, Inc., New York.

[Mai93] F. Mailänder. Theorie und praktische Anwendung von Mustererkennungssystemen. Diplomarbeit, Universität Bonn, 1993.

[May73] R.M. May. *Stability and Complexity in model Ecosystems*. Princeton University Press, Princeton, New Jersey, 1973.

[MM92] R. Männer und B. Manderick, Hrsg. *Parallel Problem Solving from Nature 2*, Elsevier, Amsterdam, 1992.

[MSB91] H. Mühlenbein, M. Schomisch und J. Born. The parallel genetic algorithm as function optimizer. In Belew and Booker [BB91].

[MSV93a] H. Mühlenbein und D. Schlierkamp-Voosen. Analysis of selection, mutation and recombination in genetic algorithms. *Neural Network World*, 3:907–933, 1993.

[MSV93b] H. Mühlenbein und D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm: Continuous parameter optimization. *Evolutionary Computation*, 1(1):25–49, 1993.

[MSV94] H. Mühlenbein und D. Schlierkamp-Voosen. The science of breeding and its application to the breeder genetic algorithm. *Evolutionary Computation*, 1(4):335–360, 1994.

[MSV95] H. Mühlenbein und D. Schlierkamp-Voosen. Analysis of selection, mutation and recombination in genetic algorithms. In W. Banzhaf und F.H. Eeckman, Hrsg., *Evolution as a Computational Process*, Band 899 der *Lecture Notes in Computer Science*, Seiten 188–214, Springer, Berlin, 1995.

[Nat92] National Institute of Standards and Technology. *The First Census Optical Character Recognition System Conference*, Gaithersburgh MD, 1992.

[Nie83] H. Niemann. *Klassifikation von Mustern*. Springer, 1983.

[NM64] J.A. Nelder und R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1964.

[Pow64a] M.J.D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7:155–162, 1964.

[Pow64b] M.J.D. Powell. A method for minimizing a sum of squares of non-linear functions without calculating derivatives. *The Computer Journal*, 7:303–307, 1964.

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Vetterling und B.P. Flannery. *Numerical Recipes in C*, 2 Auflage. Cambridge University Press, 1992.

[Rap84] G. Rappl. *Konvergenzraten von Random-Search-Verfahren zur globalen Optimierung.* Dissertation, Hochschule der Bundeswehr München, 1984.

[Rec65] I. Rechenberg. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment, Library Translation No. 1122*, August 1965. Farnborough, Hants., UK.

[Rec73] I. Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, Stuttgart, 1973.

[Rec94] I. Rechenberg. *Evolutionsstrategie '94.* Frommann-Holzboog, Stuttgart, 1994.

[Ros60] H.H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, Oktober 1960.

[Rud92] G. Rudolph. Parallel approaches to stochastic global optimization. In W. Joosen und E. Milgron, Hrsg., *Parallel Computing: From Theory to Sound Practice, Proceedings of the European Workshop on Parallel Computing (EWPC 92)*, Seiten 256–267. IOS Press, 1992.

[Rud93] G. Rudolph. Massively parallel simulated annealing and its relation to evolutionary algorithms. *Evolutionary Computation*, 1(4):361–383, 1993.

[Rud96] G. Rudolph. *Convergence Properties of Evolutionary Algorithms.* Dissertation, Universität Dortmund, 1996.

[Sac92] L. Sachs. *Angewandte Statistik.* Springer, Berlin, 1992.

[SB95] H.-P. Schwefel und Th. Bäck. Evolution strategies ii: Theoretical aspects. In G. Winter, J. Périaux, M. Galán und P. Cuesta, Hrsg., *Genetic Algorithms in Engineering and Computer Science*, Seiten 127–140, Wiley, Chichester, Las Palmas de Gran Canaria, Spain, Proc. First Short Course EUROGEN-95, 1995.

[Sch65] H.-P. Schwefel. Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. Diplomarbeit, Technische Universität Berlin, 1965.

[Sch68] H.-P. Schwefel. Projekt MHD-Staustrahlrohr: Experimentelle Optimierung einer Zweiphasendüse, Teil I. Technischer Bericht 11.034/68 35, AEG Forschungsinstitut, Berlin, 1968.

[Sch75] H.-P. Schwefel. *Evolutionsstrategie und numerische Optimierung.* Dissertation, Technische Universität Berlin, 1975.

[Sch77] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie.* Birkhäuser, Basel, 1977.

[Sch81] H.-P. Schwefel. *Numerical Optimization of Computer Models.* Wiley, Chichester, 1981.

[Sch87] H.-P. Schwefel. Collective phenomena in evolutionary systems. In P. Checkland und I. Kiss, Hrsg., *Problems of Constancy and Change – The Complementarity of Systems Approaches to Complexity*, Band 2, Seiten 1025–1033, Budapest, Int'l Soc. for General System Research, 1987. Papers presented at the 31st Annual Meeting Int'l Soc. General System Research.

[Sch89] H. Schaffer, Hrsg. *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, 1989.

[Sch92] J. Schürmann. Mustererkennung mit statistischen Methoden. Vorlesungsskript Sommersemester, 1992. Daimler-Benz AG, Forschungszentrum Ulm, Institut für Informationstechnik.

[Sch95] H.-P. Schwefel. *Evolution and optimum seeking.* Sixth-Generation Computer Technology Series. Wiley, New York, 1995.

[SGH+89] R. Suchenwirth, J. Guo, I. Hartmann, G. Hincha, M. Krause und Z. Zhang. *Optical Character Recognition of Chinese Characters.* Vieweg & Sohn, Braunschweig, 1989.

[Sha70] D.F. Shanno. Conditioning of quasi-newton methods for function minimization. *Maths. Comp.*, 24:647–656, 1970.

[SHH62] W. Spendley, G.R. Hext und F.R Himsworth. Sequential application of simplex designs in optimization and evolutionary operation. *Technometrics*, 4:441–461, 1962.

[SM90] H.-P. Schwefel und R. Männer, Hrsg. *Parallel Problem Solving from Nature 1*, Band 496 der *Lecture Notes in Computer Science*, Springer, Dortmund, 1990.

[Spe90] D.F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109–118, 1990.

[Spr94] J. Sprave. Linear neighborhood evolution strategies. In A.V. Sebald und L.J. Fogel, Hrsg., *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, Seiten 42–51, World Scientific, River Edge (NJ), 1994.

[SRB95] H.-P. Schwefel, G. Rudolph und Th. Bäck. Contemporary evolution strategies. In F. Morán, A. Moreno, J.J. Merelo und P. Chacón, Hrsg., *Advances in Artificial Life, Proceedings of the Third European Conference on Artificial Life*, Seiten 893–907, Springer, Berlin, 1995.

[SS68] M.A. Schumer und K. Steiglitz. Adaptive step size random search. *IEEE Trans. Atomatic Control*, AC-13:270–276, 1968.

[SV93] D. Schlierkamp-Voosen. PeGAsuS – Programming Environment for Parallel Genetic Algorithms. Handbuch, GMD – Forschungszentrum Informationstechnik GmbH, Research Group for Adaptive Systems, 1993.

[SV96] V. Schnecke und O. Vornberger. An adaptive parallel genetic algorithm for vlsi-layout optimization. In Voigt et al. [VERS96], Seiten 859–868.

[SVM94] D. Schlierkamp-Voosen und H. Mühlenbein. Strategy adaptation by competing subpopulations. In Davidor and Schwefel [DS94], Seiten 199–208.

[SVM96] D. Schlierkamp-Voosen und H. Mühlenbein. Adaptation of population size by competing subpopulations. In *3rd International Conference of Evolutionary Computation*, Seiten 330–335, Nagoya, Mai 1996.

[SW81] F. Solis und R. Wets. Minimization by random search techniques. *Math. of Operation Research*, 6:19–30, 1981.

[TŽ88] A. Törn und A. Žilinskas. *Global Optimization*, Band 350 der *Lecture notes in Computer Science*. Springer, 1988.

[VBH+91] T.P. Vogl, K.L. Blackwell, S.D. Hyman, G.S. Barbour und D.L. Alkon. Classifikation of japanese kanji using principal component analysis as a preprocessor to an artificial neural network. *International Joint Conference on Neural Networks*, 1:233–238, 7 1991. IEEE and International Neural Network Society.

[VERS96] H.-M. Voigt, W. Ebeling, I. Rechenberg und H.-P. Schwefel, Hrsg. *Parallel Problem Solving from Nature 4*, Lecture Notes in Computer Science, Springer, Berlin, 1996.

[VMC95] H.-M. Voigt, H. Mühlenbein und D. Cvetković. Fuzzy recombination for the breeder genetic algorithm. In Larry J. Eshelman, Hrsg., *Proceedings of the Sixth International Conference on Genetic Algorithms*, Seiten 104–111. Morgan Kaufmann, 1995.

[VMSV96] H.-M. Voigt, H. Mühlenbein und D. Schlierkamp-Voosen. Response-to-selection equation for skew fitness distributions. In *3rd International Conference of Evolutionary Computation*, Seiten 820–825, Nagoya, 1996.

[Voi92] H.-M. Voigt. Fuzzy evolutionary algorithms. Technical Report TR-92-038, International Computer Science Institute, Berkeley CA, 1992.

[Vol25] V. Volterra. Fluctuations in the abundance of a species considered mathematically. *Nature*, 118:558–560, 1925.

[WM95] D.H. Wolpert und W.G. Macready. No free lunch theorem for search. Technical Report SFI-TR-95-02-010, The Santa Fe Institute, Santa Fe NM, 1995.

[Wri91] A. H. Wright. *Foundations of Genetic Algorithms*, Kapitel Genetic Algorithms for Real Parameter Optimization, Seiten 205–218. Morgan Kaufmann, San Mateo CA, 1991.

[YdlM93] D. Yuret und M. de la Maza. Dynamic hill climbing: Overcoming the limitations of optimization techniques. In *Second Turkish Symposium on Artificial Intelligence and Artifical Neural Networks*, Seiten 254–260, 1993.

[Yur94] D. Yuret. From genetic algorithms to effizient optimization. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1994.