# Optimierung in der Simulation: Evolutionäre Algorithmen

Ulrich Hammel* und Thomas Bäck†

21. Juli 1998

## 1 Motivation

Vereinfacht dargestellt beschreibt ein Simulationsmodell einen Zusammenhang

$$\psi : S \times J \times T \rightarrow O$$

zwischen internen Zuständen $S$, Eingaben $J$, der Zeit $T$ und Ausgabegrößen $O$ eines realen Systems. Eine Simulation, d.h. die analytische oder numerische Berechnung der Funktion $\psi(s_0, j, t)$, $s_0 = s(t_0) \in S$, $j \in J$, $t$,$t_o \in T$ liefert Informationen über die Auswirkung von Szenarien $(s_0, j)$, beantwortet also Fragen der Art „Was wäre wenn ...?“.

Das eigentliche Ziel besteht aber häufig im Auffinden von Szenarien $(s_0^*, j^*)$, für welche das System bezüglich eines Kriteriums $G : O \rightarrow I\!R$ und $t' \in T$ eine gewünschte Ausgabe möglichst genau produziert. Es geht also um die Beantwortung der Frage „Was ist zu tun, um ...?“:

$$G(\psi(s_o^*, j^*, t')) \rightarrow extremum \tag{1}$$

Zum Zwecke einer allgemeinen Darstellung treffen wir hier noch keine Annahme über die Grundmengen von $S, J$ und $T$. Gleichfalls betrachten wir zunächst nur Optimierungsprobleme ohne Nebenbedingungen. (Siehe hierzu aber Abschnitt 6.5). Wegen $\min\{G(x)\} = -\max\{-G(x)\}$ unterstellen wir aber ohne Beschränkung der Allgemeinheit im folgenden eine Minimierungsaufgabe.

Zur Lösung des Optimierungsproblems (1) können je nach Gestalt der Funktionen $G$ beziehungsweise $\psi$ unterschiedliche Verfahren herangezogen werden. Zur Anwendung analytischer Lösungsverfahren müssen bestimmte Voraussetzungen wie z.B. die mindestens zweifach stetige Differenzierbarkeit der Funktion $G$ erfüllt sein. In der Regel liegt die Funktion $\psi$ aber nicht einmal in geschlossener Form vor, sondern ist implizit als Funktion einer (zeitinvarianten) Zustandsübergangsfunktion $\delta : S \times J \rightarrow S$ definiert. Zur

*Universität Dortmund, Informatik XI, D-44221 Dortmund (e-mail: hammel@ls11.cs.uni-dortmund.de).

†Informatik Centrum Dortmund, Joseph-von-Fraunhofer-Str. 20, D-44227 Dortmund und Universität Leiden, Niels Bohrweg 1, NL-2333 CA Leiden, Niederlande (e-mail: baeck@icd.de).

Berechnung der Funktion $\psi$ sind zumeist aufwendige (Laufzeit, Speicherbedarf) Simulationen erforderlich. Desweiteren kann die explizite Formulierung der Funktion $G$ aufgrund schwer formalisierbarer oder mehrfacher gegenläufiger Gütekriterien Schwierigkeiten bereiten (siehe Abschnitt 6.4). Aus diesen Gründen werden Lösungen des Optimierungsproblems (1) häufig manuell durch (mehr oder weniger) systematisches Experimentieren mit dem Simulationsmodell gesucht. Gelegentlich kommen auch Verfahren der optimalen Versuchsplanung [LK91] zum Einsatz.

Sofern die Formulierung einer geeigneten Gütefunktion $G$ gelingt und die notwendigen technischen Voraussetzungen (z.B. hinreichende Rechenkapazität, Batch-Betrieb des Simulators) erfüllt sind, ist im Grundsatz aber die Automatisierung der Optimumsuche möglich, obgleich gerade bei der Kopplung von Simulation und Optimierung Probleme auftreten können, die den Einsatz vieler Suchverfahren einschränken oder ausschließen. So kommen aus den genannten Gründen in der Regel nur direkte Suchverfahren, die keinerlei weitere analytische Information über die Funktion $G$ benötigen, in Frage.

Zur Vereinfachung der Notation gelte im folgenden $F(x) = G(\psi(x, t'))$, $x \in X \subseteq S \times J$ für ein beliebiges aber festes $t' \in T$. Mit $P, P', R \subseteq X$ orientieren sich direkte Suchverfahren an folgendem groben Schema:

**Algorithmus 1 (Direktes Suchverfahren)**

$g := 0;$
*initialisiere* $P(g)$;
*bewerte* $P(g)$;
do while not *Abbruch-Bedingung*
  $P'(g) := variiere P(g);$
  *bewerte* $P'(g)$;
  $P(g+1)$ := *berechne aus* $P'(g)$;
  $g := g + 1;$
done

Direkte Verfahren versuchen demnach durch iterative Variation bekannter Lösungen bessere (möglicherweise optimale) Lösungen zu finden. $P(g)$ ist eine Menge bereits bewerteter Lösungen. Ausgehend von $P(g)$ werden Tastschritte $P'(g)$ ausgeführt, aus deren Bewertung eine neue Lösungsmenge $P(g+1)$ berechnet wird.

Konkrete Instanzen des Algorithmus 1 unterscheiden sich hauptsächlich in der Ausprägung der Operationen "variiere" und "berechne", welche zwangsläufig von der Gestalt des Suchraumes $X$ und der Zielfunktion $F$ abhängig sind.

Im Rahmen dieses Kapitels kann kein vollständiger Überblick über das weite Feld der nichtlinearen Optimierung gegeben werden. Wir beschränken uns im folgenden auf einige potentielle Eigenschaften des Suchraums $X$, denen gerade im Kontext der simulationsgestützten Optimierung besondere Bedeutung zukommt und anhand derer die Eigenschaften evolutionärer Algorithmen im Vergleich zu anderen Verfahren deutlich werden.

**Algebraische Eigenschaften:** Der Suchraum $X$ wird zumeist durch mehrere, häufig durch eine große Zahl von Entscheidungsvariablen aufgespannt: $X \subseteq X_1 \times \ldots \times X_n$; $n$ ist die Dimension des Suchraums. Die Mengen $X_i$ können endliche (z.B. $\{0, 1\}$), abzählbare (z.B. $\mathbb{Z}$) oder überabzählbare (z.B. $\mathbb{R}$) Mengen sein. Die meisten direkten Suchverfahren wurden für spezielle Suchräume entworfen und nutzen deren algebraische Eigenschaften. So wurde seit den fünfziger Jahren eine Vielzahl numerischer Verfahren zur Lösung von Problemen der Parameteroptimierung ($X \subseteq \mathbb{R}^n$) entwickelt, die auf unterschiedlichen Techniken, wie der Approximation der ersten (und zweiten) partiellen Ableitungen oder geschickten Koordinatentransformationen beruhen [Sch95] [Rao96, Kap. 6]. Solche Algorithmen eignen sich z.B. nicht zur Lösung ganzzahliger (etwa kombinatorischer) Optimierungsprobleme. Viele Simulationsmodelle besitzen aber sowohl diskrete (z.B. strukturbeschreibende) als auch reellwertige (z.B. prozeßbeschreibende) Entscheidungsvariablen.

**Dimensionalität:** Eine vollständige Durchmusterung des Suchraums ist zumeist nur für sehr kleine $n$ realisierbar. Simulationsmodelle besitzen aber häufig eine große Anzahl von Entscheidungsvariablen. Für große $n$ treten bei manchen Verfahren numerische Instabilitäten auf [Sch95].

**Multimodalität:** Multimodale Zielfunktionen besitzen mehrere, häufig eine große oder sogar unendliche Zahl lokaler Optima. Das sind jene $x \in X$ in deren Umgebung nur größere Werte von $F$ zu finden sind:

$$\exists \epsilon > 0, y \in \{x' \in X \,|\, d(x, x') < \epsilon\} \Rightarrow F(x) \leq F(y) \quad , \tag{2}$$

wobei $d : X \times X \to \mathbb{R}^+$ ein geeignetes Abstandsmaß (z.B. euklidischer Abstand oder Hamming-Distanz) in $X$ ist. Lokale Optima können sich aber bezüglich ihrer Zielfunktionswerte stark unterscheiden. Gesucht ist daher ein globales Optimum $x^* \in X$, für das gilt:

$$\forall y \in X : F(x^*) \leq F(y). \tag{3}$$

Das globale Optimierungsproblem (3) ist nicht allgemein lösbar [Bäc96, S. 37]. In der Praxis genügt es aber zumeist, gute oder zumindest bessere als die bekannten Lösungen von (1) aufzufinden. Um in multimodalen Landschaften erfolgreich zu sein, muß ein Verfahren lokale Optima „überwinden“ können. Die pfadorientierte Suche, etwa in Gradientenrichtung, konvergiert dagegen in der Regel nur zum nächsten lokalen Optimum.

**Nichtdeterminismus:** Die Zustandsübergangsfunktion $\delta$ ist häufig nichtdeterministisch wie im Falle ereignisorientierter Simulationsmodelle. Das führt zu einer stochastischen Optimierungsaufgabe der Form $\overline{F}(x, \omega) \to \min, \omega \in \Omega$, wobei $\Omega$ ein Wahrscheinlichkeitsraum ist. Ein gängiger Ansatz lautet

$$\overline{F} = E[\![F(x, \omega)]\!] = \int F(x, \omega) P(d\omega). \tag{4}$$

$E[\![.]\!]$ ist der Erwartungswert. Gelegentlich werden aber auch andere Momente der Verteilung von $\overline{F}$ benutzt. Die Momente der Verteilung können nur auf der Basis einer endlichen (häufig geringen) Zahl von Simulationen geschätzt werden, so daß die resultierende Zielfunktion zu einem gewissen Maße unbestimmt ist. Viele direkte deterministische Suchverfahren liefern nur dann noch befriedigende Ergebnisse, wenn diese Unbestimmtheit klein bleibt im Verhältnis zum Absolutbetrag der Funktion [Ham81]. Methoden der stochastischen Programmierung setzen in der Regel die Konvexität und stetige Differenzierbarkeit der Zielfunktion voraus. (Zum Thema stochastische Optimierung siehe z.B. [KW94, EW88] und Abschnitt 6.3.)

Evolutionäre Algorithmen sind Versuche, den augenscheinlich äußerst effizienten Prozeß der biologischen Evolution nachzuahmen und zur Lösung zumeist technischer Aufgaben zu erschließen.

Im folgenden werden wir die Grundlagen evolutionärer Algorithmen (Abschnitte 2 und 3) und insbesondere deren Fähigkeit zur Selbstanpassung (Abschnitt 4) skizzieren. Gerade in den letzten Jahren ist die verfügbare Literatur zu einzelnen Varianten des evolutionären Rechnens stark angewachsen; siehe etwa [Rec94, Sch95] zu *Evolutionsstrategien*, [Gol89, Dav91, Mic96] zu *Genetischen Algorithmen*, [Fog95] zum *Evolutionary Programming*, [Koz94] zum *Genetischen Programmieren*, und [Bäc96, BFM97, BHS97] zum übergeordneten Thema *evolutionäres Rechnen*. Wir verzichten daher auf eine detaillierte Betrachtung einzelner konkreter Algorithmen zugunsten einer vergleichenden Darstellung.

Berücksichtigt man die Vielfalt der Anwendungsbereiche und die Unterschiede in der algorithmischen Umsetzung des Prinzips der evolutionären Suche, so wundert es nicht, daß bis heute keine tragfähige übergreifende Theorie des evolutionären Rechnens existiert. Dennoch gibt es eine Reihe grundlegender theoretischer Aussagen, von denen einige in Abschnitt 5 dargestellt werden.

Das evolutionäre Rechnen bietet einen sehr allgemeinen Ansatz zum Entwurf von Optimierungsverfahren. Die Umsetzung für eine konkrete Aufgabenstellung erfordert aber häufig aufgabenspezifische Anpassungen. Entsprechende Techniken sind Gegenstand des Abschnitts 6.

Das Kapitel schließt mit der Aufzählung einiger Anwendungsbeispiele (Abschnitt 7) und einer Zusammenfassung (Abschnitt 8).

# 2 Evolutionäre Algorithmen

Seit ca. 3,5 Milliarden Jahren existiert organisches Leben auf der Erde. Nach der Darwinschen Interpretation der Entwicklung von primitiven Lebensformen hin zu immer komplexeren Organismen ist die Evolution, grob vereinfacht, das Ergebnis des Wechselspiels zwischen der Erzeugung neuer genetischer Information und ihrer Bewertung und Selektion. Die Deoxyribonukleinsäure (DNS) ist Träger der genetischen Information. Vergleicht man die Anzahl möglicher Zustände der DNA höherer Organismen von etwa $10^{19.500.000}$ mit der Zeit von $3 \cdot 10^{17}$ Sekunden, die seit dem Urknall verstrichen sind, so wird deutlich, daß die Evolution auf einem äußerst effizienten Strukturbildungsprozeß beruhen muß.

Bereits in den fünfziger und sechziger Jahren haben mehrere Forscher unabhängig voneinander das Potential dieses Prozesses für unterschiedliche Aufgaben zu erschließen versucht. So entwickelten unabhängig voneinander Holland [Hol62, Hol75] die *Genetischen Algorithmen* (GA) als Modell für Adaptations- und Klassifikationsprozesse und Fogel [Fog62, FOW66] das *Evolutionary Programming* (EP) zur Zeitreihenvorhersage mittels endlicher Automaten. Nahezu zeitgleich dazu entwarfen Rechenberg [Rec65, Rec71] und Schwefel [Sch68, Sch75] die Evolutionsstrategien (ES) als Heuristiken für die experimentelle Optimierung.

Überwiegend werden evolutionäre Algorithmen (EA) heute als direkte Suchverfahren zur Lösung von Optimierungsaufgaben eingesetzt. EAs orientieren sich ebenfalls an dem in Algorithmus 1 dargestellten Iterationsschema:

**Algorithmus 2 (Evolutionärer Algorithmus)**

$g := 0;$
*initialisiere* $P(g)$;
*bewerte* $P(g)$;
do while not *Abbruch-Bedingung*
    $P'(g)$ := *rekombiniere* $P(g)$;
    $P''(g)$ := *mutiere* $P'(g)$;
    *bewerte* $P''(g)$;
    $P(g+1)$ := *selektiere aus* $P''(g) \cup R(g)$;
    $g := g + 1;$
done

Wesentliche Unterschiede zu traditionellen direkten Suchverfahren sind:

1. EAs verwenden kein **zentrales** internes Modell (lokale Approximation) der Zielfunktion. Die Operation „variiere“ aus Algorithmus (1) wird durch zwei numerisch vergleichsweise einfache Funktionen „rekombiniere“ (Abschnitt 3.2) und „mutiere“ (Abschnitt 3.3) realisiert. An die Stelle der Operation „berechne“ tritt die Selektion der Menge $P(g+1)$ als Teilmenge von $P''(g) \cup R(g)$; $R(g) = \emptyset$ oder $R(g) = P(g)$ (siehe Abschnitt 3.4).

2. Die Objektvariablen $x \in X$ werden in der Regel kodiert: $P(g) \subseteq \{\Gamma(x) | \, x \in X\}$. $\Gamma : X \to \mathcal{R}$ ist eine Kodierungsfunktion aus dem Suchraum $X$ in eine Repräsentation $\mathcal{R}$ (Abschnitt 3.1).

3. EAs arbeiten zumeist mit einer größeren aber wählbaren Kardinalität von $P(g)$.

4. EAs arbeiten nichtdeterministisch.

5. EAs haben die Fähigkeit, während der Suche ein **verteiltes**, differenziertes internes Modell der Zielfunktionstopologie aufzubauen.

Aufgrund der Eigenschaften 1 und 2 sind die Algorithmen weitgehend unabhängig von der konkreten Gestalt des Suchraums $X$ und bieten damit ein breites Anwendungsspektrum (z.B. kombinatorische Optimierung, Parameteroptimierung, Mixed-Integer-Optimierung und Strukturoptimierung). Wegen der Eigenschaften 3 und 4 reagieren EAs robust im Falle multimodaler, unstetiger und nichtdeterministischer oder gestörter Zielfunktionen. Aus 3 ergibt sich die Skalierbarkeit der Verfahren zwischen volumenorientierter und pfadorientierter Suche und damit eine gute Ausnutzung vorhandener Resourcen. Wegen Eigenschaft 4 bewahren EAs ihre Effizienz auch bei der lokalen Suche.

Die verwendete Metapher legt für die Beschreibung von EAs ein der Genetik entlehntes Vokabular nahe: Die Menge $P(g)$ wird als *Population* der *Generation* $g$ und deren Elemente $I \in P(g)$ als *Individuen* bezeichnet. Die genetische Information eines Individuums $I$ wird als *Genotyp* von $I$ und der Träger dieser Information als *Chromosom* bezeichnet. Der Genotyp bestimmt gemäß der Kodierungsfunktion die Eigenschaften eines Individuums, seinen *Phänotyp*, also ein Element des Suchraumes $x \in X$. Für jedes Individuum wird eine *Fitneß* auf der Basis des Zielfunktionswertes $F(x)$ bestimmt. Im Selektionsschritt wird schließlich die neue Generation $P(g+1)$ aus der Elternpopulation $P''(g)$ (oder gegebenenfalls $P''(g) \cup R(g)$) unter Bevorzugung von Individuen mit überdurchschnittlicher Fitneß gebildet.

Aus der Genetik sind verschiedene Prozesse bekannt, die für die Varianz der genetischen Information sorgen. EAs verwenden in der Regel die *Rekombination*, also die Kombination genetischer Information aus zwei oder mehreren elterlichen Individuen und die *Mutation*, die zufällige, richtungslose Modifikation des Genotyps.

Der Zusammenhang wird in Graphik 1 skizziert.

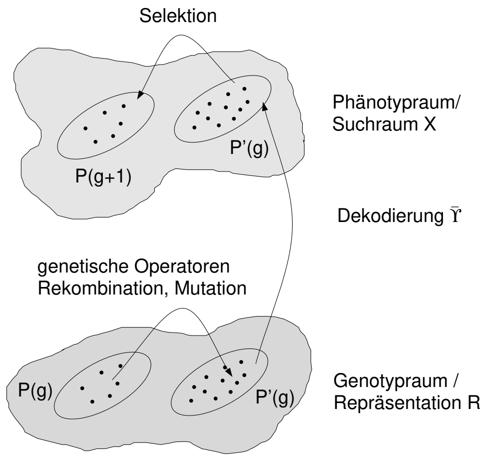


Abbildung 1: Genotyp-Phänotyp Beziehung nach Fogel [Fog95, S. 39].

Das „No-Free-Lunch-Theorem“ [WM97] besagt, daß sich zwei beliebige Suchverfahren bezüglich Ihrer Leistungsfähigkeit gemittelt über alle Optimierungsprobleme nicht unterscheiden. EAs wurden für Aufgaben entworfen, welche eine besondere Robustheit der Lösungsverfahren erfordern, wie z.B. Robustheit bezüglich Multimodalität, Unstetigkeiten oder Unbestimmtheit der Zielfunktion. Folglich gibt es auch Anwendungsbereiche, für welche EAs ungeeignet oder ineffizient sind wie z.B. lineare, quadratische oder separable Probleme. In seinem Anwendungsbereich stellt das evolutionäre Rechnen aber ein flexibles und vergleichsweise leicht handhabbares Instrumentarium zur Automatisierung schwieriger Suchprobleme bereit, wie durch die große Zahl dokumentierter Anwendungen belegt wird [Ala94].

# 3 Die Bausteine evolutionärer Algorithmen

Simulationsmodelle beschreiben Zusammenhänge zwischen Objekten realer Systeme, wie z.B. das Zusammenspiel zwischen Reaktoren, Pumpen, Heizern und Kühlern einer chemischen Anlage, deren Parameter den Phänotypraum bestimmen. EAs operieren auf abstrakten mathematischen Objekten, wie Zeichenketten oder reellwertigen Vektoren, dem Genotypraum oder auch *Repräsentation* $\mathcal{R}$. In vielen Fällen wird daher eine *Kodierungsfunktion* $\Gamma : X \to \mathcal{R}$ bzw. *Dekodierungsfunktion* $\overline{\Gamma} : \mathcal{R} \to X$ zwischen diesen Räumen benötigt. Die Anwendung von EAs erfordert somit zunächst den Entwurf von $\mathcal{R}, \Gamma$ und $\overline{\Gamma}$. Auf der Basis der Repräsentation werden dann die Variationsoperatoren *Rekombination* und *Mutation* definiert. Die Wahl der Selektion ist dagegen unabhängig von der Repräsentation formulierbar, hat aber gegebenenfalls die Architektur des Rechnersystems zu berücksichtigen (siehe Abschnitt 6.1).

## 3.1 Repräsentationen

**Binär-Strings:** Die wohl bekannteste Spielart des evolutionären Rechnens, die genetischen Algorithmen (GA), verwenden in ihrer ursprünglichen Form binäre Strings einer festen Länge $l$: $\mathcal{R}_{\boldsymbol{B}} = \mathbb{B}^l$. Diese Repräsentation eignet sich daher unmittelbar zur Kodierung pseudoboolescher Optimierungsprobleme der Form $F : X \subseteq \mathbb{B}^l \to \mathbb{R}$. GAs wurden aber auch erfolgreich zur Lösung kombinatorischer und reellwertiger Optimierungsprobleme ($F : X \subseteq \mathbb{R}^n \to \mathbb{R}$) eingesetzt. In der Standardkodierung wird jeder Wert $(x_1, .., x_n) \in X$ durch einen Binär-String $(b_{1_1}, .., b_{1_{l_1}}, .., b_{n_1}, .., b_l)$ repräsentiert (hier dargestellt für $X \subseteq \mathbb{R}^n$):

$$x_i = u_i + \frac{v_i - u_i}{2^{l_i} - 1} \cdot (\sum_{j=0}^{l_i - 1} b_{i_{(l_i - j)}} \cdot 2^j) \tag{5}$$

für $i = 1 \ldots n$ und $x_i$ aus dem Intervall $[u_i, v_i]$. Die Wahl der Kodierung hat aber großen Einfluß auf die Effizienz des Suchprozesses. Aus diesem Grunde werden auch alternative Kodierungen wie der Gray-Code verwendet [Bäc96, S. 109ff.].

Die ursprüngliche Bevorzugung der binären Repräsentation in GAs wird formal durch das Schematheorem [Hol75] begründet. Ein Schema bezeichnet eine Menge von Strings, die an gewissen Positionen übereinstimmen. Schemata werden als Zeichenketten über dem Alphabet $\{0, 1, *\}$ notiert, wobei $*$ den Wert "undefiniert" darstellt. Das Schema $[1 * 0*]$ enthält demnach die vier Strings $1000, 1100, 1001$ und $1101$. Die Ordnung des Schemas $H = h_1, \ldots, h_l$ ist definiert als $o(H) = |\{i | h_i \in \{0, 1\}\}|$. $o(H)$ ist also die Anzahl der definierten Positionen in $H$. Die definierende Länge von $H$ gibt die Länge des längsten Teilstrings zwischen zwei definierten Positionen an: $\Delta(H) = \max\{i | h_i \in \{0, 1\}\} - \min\{i | h_i \in \{0, 1\}\}$. Das Schematheorem besagt, daß insbesondere jene Schemata mit überdurchschnittlich positivem Einfluß auf die Zielfunktion, geringer definierender Länge und kleiner Ordnung (sogenannte Building Blocks) sich in den Folgepopulationen exponentiell anreichern werden. Die binäre Repräsentation in Verbindung mit der proportionalen Selektion (s.u.) bietet hierfür die optimale Sampling-Strategie [Hol75]. Die Interpretation des Schematheorems wird in der Literatur aber kontrovers diskutiert [Gre93, Fog95].

**Reellwertige Repräsentation:** Im Falle $X \subseteq I\!R^n$ bietet es sich an, die Elemente aus $X$ direkt als reellwertige Vektoren zu repräsentieren wie dies in *Evolutionsstrategien* (ES) wie auch beim *Evolutionary Programming* (EP) geschieht. Aber auch viele Varianten genetischer Algorithmen verwenden für $X \subseteq I\!R^n$ inzwischen reellwertige Repräsentationen $\mathcal{R}_{I\!R} \subseteq I\!R^n$.

**Ganzzahlige Repräsentationen:** Für den Fall $X \subseteq Z\!\!\!Z$ können die Konstruktionsprinzipien (z.B. Wahrscheinlichkeitsverteilungen) aus $\mathcal{R}_{I\!R}$ auf $\mathcal{R}_{Z\!\!\!Z} \subseteq Z\!\!\!Z^l$ übertragen werden [Rud94b]. Endliche Teilmengen aus $Z\!\!\!Z$ werden in der Regel auf $\mathcal{R}_{I\!B}$ abgebildet. Häufig sind auch gemischt-ganzzahlige Probleme anzutreffen, für die sich gemischte Repräsentationen $\mathcal{R}_{I\!R} \times \mathcal{R}_{Z\!\!\!Z}$ anbieten. Allerdings ist darauf zu achten, daß im Falle gemischter Repräsentationen auch die genetischen Operatoren „kompatibel“ entworfen werden (z.B. Adaptivität der Operatoren)[BS95].

**Permutationen:** Permutationen wie sie beispielsweise bei Scheduling-Problemen oder in der Routenplanung auftreten werden gelegentlich auf $\mathcal{R}_{Z\!\!\!Z}$ abgebildet. Die Standard-Operatoren, wie sie weiter unten beschrieben werden, generieren dann in der Regel Nachkommen, welche keine gültigen Permutationen darstellen. Zur Vermeidung dieses Problems wurden zahlreiche (zumeist applikationsspezifische) Varianten vorgeschlagen [Whi97].

**Andere Repräsentationen:** Strukturbeschreibende Variablen oder geometrische Objekte lassen sich zwar grundsätzlich auch auf die bereits dargestellte Weise repräsentieren, dabei gehen aber strukturelle oder räumliche Beziehungen der repräsentierten Objekte verloren. In der Praxis werden daher auch häufig komplexere Repräsentationen (z.B. Graphen, Parse-Trees) [Ang97a, SHMM96] verwandt.

Obgleich schon aufgrund der Vielfalt möglicher Anwendungen keine generelle Rezeptur für den Entwurf einer geeigneten Repräsentation angegeben werden kann, nennen wir nachfolgend einige Punkte, die zu beachten sind (vergleiche auch [Ron97]):

1. Die Dekodierungsfunktion $\Gamma$ sollte möglichst stark kausal sein. D.h., kleine Änderungen des Genotyps bewirken kleine Änderungen des Phänotyps [Rec94].

2. Die Repräsentation sollte “fair” sein in dem Sinne, daß gleich große Regionen des Suchraumes gleich viele Repräsentanten in $\mathcal{R}$ besitzen. Sinngemäß sollte gelten: $|\{I|d(\Gamma(I),x) < \epsilon\}| \simeq |\{I|d(\Gamma(I),y) < \epsilon\}|$ für alle $x, y \in X$.

3. Durch eine geschickte Wahl der Repräsentation läßt sich häufig die Generierung ungültiger Individuen vermeiden [Whi97] oder auch die Zielfunktion „glätten“ [Bea93, Rud91].

4. Problemnahe Repräsentationen, die möglichst viel Information aus $X$ erhalten, führen in der Regel zu effizienteren Algorithmen und lassen sich leichter erweitern, z.B. bezüglich der Integration von Expertenwissen [Dav91, Mic96].

## 3.2 Rekombination

Die Rekombination $P(g)^N \rightarrow P(g)$ ist eine richtungslose, N-stellige Operation, wobei zumeist wie auch im folgenden $N = 2$ gilt. Gelegentlich werden aber auch globale Varianten, d.h. $N = |P|$ verwendet.

Ein neues Individuum $I' = (I'_1, \ldots, I'_l)$ wird dabei durch Kombination der genetischen Informationen von $N$ Eltern $I^1, \ldots, I^N, I^i = (I^i_1, \ldots, I^i_l)$ erzeugt. In der Literatur wurden zahlreiche Operatoren vorgeschlagen, von denen hier nur die gebräuchlichsten dargestellt werden.

**z-Point-Crossover:** Aus dem Intervall $[1, l-1]$ werden $z$ Positionen $j_1 < j_2 < \ldots < j_z$ gleichverteilt gezogen. $I'$ wird (hier dargestellt für ungerades z) gebildet als $I' = I^1_1, \ldots I^1_{j_1}, I^2_{j_1+1}, \ldots, I^2_{j_2}, I^1_{j_2+1}, \ldots, I^1_z, I^2_{z+1}, \ldots, I^2_l$. Die gebräuchlichsten Varianten sind das *One-Point-Crossover* ($z = 1$) und das *Two-Point-Crossover* ($z = 2$).

**Uniform-Crossover:** Es gilt: $z = l - 1$. Die $I'_i$ werden für jedes $i$ gleichverteilt aus der Menge der korrespondierenden Elemente der Eltern gezogen. Einen Vergleich der Verfahren findet man z.B. in [Bäc96].

**Weitere Crossover-Varianten:** Es wurden Erweiterungen des Crossover mit dem Ziel vorgeschlagen, den durch die Position der Gene verursachten Bias zu vermindern (*Shuffle-Crossover*, [ECS89]) oder eine adaptive Wahl der Crossover-Punkte (*Punctuated Crossover*, [SM87]) zu ermöglichen. (Siehe auch Abschnitt 4.) Für $\mathcal{R}_R$ werden bei der *intermediären Rekombination* die $I'_i$ arithmetisch bestimmt: $I'_i = u_i I^1_i + (1 - u_i) I^2_i$, wobei zumeist $u_i = 1/2$ gesetzt wird [Sch95].

## 3.3 Mutation

Die Mutation sorgt für eine zufällige und richtungslose Modifikation einzelner Gene. Sie ist daher eine einstellige Operation $P(g) \rightarrow P(g)$. Im Falle binärer Repräsentationen wird jedes $I_i$ mit einer kleinen Wahrscheinlichkeit $p_m$ invertiert. In der Regel werden Werte von $p_m \in [0.001, 0.01]$ verwendet. Die Analyse einfacher Beispiele sowie empirische Befunde legen aber nahe, $p_m = 1/l$ umgekehrt proportional zur Länge des Strings $l$ oder sogar zeitlich variabel zu wählen [Bäc96, BS96].

Im Falle reellwertiger Repräsentationen wird die Mutation zumeist als additiv normalverteilter Prozeß modelliert: $I'_i = I_i + N(0, \sigma_i)$. Auch hier stellt sich die Frage nach der Wahl der Standardabweichungen (*Schrittweiten*) $\sigma_i$. Der Bestwert (z.B. bezüglich der lokalen Fortschrittsgeschwindigkeit) von $\sigma_i$ ist sowohl von der Zielfunktion, als auch vom aktuellen Fortschritt der Suche abhängig. Letztlich stellt die optimale Steuerung der Schrittweiten $\sigma_i$ wieder ein (dynamisches) Optimierungsproblem dar. Insofern liegt es nahe, solche *Strategievariablen* endogen durch den Evolutionsprozeß selbst anpassen zu lassen. Derartige Techniken sind Gegenstand des Kapitels 4.

## 3.4 Selektion

Während die richtungslosen Operatoren Mutation und Rekombination lediglich für die Entstehung neuer genetischer Information sorgen, wird die evolutionäre Suche erst durch die Selektion zu einem zielgerichteten Prozeß indem jenen Individuen $I^i \in P(g)$ mit überdurchschnittlicher Güte $F(\overline{\Gamma}(I^i))$ eine überdurchschnittliche Selektionswahrscheinlichkeit $p_{s_i}$ zugeordnet wird. $P^F$ bezeichne im folgenden eine bezüglich $F \circ \overline{\Gamma}$ geordnete Menge: $I^i, I^j \in P^F, i, j \in \mathbb{N}, i \leq j \Rightarrow F(\overline{\Gamma}(I^i)) \leq F(\overline{\Gamma}(I^j))$.

Die gebräuchlichsten Selektionsverfahren sind:

**Proportionale Selektion:** Jedes $I^i \in P''(g)$ wird mit einer der relativen Fitneß von $I^i$ proportionalen Wahrscheinlichkeit selektiert:

$$p_{s_i} = \frac{S(F(\Gamma(I^i)))}{\sum_{j=1}^{|P''(g)|} S(F(\Gamma(I^j)))} \tag{6}$$

Durch eine Skalierungsfunktion $S$ wird die Einhaltung der Bedingung $p_{s_i} \in [0, .., 1]$ sichergestellt.

**$(\mu, \lambda)$-Selektion:** Aus den $\mu$ besten der $\lambda$ Individuen in $P''^F(g)$ wird gleichverteilt gezogen:

$$p_{s_i} = \begin{cases} 1/\mu, & 1 \leq i \leq \mu \\ 0, & \mu < i \leq \lambda \end{cases} \tag{7}$$

Aus empirischen Befunden und der Analyse einfacher Zielfunktionen wird ein Verhältnis $\mu/\lambda = 1/7$ empfohlen [Sch95].

**Turnier-Selektion:** Es werden gleichverteilt zufällig $q$ Individuen aus $P''(g)$ gezogen. Das beste der $q$ Individuen wird selektiert. Dieser Vorgang wird $\lambda$-fach wiederholt. Es gilt:

$$p_{s_i} = 1/\lambda^q \cdot ((\lambda - i + 1)^q - (\lambda - i)^q). \tag{8}$$

In der Regel werden Turniergrößen $q = 2$ verwendet.

Das Konvergenzverhalten der Algorithmen wird wesentlich durch den das jeweilige Verfahren charakterisierenden Selektionsdruck bestimmt. Zur groben Orientierung kann gesagt werden, daß die $(\mu, \lambda)$-Selektion den größten und die proportionale Selektion den kleinsten Selektionsdruck der drei Verfahren ausüben [Bäc96].

Die optimale Wahl der Populationsgrößen ist abhängig von der Dimensionalität des Suchraumes $X$, der "Gutartigkeit" der Zielfunktion $F$, der verwendeten Operatoren und der Verfügbarkeit von Rechnerresourcen. Nachfolgende Aussagen dienen lediglich der Orientierung:

1. Geringer Selektionsdruck führt zu einer eher volumenorientierten Suche, hoher Selektionsdruck zu einer pfadorientierten Suche.

2. Der Selektionsdruck steigt proportional zum Verhältnis $\lambda/\mu$.

3. Kleine konstante Mutationsraten erforden große Populationen zum Erhalt der Diversität.

4. Die Selbstanpassung von Strategievariablen erfordert eine minimale Populationsgröße [Kur95].

Einen wesentlichen Einfluß hat auch die Größe der Grundmenge $[P''(g) \cup R(g)]$ (Algorithmus 1), aus der selektiert wird. Mit $R(g) = P(g)$ kann (unter Verwendung eines geeigneten Selektionsverfahrens) die Monotonie des Suchprozesses sichergestellt werden: Aus $P^F(g) = (I^1, \ldots, I^{|P(g)|})$ und $P^F(g+1) = (L^1, \ldots, L^{|P(g+1)|})$ folgt $F(\overline{\Gamma}(I^1)) \geq F(\overline{\Gamma}(L^1))$. Selektionsverfahren, die immer die bisher besten gefundenen Lösungen in der Population bewahren, heißen "elitär". Die elitäre Variante der $(\mu, \lambda)$-Selektion wird als $(\mu+\lambda)$-Selektion bezeichnet.

Augenscheinlich ist die elitäre Selektion von Vorteil. Solche Algorithmen sind mit $g \to \infty$ in der Regel global konvergent [Rud96]. Einerseits ist dieses Resultat aber in der Praxis für $g \ll \infty$ nicht sonderlich relevant. Andererseits verursacht elitäre Selektion häufig vorzeitige Konvergenz, da es schwieriger wird die Attraktionsgebiete lokaler Optima wieder zu verlassen. Im Falle gestörter, stochastischer oder dynamischer Zielfunktionen versagen solche Algorithmen häufig vollständig. Zudem reduziert die elitäre Selektion die Selbstanpassungsfähigkeit des Prozesses [Sch92]. Das Prinzip der Alterung, also die Möglichkeit veraltete Information zu vergessen, ist offensichtlich nicht nur in der belebten Natur von entscheidender Bedeutung für die Effizienz des Evolutionsprozesses.

# 4 Adaptivität evolutionärer Algorithmen

Ist die Funktion $F$ wenigstens lokal kausal (kleine Parametervariationen haben kleine Wirkung), so werden wir erwarten dürfen, in der Nähe guter Lösungen weitere gute, möglicherweise bessere, Lösungen zu finden. Ein effizientes Verfahren muß daher die lokale Topologie der Zielfunktion berücksichtigen. Traditionelle numerische Verfahren versuchen dies auch, etwa durch iterative Approximation des Gradienten und der zweiten partiellen Ableitungen. EAs gehen hier einen Schritt weiter, indem topologische Information in den Individuen, also verteilt über die ganze Population gespeichert und gewissermaßen kollektiv „erlernt“ wird [Sch97]. Wir betrachten in diesem Abschnitt zwei Beispiele.

Der Vorteil, der durch die Selbstanpassung der genetischen Information erzielt werden kann, wird allerdings durch einen zusätzlichen Aufwand für das „Erlernen“ dieses internen Modells der Zielfunktion erkauft und macht sich typischerweise erst nach einer größeren Zahl von Generationen bemerkbar. Zuvor stellen wir daher noch eine einfache pfadorientierte Variante der Evolutionsstrategie, die (1+1)-ES vor, welche auch dann gute Resultate liefern kann, wenn nur eine vergleichsweise geringe Zahl von Zielfunktionsauswertungen realisierbar ist.

**(1+1)-ES:** Ursprünglich für die experimentelle Optimierung entwickelt [Sch68], wird die (1+1)-ES heute häufig als robustes pfadorientiertes Suchverfahren eingesetzt, wenn aufgrund technischer Randbedingunen nur eine geringe Zahl von Zielfunktionsauswertungen realisierbar sind. Aus einem Elternindividuum wird per normalverteilter additiver Mutation (siehe Abschnitt 3.3) ein Nachkomme erzeugt. Der bessere von beiden wird selektiert. Für konvexe Zielfunktionen kann gezeigt werden, daß ein maximaler lokaler Fortschritt erreicht wird, wenn im Mittel jede fünfte Mutation erfolgreich ist (sog. 1/5-*Erfolgsregel*) [Rec94]. Bei Unterschreitung der Erfolgsrate wird die Schrittweite verringert, bei Überschreitung vergrößert. Die Schrittweite wird hier also exogen auf der Basis des lokalen Fortschritts gesteuert.

Für den linearen Fall kann gezeigt werden, daß die (1+1)-ES mit einer Konvergenzgeschwindigkeit von $1/\sqrt{n}$ deutlich effizienter arbeitet als eine Gradientenstrategie [Rec94].

**$(\mu, \lambda)$-ES:** Die endogene Anpassung der Schrittweiten $\sigma_i$ wird in $(\mu, \lambda)$-Evolutionsstrategien erreicht, indem die $\sigma_i$ selbst als *Strategieparameter* in den Individuen gespeichert und ebenfalls dem Evolutionsprozeß unterworfen werden. Der Mutationsoperator wird wie folgt modifiziert:

$$\begin{aligned} \sigma_i' &= \sigma_i \cdot \exp[\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)] \\ I_i' &= I_i + \sigma_i' \cdot N_i(0,1) \end{aligned} \tag{9}$$

Die Notation $N_i(0,1)$ zeigt an, daß die Zufallszahlen für jede Komponente $\sigma_i$ neu gezogen werden. Für die sogenannten Lernraten $\tau$, $\tau'$ wird $\tau \propto (\sqrt{2n})^{-1}$ und $\tau' \propto (\sqrt{2\sqrt{n}})^{-1}$ empfohlen [Sch95].

Zusätzlich zur Adaption der Schrittweiten schlägt Schwefel [Sch95] vor, auch die räumliche Orientierung der Mutationshyperellipsoide endogen anzupassen. Rudolph [Rud92] konnte allerdings nachweisen, daß dieses Vorgehen eine mit der Problemdimension $n$ quadratisch wachsende Populationsgröße erfordert und daher nur für vergleichsweise kleine Problemdimensionen geeignet ist. Für kleinere Populationsgrößen mit ähnlicher Skalierung der Objektvariablen empfiehlt sich gelegentlich die Vereinfachung des Mutationsschemas (9), indem nur eine einzelne globale Schrittweite $\sigma$ endogen adaptiert wird [Sch95].

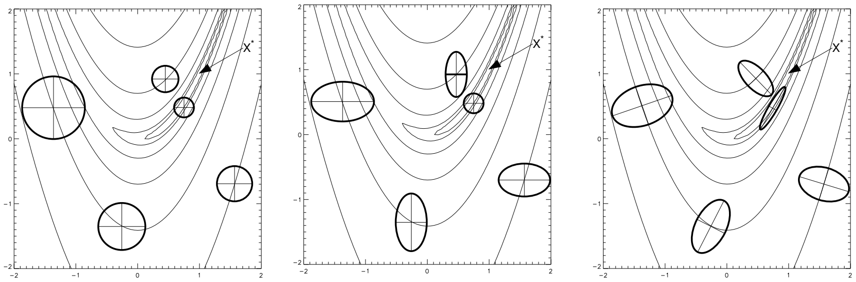

Abbildung 2: Höhenlinien der Funktion nach Rosenbrock (siehe [Sch95]) und die Effekte selbstadaptiver Mutationsoperatoren mit einer Schrittweite (links), $n$ Schrittweiten (mitte) und variablen Kovarianzen (rechts). $X^*$ bezeichnet das globale Optimum.

Die Effekte verschieder Mutationsoperatoren werden in Abbildung 2 skizziert. Mit nur einer globalen Schrittweite liegen gleichwahrscheinliche Mutationsereignisse auf dem Rand einer Hyperkugel. Bei $n$ Schrittweiten entartet die Hyperkugel zu einem achsenparallelen Hyperellipsoid. Durch Mutation der Kovarianzen der freien Variablen kann auch die räumliche Orientierung der Hyperellipsoide an die Topologie der Zielfunktion angepaßt werden.

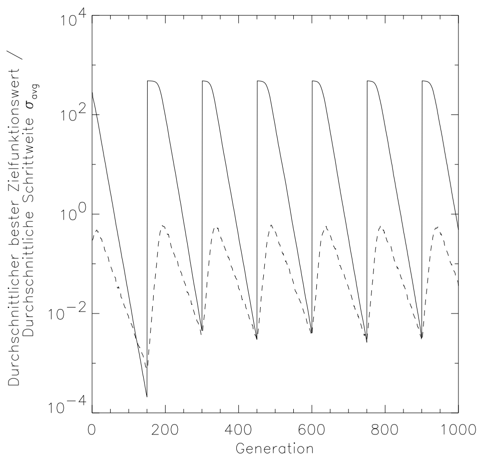


Abbildung 3: Experimentelle Überprüfung der Schrittweitenanpassung (gestrichelt) anhand der einfachen quadratischen Funktion $f(\vec{x}) = \sum_{i=1}^{n} x_i^2$.

Abbildung 3 zeigt das Ergebnis eines Experimentes mit einer (15,100)-Evolutionsstrategie und einer selbstadaptiven Schrittweite angewandt auf die quadratische Funktion $f(\vec{x}) = \sum_{i=1}^{n} x_i^2$. Das Optimum der Zielfunktion wird alle 150 Generationen verschoben. Deutlich ist der lineare Konvergenzverlauf wie auch die verzögerte Anpassung der mittleren Schrittweite zu erkennen.

**Linkage-Learning:** Wie bereits in Kapitel 3.1 ausgeführt wurde, ist die Wahl der Repräsentation wesentlich für die Konvergenz eines EA. Dabei ist der Einfluß der Kodierung auf die Konvergenz wiederum abhängig von den verwendeten Operatoren. Es ist z.B. unmittelbar klar, daß das Verhalten des Algorithmus unter $z-$Point-Crossover von der Anordnung der Gene abhängt, da Schemata großer definierender Länge selbst hoher Fitneß durch Crossover mit höherer Wahrscheinlichkeit zerstört werden, als jene mit kleinerer definierender Länge.

Aus diesem Grunde gehen zahlreiche Bemühungen dahin, den Einfluß der Gen-Anordnung zu verringern. Ein Ansatz ist die Entwicklung von Crossover-Varianten, die keinerlei Lokalität der Gene berücksichtigen wie das Uniform-Crossover (Kapitel 3.2). Andere Ansätze zielen darauf ab, die Gen-Anordnung während des Suchprozesses zu adaptieren. Stellvertretend sei hier der *Linkage Learning Genetic Algroithm* (LLGA) [Har97] skizziert, der wiederum auf Arbeiten von Goldberg et al. (*Messy Genetic Algorithm* [GKD89]) beruht.

Gegenüber dem Standard-GA werden folgende Modifikationen eingeführt:

- Jedes Gen besteht aus einem Tupel $(r, p)$, wobei $r$ einen Wert und $p$ die Bedeutung von $r$, also etwa die Position in einem Positions-Code wie in Formel 5 beschreibt.
- Um Randeffekte zu vermeiden, wird ein Chromosom als Ring aufgefaßt.
- Es wird ein 2-Punkt-Crossover verwendet.

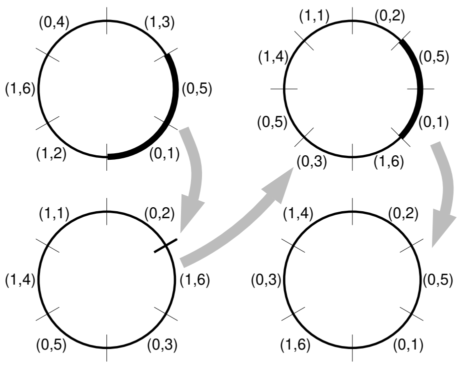


Abbildung 4: Rekombination beim Linkage-Learning: Zwei Chromosome (linke Seite) werden rekombiniert (rechts oben). Aus dem Produkt werden überzählige Gene entfernt (rechts unten).

Die Rekombination verläuft , wie in Graphik 4 für eine binäre Repräsentation dargestellt, in zwei Phasen: Zunächst wird aus einem Chromosom ein zufällig gewähltes Kreissegment herausgeschnitten und in ein zweites Chromosom an einer zufällig gewählten Stelle eingesetzt. Offenbar wird das neue Chromosom damit überbestimmt indem es für gleiches $p$ mehrere Tupel mit unterschiedlichem $r$ enthalten kann. Diese überzähligen Tupel werden aus dem neuen Chromosom entfernt.

Mit diesem Mechanismus ist der LLGA in der Lage, die Gen-Anordnung zu variieren, und es besteht die Hoffnung, daß sich, ähnlich wie im Falle der Schrittweitensteuerung bei der Mutation, Individuen mit günstiger Gen-Anordung aufgrund ihres Selektionsvorteils durchsetzen. Harik [Har97] konnte zeigen, daß sich dieser Effekt unter günstigen Voraussetzungen auch einstellt. Unter schlechteren Randbedingungen konvergierte der LLGA aber bereits zu einem lokalen Optimum bevor eine günstige Gen-Anordnung erlernt werden konnte. Harik führt dies auf eine zu starke diversiätsverringernde Selektion zurück.

Um dem Diversitätsverlust entgegenzuwirken, wird eine probabilistische Interpretation der Chromosomen eingeführt: Jedes Chromosom enthält mehrere Gene mit gleichem $p$. Ausgehend von einem beliebig gewählten Startpunkt wird das Chromosom im Uhrzeigersinn abgelesen, bis für jedes $p$ genau ein $x$ ermittelt wurde.

Tatsächlich wird durch diese Maßnahme der Diversitätsverlust soweit reduziert, daß der LLGA günstige Gen-Positionen erlernen kann.

# 5 Eigenschaften evolutionärer Algorithmen

Zur Beurteilung der Leistungsfähigkeit evolutionärer Algorithmen sind insbesondere Aussagen hinsichtlich der Aspekte *Konvergenzgeschwindigkeit* und *Konvergenzsicherheit* von Interesse. Bei der Herleitung von Konvergenzsätzen steht man vor dem Dilemma, daß entweder restriktive Annahmen über die Zielfunktion $F$ (z.B. Konvexität) oder über den Algorithmus (z.B. $|P(g)| \to \infty$ oder $g \to \infty$) getroffen werden müssen. Folglich sind diese Aussagen nur von begrenztem praktischen Nutzen und können den experimentellen Entwurf und die Validierung der Algorithmen nicht ersetzen. Andererseits können aber Mindestanforderungen formuliert werden wie z.B. die lineare Konvergenz im Falle konvexer Zielfunktionen.

Unter der Konvergenzgeschwindigkeit $\varphi$ versteht man den Erwartungswert der Veränderung des Abstands zum Optimum, meist bezogen auf den Schwerpunkt der Population, zwischen der aktuellen und der nachfolgenden Generation des Algorithmus. Das Abstandsmaß kann dabei sowohl auf die Objektvariablenvektoren (z.B. Euklidischer Abstand, falls $X \subseteq I\!R^n$) als auch auf die Zielfunktionswerte bezogen sein. Die Konvergenzgeschwindigkeit kann als Maß für die lokalen Sucheigenschaften eines Algorithmus verstanden werden, da sie bisher nur für unimodale Zielfunktionen theoretisch untersucht werden konnte und ihre Maximierung zu einer gradientenorientierten Suche führt. Insbesondere für kontinuierliche, konvexe Zielfunktionen liegen umfangreiche Resultate zur Konvergenzgeschwindigkeit von Evolutionsstrategien vor. Für die einfache (1+1)-Strategie ergibt sich die maximale Konvergenzgeschwindigkeit im konvexen Fall zu $0.2 \cdot R/n$, wobei $R$ den aktuellen Abstand zum Optimum angibt [Rec94].

Aufbauend auf den Arbeiten von Schwefel zur (1,$\lambda$)-Strategie [Sch95] sind heute alle Strategievarianten der Evolutionsstrategie auf konvexen Zielfunktionen gut untersucht und erlauben ein gutes Verständnis der lokalen Sucheigenschaften dieser Verfahren. So ergibt sich für die (1,$\lambda$)-Strategie das Resultat [Bey93, Rec94]

$$\varphi'_{(1,\lambda)} = c_{1,\lambda}\sigma' - \frac{\sigma'^2}{2} \quad , \tag{10}$$

wobei $\varphi' = \varphi \cdot n/R$ und $\sigma' = \sigma \cdot n/R$. Die einfache quadratische Form der Gleichung macht deutlich, daß sich der Fortschritt aus positiven und negativen Anteilen zusammensetzt und eine Annäherung an das Optimum nur unter Einhaltung der *Evolutionsbedingung* $\sigma' < 2c_{1,\lambda}$ möglich ist. Die nur von der Populationsgröße abhängige Größe $c_{1,\lambda}$ wird als *Fortschrittsbeiwert* [Rec94] oder *Selektionsintensität* [MSV93a] bezeichnet und charakterisiert die Stärke der Selektion. Die maximale Konvergenzgeschwindigkeit ergibt sich aus Gleichung (10) unter Verwendung der Näherung $c_{1,\lambda} \approx \sqrt{2\ln\lambda}$ zu $\varphi' = c_{1,\lambda}^2/2 \approx \ln\lambda$.

Im Falle einer ($\mu$,$\lambda$)-Strategie *ohne* Rekombination verallgemeinert sich Gleichung (10) zu [Bey94, Rec94]

$$\varphi'_{(\mu,\lambda)} = c_{\mu,\lambda}\sigma' - \frac{\sigma'^2}{2} \quad , \tag{11}$$

woraus sich mit $c_{\mu,\lambda} \approx \sqrt{2\ln(\lambda/\mu)}$ eine maximale Konvergenzgeschwindigkeit von $\varphi' \approx \ln(\lambda/\mu)$ ergibt.

Die Verallgemeinerung der Theorie auf den Fall $\mu > 1$ mit Mutation *und* Rekombination ergibt für globale ($N = |P|$), intermediäre Rekombination das Resultat [Bey95, Rec94]

$$\varphi'_{(\mu/\mu_I,\lambda)} = c_{\mu,\lambda}\sigma' - \frac{\sigma'^2}{2\mu} \tag{12}$$

sowie für diskrete Rekombination (Uniform-Crossover) das Resultat [Bey95, Rec94]

$$\varphi'_{(\mu/\mu_D,\lambda)} = \sqrt{\mu}c_{\mu,\lambda}\sigma' - \frac{\sigma'^2}{2} \quad , \tag{13}$$

wobei sich in beiden Fällen eine maximale Konvergenzgeschwindigkeit von $\varphi' = \mu c^2_{\mu,\lambda}/2 \approx \mu \cdot \ln(\lambda/\mu)$ ergibt. Beide Varianten der Rekombination erzielen also im konvexen Fall eine Konvergenzbeschleunigung um den Faktor $\mu$, so daß hiermit ein theoretisch fundiertes Argument für die Nutzung eines Rekombinationsoperators vorliegt. Für alle vorgestellten Varianten der Evolutionsstrategie kann auf der Basis dieser Resultate gezeigt werden, daß im konvexen Fall für eine optimal eingestellte Schrittweite *lineare Konvergenz* erzielt wird, d.h. es gilt $R_g \leq c \cdot \theta^g$ mit $0 \leq \theta < 1$ für den Abstand $R_g$ zum Optimum $x^*$ in Generation $g$ [BRS93, Bey96, Sch95]. Diese Resultate verdeutlichen, daß Evolutionsstrategien eine Konvergenzgeschwindigkeit erzielen, die mit der spezialisierter lokaler Verfahren vergleichbar ist, d.h., die Effizienz dieser Variante evolutionärer Algorithmen ist damit nachgewiesen.

Im Bereich der genetischen Algorithmen gibt es Resultate zur Konvergenzgeschwindigkeit erst seit wenigen Jahren, als Resultat der Übertragung des theoretischen Ansatzes der Analyse von Evolutionsstrategien auf diskrete Suchräume [Bäc92, Müh92]. Das Hauptaugenmerk der Theorie genetischer Algorithmen liegt traditionell auf der durch das sogenannte *Schema-Theorem* (siehe Abschnitt 3.1) formalisierten Analogie zu einem spieltheoretischen Entscheidungsproblem [Hol75]. Dabei wird gezeigt, daß unter der Voraussetzung unvollständiger Information über die Qualität gewisser Kombinationen von Variableneinstellungen die optimale Strategie darin besteht, die beobachteten besten Kombinationen mit exponentiell wachsender Häufigkeit zu testen — eine Strategie, die durch genetische Algorithmen realisiert wird. Seit einiger Zeit wird in Frage gestellt, ob sich die der spieltheoretischen Analogie zugrundeliegenden Optimalitätskriterien mit der Fragestellung der globalen Optimierung in Einklang bringen lassen [Fog95]. Damit einher geht eine Neuorientierung hinsichtlich der Analysetechniken, wobei insbesondere den Markov-Ketten eine tragende Rolle zukommt [Vos95, Whi95]. Die Frage nach der Konvergenzgschwindigkeit (bzw. Absorptionszeit) ist im Zusammenhang mit Markov-Ketten erheblich naheliegender und theoretisch handhabbarer als im Kontext der Schema-Analyse.

Bisher am besten untersucht ist die sogenannte „counting ones" Funktion $f(x) = \sum_{i=1}^{\ell} x_i,\ x_i \in \{0,1\}$, mit Resultaten für Mutations-Selektions Algorithmen (d.h., ohne Rekombination) [Bäc96, Bey97] sowie für Rekombinations-Selektions Algorithmen (d.h., ohne Mutation) [MSV93a, MSV93b, MG96]. Die Mutationsrate $p = 1/\ell$ ist für das „counting ones" Problem eine gute Wahl und führt für einen (1+1)-GA zu einer Zeitkomplexität von $\mathcal{O}(\ell \cdot \ln \ell)$ [Müh92]. Die Verwendung von Rekombination anstelle von Mutation reduziert hingegen die Zeitkomplexität auf $\mathcal{O}(\sqrt{\ell})$, wobei allerdings eine hinreichend große Population vorauszusetzen ist [MSV93a, MG96].

Die *Konvergenzsicherheit* eines Algorithmus ist dann gewährleistet, wenn theoretisch nachgewiesen werden kann, daß der Algorithmus zu gegebener Zielfunktion $F : M \rightarrow I\!R$ eine global optimale Lösung $x^*$ liefert. Der Nachweis dieser Eigenschaft, auch *globale Konvergenz mit Wahrscheinlichkeit Eins* genannt, ist bestenfalls unter der Voraussetzung unendlicher Laufzeit möglich — einer Aussage, die wiederum nur theoretisch von Interesse ist. Derartige Konvergenzresultate liegen heute für eine Vielzahl von Varianten evolutionärer Algorithmen vor [Suz95, Rud94a, BRS93].

Praktische Untersuchungen zur Konvergenzsicherheit müssen daher generell auf einem experimentellen Vergleich verschiedener Algorithmen auf einer Vielzahl von multimodalen Zielfunktionen basieren, um zu für den Anwender nutzbaren Aussagen zu gelangen. Eine umfangreiche derartige Bewertung von Evolutionsstrategien liegt seit mehr als zwanzig Jahren vor und hat eindeutige Resultate zugunsten dieser Algorithmen erbracht [Sch95]. Praktische Erfahrungen mit allen Varianten evolutionärer Algorithmen haben zu unabhängigen Bestätigungen geführt, so daß deren globale Sucheigenschaften heute weithin anerkannt sind und in einer Vielzahl von Anwendungen genutzt werden.

# 6 Spezielle Techniken

## 6.1 Parallelisierung

In der belebten Natur laufen evolutionäre Prozesse hochgradig parallel ab. Auch EAs besitzen eine inhärente Parallelität, da die Reihenfolge der Zielfunktionsauswertungen für die Individuen der Population $P(g)$ die Konvergenz des Algorithmus nicht beeinflußt. Zudem lassen sich die Verfahren durch Variation der Populationsgröße bezüglich der verfügbaren Hardware leicht skalieren. In den meisten Anwendungen kann der Kommunikationsaufwand zwischen den parallelen Prozessen gegenüber den Kosten für die Zielfunktionsberechnung vernachlässigt werden. Das gilt insbesondere für zeitaufwendige Simulationsläufe. Daher eignen sich neben Parallelrechnern auch Rechnernetze zur Parallelisierung des Suchprozesses [BBNH95].

Die meisten traditionellen direkten Optimierungsalgorithmen (vergleiche [Rao96, Kap. 6] und [Sch95, Kap. 3.2]) sind dem Wesen nach sequentiell. Als einfache Parallelisierungsansätze werden gelegentlich mehrere unabhängige Suchprozesse mit unterschiedlicher Initialisierung gestartet, oder es werden, falls es der Algorithmus erlaubt, einzelne Tastschritte parallel durchgeführt. Im ersten Ansatz wird aufgrund der vollständigen Isolation der Suchprozesse Information verschenkt. Der erreichbare Parallelisierungsgrad durch die gleichzeitige Ausführung einzelner Tastschritte ist je nach Algorithmus stark beschränkt und zumeist deutlich kleiner als die Problemdimension.

Zudem verwenden die meisten Verfahren zentrale Datenstrukturen wie Transformationsmatrizen oder Approximationen der ersten oder zweiten partiellen Ableitungen. In EAs besitzen die Individuen einer Population dagegen keine gemeinsame Datenstruktur. Vielmehr sind topologische Informationen implizit über die gesamte Population verteilt in den Strategievariablen gespeichert. Allein der Selektionsschritt kann die Ausnutzung der Parallelität begrenzen, da je nach Operator entweder die gesamte Population oder zumindest eine gewisse Teilmenge davon bereits bewertet worden sein muß (siehe Abschnitt 3.3).

Dieser Umstand motivierte die Suche nach Selektionsoperatoren, die eine bessere Ausnutzung paralleler Architekturen erlauben. Im folgenden werden einige gebräuchliche Verfahren aufgezählt.

**Steady-State Selektion:** Dieses Selektionsverfahren wird häufig zusammen mit einer Master-Slave Implementierung benutzt. Der Master-Prozeß implementiert den gesamten EA. Allein die Zielfunktionsauswertung wird durch parallele Slave-Prozesse realisiert. Jedes neu bewertete Individuum $I'$ ersetzt unmittelbar das Individuum mit der schlechtesten Fitneß $I^{|P(g)|}$ in der Population $P^F(g)$, falls $F(\overline{\Gamma}(I')) \leq F(\overline{\Gamma}(I^{|P(g)|}))$.

Die Steady-State Selektion, gelegentlich auch als $(\mu+1)$-Selektion bezeichnet, eignet sich vor allem in Netzen mit wenigen Knoten und aufwendigen Zielfunktionsberechnungen. Sie führt zu einer pfadorientierten Suche [SJ97, BBNH95].

**Migrationsmodelle:** Jedem Prozessor wird eine Subpopulation zugeordnet. Die Evolution der Subpopulationen erfolgt asynchron. Die Populationen sind bezüglich einer Nachbarschaftsrelation geordnet. In regelmäßigen Abständen (Epochen) werden Teilmengen der Subpopulationen zwischen den unmittelbaren Nachbarn ausgetauscht.

Migrationsmodelle stellen einen grobkörnigen Ansatz zur Parallelisirung dar. Sie werden häufig in Rechnernetzen mit einer mittleren Anzahl von Knoten verwendet. Typischerweise werden vergleichsweise kleine Populationen $|P(g)| < 100$ auf einem Ring angeordnet. Im Abstand von 10-50 Generationen werden die aktuell 1-5 besten Individuen mit den Nachbarpopulationen ausgetauscht [MLC97].

**Diffusionsmodelle:** Auf der gesamten Population wird eine Nachbarschaftsrelation definiert. Rekombination und Selektion sind bezüglich der lokalen Nachbarschaften der Individuen definiert.

Diffusionsmodelle erlauben eine feinkörnige Parallelisierung und eignen sich daher auch für eng gekoppelte Systeme mit einer großen Anzahl von Prozessoren. Als Nachbarschaftsrelation wird zumeist ein äquidistantes Gitter oder ein Torus verwendet. Der Parallelisierungsgrad ist dann nur durch die Anzahl der Gitterpunkte beschränkt. Die Entfernung zwischen zwei Individuen wird als minimaler Pfad (Anzahl der Kanten) zwischen den Individuen definiert. Selektion und Rekombination operieren auf kleinen Nachbarschaften der Größe 4-8. Zumeist werden große Populationen ($> 100$) verwendet [Pet97, Spr94].

## 6.2 Zeitvariante Zielfunktionen

In den vorangegangenen Betrachtungen bestand die Aufgabe in dem Auffinden einer (globalen) Optimalstelle des statischen Optimierungsproblems (1). Wird die Zielfunktion aber durch zeitlich variable Größen beeinflußt, so können auch die Optimalstellen der Funktion variieren. Vorausgesetzt, daß die Veränderung der Optimalstellen nicht zu sprunghaft erfolgt, ist es einigen Varianten evolutionärer Algorithmen (insbesondere mit nichtelitärer

Selektion) möglich solchen Veränderungen zu folgen [Bäc98, Ang97b]. Ein Beispiel findet sich auf Seite 13.

EAs werden daher gelegentlich auch zur Implementierung selbstadaptiver Regler verwendet, wenngleich die überwiegende Zahl der Publikationen im Bereich *evolutionary Control* den Entwurf linearer PID-Regler oder nichtlinearer Regler z.B. auf der Basis von Fuzzy-Regelmengen oder neuronaler Netze beschreibt [McD97].

## 6.3 Stochastische Zielfunktionen und Robust Design

Wie bereits in Abschnitt (1) erwähnt führen nichtdeterministische Simulationsmodelle wie im Falle ereignisorientierter Simulationen zu stochastischen Optimierungsaufgaben. Gleiches gilt auch in der experimentellen Optimierung, wenn Parameter- oder Zielfunktionswerte mit einer Ungenauigkeit behaftet sind.

Den Autoren sind nur wenige anwendungsorientierte Veröffentlichungen über EAs als Verfahren zur Lösung stochastischer Optimierungsaufgaben bekannt (siehe z.B. [HBH92, NB95]). Allerdings liegen einige theoretische und empirische Arbeiten zur Robustheit evolutionärer Algorithmen vor [FG88, Bey93, Rec94, HB94]. Danach ergibt sich etwa folgendes Bild:

- Die Konvergenz des EA wird solange nicht beeinträchtigt, wie die Ungenauigkeit der Schätzung des Erwartungswertes in Gleichung (4) klein ist gegenüber dem Erwartungswert selbst. In der Praxis ist zumeist eine Standardabweichung der Schätzung von 10% des Erwartungswertes noch ohne Einfluß, vorausgesetzt, daß die Schätzung einer Normalverteilung genügt. Größere Standardabweichungen können den Suchprozeß verlangsamen und schließlich stoppen. Das Optimum wird nur noch mit einer Ungenauigkeit (proportional zur Größe der Standardabweichung bei konvexen Zielfunktionen) lokalisiert.

- Die Erhöhung der Populationsgröße bringt dann in der Regel nur eine marginale Verbesserung des Konvergenzverhaltens. Daher ist es vorzuziehen, die Schätzung durch Vergrößerung des Stichprobenumfangs zu verbessern [Bey93, HB94].

- Rekombination kann einen superlinearen Speed-Up (in Bezug auf die Größe der Elternpopulation) bewirken. Die Rekombination scheint gerade in „verrauschten" Suchräumen von großem Nutzen zu sein [Rec94, Kapitel 14].

Die Unempfindlichkeit evolutionärer Algorithmen gegenüber stochastischen Einflüssen kann zur Beschleunigung des Suchprozesses ausgenutzt werden, indem fallweise Zielfunktionswerte unter Verwendung schneller zu berechnender Ersatzzielfunktionen oder einfacherer Modelle approximiert werden [PP98, GAP+96].

Die Umsetzung von unter idealisierten Bedingungen simulativ gewonnenen Lösungen in reale Produkte und Verfahren erfordert zumeist die Berücksichtigung von Randbedingungen wie Fertigungstoleranzen, Verschleiß oder die Ausfallwahrscheinlichkeiten von Komponenten. Daher ist der Praktiker weniger an den theoretisch erreichbaren Optima, als vielmehr an robusten Lösungen interessiert. Solche Randbedingungen sollten bereits im Entwurf des Simulationsmodells oder der Zielfunktion berücksichtigt werden. Ihr Einfluß kann zumeist in Form stochastischer Störungen modelltechnisch erfaßt werden. Daher bieten sich auch hier EAs als Suchverfahren an [WHB98].

## 6.4 Multikriterielle Optimierung

In praktischen Aufgaben ist es mitunter notwendig mehrere häufig gegenläufige Ziele (z.B. minimale Kosten versus maximale Haltbarkeit) zu berücksichtigen. Die naheliegende Lösung, die Zielfunktion als gewichtete Summe der Teilziele zu definieren, erfordert a priori eine (subjektive) Gewichtung der Teilziele. Zudem wird während des Suchprozesses Information über die Bewertung einzelner Lösungen gegenüber den Teilzielen verschenkt. Ändern sich die Gewichtungen etwa aufgrund der Ergebnisse der Suche, so ist der Suchprozess neu zu starten.

Alternativ wird daher bei Aufgaben mit mehrfacher Zielsetzung auch versucht, die sogenannte *Pareto-Menge* zu approximieren. Das sind jene Lösungen, für die eine Verbesserung hinsichtlich eines Teilziels nur auf Kosten der Verschlechterung bezüglich eines anderen Teilziels erreicht werden kann.

Hauptsächlich durch Modifikation der Selektion und der Populationsstruktur lassen sich EAs auch multikriteriellen Optimierungsaufgaben anpassen [FF97].

## 6.5 Optimierung unter Nebenbedingungen

Bislang haben wir zugunsten einer kompakteren Darstellung die Behandlung einer wichtigen Klasse von Optimierungsaufgaben, nämlich restriktionsbehaftete Probleme, vernachlässigt. Für $X \subseteq I\!R^n$ können solche Propleme wie folgt allgemein notiert werden:

$$\begin{array}{ll} F(x) \to \min, & x = (x_1, \ldots, x_n) \in X \\ g_i(x) \leq 0, & i = 1, \ldots, p \end{array} \tag{14}$$

Um die Einhaltung der Restriktionen $g_i$ zu erzwingen, wurden für EAs eine Reihe von Modifikationen vorgeschlagen, die hier nur kurz skizziert werden sollen. Eine umfassende Darstellung findet sich in [BFM97, Kapitel 5.1].

**Straffunktionen:** Die Zielfunktion wird mit einem von der Stärke der Restriktionsverletzungen abhängigen Strafterm beaufschlagt.

**Dekoder:** Durch geschickte Wahl der Repräsentation $\mathcal{R}$ und der Dekodierungsfunktion $\overline{\Gamma}$ kann unter Umständen die Einhaltung der Restriktionen erzwungen werden.

**Restriktionserhaltende Operatoren:** Dieser Ansatz entspricht dem vorhergehenden, nur werden die Variationsoperatoren so entworfen, daß aus zulässigen Lösungen keine unzulässigen Lösungen generiert werden können.

**Reparaturmechanismen:** Auf Genotyp-Ebene werden zusätzliche Operatoren definiert, die unzulässige Lösungen in zulässige Lösungen überführen.

**Co-Evolution:** Dieser Ansatz ist durch biologische Vorbilder co-evolvierender Populationen (z.B. Parasiten und Wirte) motiviert. Die erste Population enthält Individuen aus der gesamten Repräsentation $\mathcal{R}$. Deren Fitneß wird gemäß einer Straffunktion bestimmt. Die zweite Population besteht aus den Restriktionen. Deren Fitneß wird aus der Anzahl der Individuen aus der ersten Population bestimmt, die diese Restriktionen verletzen. Restriktionen, die häufiger verletzt werden, erhalten damit eine stärkere Gewichtung.

## 6.6 Hybridisierung

Zur Steigerung der Effizienz und Exaktheit der Lokalisierung von Optimalstellen wurden verschiedentlich Kombinationen der evolutionären Suche mit lokalen Suchverfahren (z.B. Gradientenstrategien, Greedy-Algorithmen), anderen allgemeinen Heuristiken (z.B. Tabu-Search, Simulated Annealing) und problemspezifischen Heuristiken vorgeschlagen [Iba97b].

In den letzten Jahren wurde das Potential von evolutionären Algorithmen zur subsymbolischen (numerischen) Wissensverarbeitung und damit die Verwandtschaft zu anderen Verfahren aus diesem Bereich, das sind insbesondere die Fuzzy-Logik (FL) und die künstlichen neuronalen Netze (NN), erkannt. Als Oberbegriffe haben sich *Computational Intelligence* (CI) sowie *Softcomputing* etabliert. Nach Bezdek zeichnen sich die Methoden der CI durch Adaptivität, Fehlertoleranz, Verarbeitungsgeschwindigkeiten vergleichbar mit (menschlichen) Kognitionsprossen und Optimalität von Fehlerraten aus [Bez94].

Die Kopplung von CI-Verfahren ist daher naheliegend und wurde bereits zahlreich mit Erfolg vorgenommen. Typische Ansätze sind die Steuerung von EAs durch FL (z.B. Populationsgrößen, genetische Operatoren usw.), der Entwurf von FL-Controllern mit EAs (Regelgenerierung und Optimierung der Zugehörigkeitsfunktionen) und die Optimierung von neuronalen Netzen mit EAs (EAs als Lernverfahren und zur Optimierung der Netzstruktur) [Sch94]. Einen umfassenden Überblick gibt [Bon97].

# 7 Anwendungsbeispiele

Die Publikationen über erfolgreiche Anwendungen evolutionärer Algorithmen sind ebenso umfangreich wie divers. Wir beschränken die Aufzählung in diesem Kapitel auf Anwendungen aus dem Bereich Simulationstechnik und verwandter Gebiete. Umfangreichere Bibliographien sind in [BFM97] und [Ala94] zu finden. Letztere enthält allein über 3000 Einträge!

**Optimierung und Simulation:** Veröffentlichungen über die Kopplung von EAs mit Simulationsmodellen sind wohl aus den in Kapitel 1 genannten Gründen nicht sehr zahlreich. Poloni und Pediroda [PP98] beschreiben die Optimierung von Tragflügeln auf der Basis eines Simulationsmodells und geben Hinweise zur Beschleunigung des Suchprozesses. Haupt [Hau98] erläutert die Regelung einer Kläranalage auf der Basis simulativ gewonnener Prognosen. In [BHL+98] werden unter anderem die simulative Optimierung von Beladeplänen für Druckwasserreaktoren sowie von verfahrenstechnischen Anlagen beschrieben. Allgemeinere Darstellungen finden sich in [Ham97, BHS93].

**Experimentelle Optimierung und interaktive Evolution:** EAs finden vorwiegend in sogenannten *Generate-and-Test*-Systemen Anwendung. Dabei muß die Testkomponente nicht zwingend eine numerische (rechnergestützte) Berechnung einer Zielfunktion beinhalten. Vielmehr kann die Bewertung und Selektion von Lösungen auch experimentell oder interaktiv durch den Benutzer geschehen. In der Tat wurde die $(1 + 1)$-Variante der Evolutionsstrategie (siehe Kapitel 4) für die diskrete Optimierung von Windkanalexperimenten entwickelt [Rec65, Sch68].

Interaktive Evolution wurde unter anderem erfolgreich zur Optimierung von Stoffgemischen [HH], zur Erstellung von Phantombildern [CJ91] und in der Architektur [Ros97] angewendet. Einen Überblick gibt [Ban97].

**Design- und Strukturoptimierung:** Solche Aufgaben zählen zu den technisch schwierigsten Optimierungsproblemen, da zumeist nicht einmal eine kanonische Problemrepräsentation existiert. Es wundert daher nicht, daß gerade in diesem Bereich EAs aufgrund ihrer Flexibilität sehr beliebt sind. Kost [Kos95] beschreibt z.B. die Topologieoptimierung von Tragwerken mit Evolutionsstrategien. Weitere typische Anwendungsfelder sind die Auslegung technischer Anlagen, die Optimierung von Verbundwerkstoffen, das VLSI-Design und der Datenbankentwurf [BFM97, DM97, Par96] (siehe auch [Rec94, Kap. 13]).

Im weiteren Sinne kann auch die automatische Synthese von Programmen, das sogenannte *Genetic Programming*, als Strukturoptimierungsproblem aufgefasst werden [Koz94].

**Regelungstechnik:** Wie bereits in Kapitel 6.2 erwähnt eignen sich EAs zur Konstruktion adaptiver Regler. Ein vielzitiertes Beispiel, die adaptive Reglung von Gas-Pipeline-Systemen auf der Basis von sogenannten Classifier-Systemen, stammt von Goldberg [Gol89, S. 288 ff.]. Fogel [Fog95, S. 188 ff.] beschreibt die Anwendung von EAs auf ein klassisches Benchmark-Problem, das invertierte Pendel. Die EA-basierte Regelung einer Kläranlage und eines Produktionsprozesses ist in [Hau98] dargestellt. Allgemeinere Darstellungen finden sich in [McD97] und [Dra97].

**Identifikation:** Unter Identifikation verstehen wir die Generierung von beschreibenden (deskriptiven) Modellen, die einen beobachteten Zusammenhang möglichst gut approximieren unter Verwendung einer möglichst einfachen Struktur (z.B. Polynom geringen Grades). Fogel [Fog91] beschreibt die Parmeterschätzung nichtlinearer Regressionsmodelle mit EP. Die Struktur- und Parameter-Optimierung von nichtlinearen deskriptiven Modellen ist eine typische Anwendung des Genetic Programming [Iba97a].

**Ökonomie:** Der Vollständigkeit halber seien auch die zahlreichen Anwendungen im Bereich der Ökonomie erwähnt. Dazu zählen Resource Management, Routenplanung, Lagerhaltungsstrategien, Produktionsplanung sowie diverse Arten von Scheduling-Problemen [Nis97, BN95].

# 8 Schlußbetrachtung

Evolutionäre Algorithmen sind keine universellen Problemlöser. Sie sind vielmehr universelle Suchheuristiken, die auch dann noch befriedigende Resultate liefern können, wenn keine aufgabenspezifischen Verfahren existieren oder deren Entwicklung einen unvertretbar hohen Aufwand erfordert. Sie sind in vielen Anwendungsbereichen (z.B. lineare oder quadratische Optimierung) traditionellen Verfahren völlig unterlegen. Ihre Stärken sind dagegen Flexibilität, Robustheit und Adaptivität.

Obgleich einige wichtige theoretische Resultate vorliegen ist heute die Wirkungsweise von EAs immer noch zu einem großen Teil unverstanden. Anwendungen werden nach dem Trial-and-Error-Prinzip entworfen. Ein wichtiges Ziel der EA-Forschung ist daher die Vervollständigung des theoretischen Fundamentes, die formale Abgrenzung des Anwendungsbereichs und die Entwicklung einer ingenieurtechnischen, konstruktiven Herangehensweise.

Durch die Synthese des evolutionären Rechnens mit anderen Ansätzen der Computational Intelligence ist zu hoffen, zu noch flexibleren, effizienteren, leichter handhabbaren und vielleicht sogar zu lernfähigen Verfahren zu gelangen.

Evolutionäre Algorithmen sind selbst Modelle natürlicher Prozesse und als solche starke Abstraktionen. Sie erheben nicht den Anspruch, die biologische Evolution vollständig beschreiben zu können. Andererseits birgt das natürliche Vorbild viele, bis heute noch nicht ausgeschöpfte Aspekte zur Verbesserung der Modelle und damit auch der Suchheuristiken. Schlagwortartig seien nur Diploidie (doppelte) bzw. Polyploidie (mehrfache Chromosomensätze), Co-Evolution mehrerer Spezies, Spezienbildung, Ontogenese (die Entwicklung vielzelliger Organismen aus eine Urzelle) und Epigenese (die Entwicklung aufgrund von Umwelteinflüssen) genannt.

# Literatur


[Ala94] J. T. Alander. An Indexed Bibliography of Genetic Algorithms: Years 1957–1993. Art of CAD Ltd, Espoo, Finland, 1994.

[Ang97a] P. J. Angeline. C 1.6 Parse Trees. In T. Bäck, D. Fogel, und Z. Michalewicz, Hrsg., *Handbook of Evolutionary Computation*, Seiten C1.6:1 – C1.6:3. Oxford University Press, New York, 1997.

[Ang97b] P. J. Angeline. Tracking Extrema in Dynamic Environments. In P. J. Angeline, R. G. Reynolds, J. R. McDonnell, und R. Eberhart, Hrsg., *Proceedings of the Sixth International Conference on Evolutionary Programming, EP97*, Band 1213 der Reihe *Lecture Notes in Computer Science*, Seiten 335–345. Springer, Berlin, 1997.

[Bäc92] Th. Bäck. The Interaction of Mutation Rate, Selection, and Self-Adaptation Within a Genetic Algorithm. In R. Männer und B. Manderick, Hrsg., *Parallel Problem Solving from Nature, 2*, Seiten 85–94. Elsevier, Amsterdam, 1992.

[Bäc96] Th. Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York, 1996.

[Bäc98] Th. Bäck. On the Behavior of Evolutionary Algorithms in Dynamic Environments. In *Proceedings of the Fifth IEEE Conference on Evolutionary Computation.* IEEE Press, Piscataway, NJ, 1998.

[Ban97] W. Banzhaf. C 2.9 Interactive Evolution. In Bäck et al. [BFM97], Seiten C2.9:1–C2.9:6.

[BBNH95] Th. Bäck, Th. Beielstein, B. Naujoks, und J. Heistermann. Evolutionary Algorithms for the Optimization of Simulation Models Using PVM. In J. Dongarra, M. Gengler, B. Tourancheau, und X. Vigouroux, Hrsg., *EuroPVM '95: Second European PVM Users' Group Meeting*, Seiten 277–282. Hermes, Paris, 1995.

[Bea93] J. C. Bean. Genetics and Random Keys for Sequences and Optimization. Interner Bericht 92–43, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI, 1993.

[Bey93] H.-G. Beyer. Towards a Theory of Evolution Strategies: Some Asymptotical Results from the $(1 \overset{+}{,} \lambda)$-Theory. *Evolutionary Computation*, 1(2):165–188, 1993.

[Bey94] H.-G. Beyer. Towards a Theory of Evolution Strategies: The $(\mu,\lambda)$-Theory. *Evolutionary Computation*, 2(4):381–408, 1994.

[Bey95] H.-G. Beyer. Towards a Theory of Evolution Strategies: On the Benefits of Sex — the $(\mu/\mu, \lambda)$-Theory. *Evolutionary Computation*, 3(1):81–111, 1995.

[Bey96] H.-G. Beyer. On the Asymptotic Behavior of Multirecombinant Evolution Strategies. In H.-M. Voigt, W. Ebeling, I. Rechenberg, und H.-P. Schwefel, Hrsg., *Parallel Problem Solving from Nature IV. Proceedings of the International Conference on Evolutionary Computation*, Band 1141 der Reihe *Lecture Notes in Computer Science*, Seiten 122–133. Springer, Berlin, 1996.

[Bey97] H.-G. Beyer. An Alternative Explanation for the Manner in which Genetic Algorithms Operate. *BioSystems*, 41:1–15, 1997.

[Bez94] J. C. Bezdek. What is Computational Intelligence ? In Zurada et al. [ZMR94], Seiten 1–12.

[BFM97] Th. Bäck, D. B. Fogel, und Z. Michalewicz, Hrsg. *Handbook of Evolutionary Computation.* Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997.

[BHL+98] Th. Bäck, U. Hammel, R. Lewandowski, M. Mandischer, B. Naujoks, S. Rolf, M. Schütz, H.-P. Schwefel, J. Sprave, und S. Theis. Evolutionary Algorithms: Applications at the Informatik Center Dortmund. In Quagliarella et al. [QPPW98], Kapitel 9, Seiten 175–204.

[BHS93] Th. Bäck, U. Hammel, und H.-P. Schwefel. Modelloptimierung mit evolutionären Algorithmen. In A. Sydow, Hrsg., *Simulationstechnik: 8. Symposium in Berlin*, Fortschritte in der Simulationstechnik, Seiten 49–57. Vieweg, Wiesbaden, 1993.

[BHS97] Th. Bäck, U. Hammel, und H.-P. Schwefel. Evolutionary Computation: History and Current State. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[BN95] J. Biethan und V. Nissen. *Evolutionary Algorithms in Management Applications*. Springer-Verlag, Berlin, 1995.

[Bon97] P. P. Bonissone. Soft Computing: The Convergence of Emerging Reasoning Technologies. *Soft Computing*, 1(1):6–18, 1997.

[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 Second Annual Conference on Evolutionary Programming*, Seiten 11–22. Evolutionary Programming Society, San Diego, CA, 1993.

[BS95] Th. Bäck und M. Schütz. Evolution Strategies for Mixed-Integer Optimization of Optical Multilayer Systems. In J. R. McDonnell, R. G. Reynolds, und D. B. Fogel, Hrsg., *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming*, Seiten 33–51. MIT Press, Cambridge, MA, 1995.

[BS96] Th. Bäck und M. Schütz. Intelligent Mutation Rate Control in Canonical Genetic Algorithms. In Z. W. Ras und M. Michalewicz, Hrsg., *Foundations of Intelligent Systems, 9th International Symposium, ISMIS '96*, Band 1079 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 158–167. Springer, Berlin, 1996.

[CJ91] C. Caldwell und V. S. Johnston. Tracking a Criminal Suspect through 'Face-Space' with a Genetic Algorithm. In R. K. Belew und L. B. Booker, Hrsg., *Proceedings of the Fourth International Conference on Genetic Algorithms*, Seiten 416–421. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[Dav91] L. Davis, Hrsg. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[DM97] D. Dasgupta und Z. Michalewicz, Hrsg. *Evolutionary Algorithms in Engineering Applications*. Springer, Berlin, 1997.

[Dra97] D. C. Dracopoulos. Genetic Algorithms and Genetic Programming for Control. In Dasgupta und Michalewicz [DM97], Seiten 329–343.

[ECS89] L. J. Eshelman, R. A. Caruna, und J. D. Schaffer. Biases in the crossover landscape. In J. D. Schaffer, Hrsg., *Proceedings of the Third International Conference on Genetic Algorithms*, Seiten 10–19. Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[EW88] Y. Ermoliev und J.-B. Wets, Hrsg. *Numerical Techniques for Stochastic Optimization*. Springer, Berlin, 1988.

[FF97] C. M. Fonseca und P. J. Fleming. C 4.5 Multiobjective Optimization. In Bäck et al. [BFM97], Seiten C4.5:1 – C4.5:9.

[FG88] J. M. Fitzpatrick und J. J. Grefenstette. Genetic Algorithms in Noisy Environments. *Machine Learning*, 3:101–120, 1988.

[Fog62] L. J. Fogel. Autonomous Automata. *Industrial Research*, 4:14–19, 1962.

[Fog91] D. B. Fogel. *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling*. Ginn Press, Needham Heights, 1991.

[Fog95] D. B. Fogel. *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, 1995.

[FOW66] L. J. Fogel, A. J. Owens, und M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, New York, 1966.

[GAP+96] E. D. Goodmann, R. C. Averill, W. F. Punch, Y. Ding, und B. Malott. Design of Special-Purpose Composite Material Plates via Genetic Algorithms. In Parmee [Par96], Seiten 3–9.

[GKD89] D.E. Goldberg, B. Korb, und K. Deb. Messy Genetic Algorithms: Motivation, Analysis and First Results. *Complex Systems*, 3(5):493–530, 1989.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, MA, 1989.

[Gre93] J. J. Grefenstette. Deception Considered Harmful. In L. D. Whitley, Hrsg., *Foundations of Genetic Algorithms 2*, Seiten 75–91. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[Ham81] C. Hampel. *Ein Vergleich von Optimierungsverfahren für die zeitdiskrete Simulation*. Dissertation, Technische Universität Berlin, 1981.

[Ham97] U. Hammel. F 1.8 Simulation Models. In Bäck et al. [BFM97], Seiten F1.8:1 – F1.8:9.

[Har97] G. R. Harik. *Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms.* Dissertation, University of Michigan, 1997.

[Hau98] M. Haupt. Evolutionsstrategische Prozeßoptimierung in der Kunststoffverarbeitung und der Umwelttechnik. In Verein Deutscher Ingenieure, Hrsg., *VDI Berichte 1381: Computational Intelligence – Neuronale Netze, Evolutionäre Algorithmen, Fuzzy Control im industriellen Einsatz.* VDI Verlag, Düsseldorf, 1998.

[HB94] U. Hammel und Th. Bäck. Evolution Strategies on Noisy Functions: How to Improve Convergence Properties. In Y. Davidor, H.-P. Schwefel, und R. Männer, Hrsg., *Parallel Problem Solving from Nature — PPSN III, International Conference on Evolutionary Computation*, Band 866 der Reihe *Lecture Notes in Computer Science*, Seiten 159–168. Springer, Berlin, 1994.

[HBH92] S. Hahn, K. H. Becks, und A. Hemker. Optimizing Monte Carlo Generator Parameters Using Genetic Algorithms. In D. Perret-Gallix, Hrsg., *New Computing Techniques in Physics Research II — Proceedings 2nd International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics*, Seiten 255–265, La Londe-Les-Maures, France, January 13–18 1992. World Scientific, Singapore, 1992.

[HH] M. Herdy und D. Holste. Optimieren mit subjektiven Gütekriterien.

[Hol62] J. H. Holland. Outline for a Logical Theory of Adaptive Systems. *Journal of the Association for Computing Machinery*, 3:297–314, 1962.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI, 1975.

[Iba97a] H. Iba. F 1.4 Identification. In Bäck et al. [BFM97], Seiten F1.4:1–F1.4:4.

[Iba97b] T. Ibaraki. D 3 Combinations with Other Optimization Methods. In Bäck et al. [BFM97], Seiten D3.1:1–D3.6.2.

[Kos95] B. Kost. Evolution Strategies in Structural Topology Optimization of Trusses. In P.J. Pahl und H. Werner, Hrsg., *Computing in Civil and Building Engineering*, Seiten 675–681. Balkema, Rotterdam, 1995.

[Koz94] John Koza. *Genetic Programming II.* MIT Press, Cambridge, MA, 1994.

[Kur95] F. Kursawe. Towards Self-Adapting Evolution Strategies. In *Proceedings of the Second IEEE Conference on Evolutionary Computation, Perth, Australia*, Seiten 283–288. IEEE Press, Piscataway, NJ, 1995.

[KW94] P. Kall und S.W. Wallace. *Stochastic Programming.* John Wiley & Sons, Chichster, 1994.

[LK91] A. M. Law und W. D. Kelton. *Simulation Modeling & Analysis.* McGraw-Hill, New York, 1991.

[McD97] J. R. McDonnel. F 1.3 Control. In Bäck et al. [BFM97], Seiten F1.3:1–F1.3:7.

[MG96] B. L. Miller und D. E. Goldberg. Genetic Algorithms, Selection Schemes, and the Varying Effects of Noise. *Evolutionary Computation*, 4(2):113–132, 1996.

[Mic96] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, Berlin, 1996.

[MLC97] W. N. Martin, J. Lienig, und J. P. Cohoon. C 6.3 Island (Migration) Models: Evolutionary Algorithms Based on Punctuated Equilibria. In Bäck et al. [BFM97], Seiten C6.3:1–C6.3:16.

[MM92] R. Männer und B. Manderick, Hrsg. *Parallel Problem Solving from Nature 2.* Elsevier, Amsterdam, 1992.

[MSV93a] H. Mühlenbein und D. Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm. *Evolutionary Computation*, 1(1):25–49, 1993.

[MSV93b] H. Mühlenbein und D. Schlierkamp-Voosen. The Science of Breeding and Its Application to the Breeder Genetic Algorithm (BGA). *Evolutionary Computation*, 1(4):335–360, 1993.

[Müh92] H. Mühlenbein. How Genetic Algorithms Really Work: I. Mutation and Hill-climbing. In Männer und Manderick [MM92], Seiten 15–25.

[NB95] V. Nissen und J. Biethahn. Determining a Good Inventory Policy with a Genetic Algorithm. [BN95], Seiten 240–249.

[Nis97] V. Nissen. F 1.2 Management Applications and other Classical Optimization Problems. In Bäck et al. [BFM97], Seiten F1.2:1–F1.2:50.

[Par96] I. C. Parmee, Hrsg. *Adaptive Computing in Engineering Design and Control-'96.* University of Plymouth, 1996.

[Pet97] C. C. Pettey. C 6.4 Diffusion (cellular) models. In Bäck et al. [BFM97], Seiten C6.4:1–C6.3:6.

[PP98] C. Poloni und V. Pediroda. GA Coupled with Computationally Expensive Simulations: Tools to Improve Efficiency. In Quagliarella et al. [QPPW98], Seiten 267–288.

[QPPW98] D. Quagliarella, J. Périaux, C. Poloni, und G. Winter, Hrsg. *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science.* Wiley, Chichester, 1998.

[Rao96] S. Rao, Hrsg. *Engineering Optimization — Theory and Practice.* Wiley & Sons, New York, 1996.

[Rec65] I. Rechenberg. Cybernetic Solution Path of an Experimental Problem. Royal Aircraft Establishment, Library translation No. 1122, Farnborough, Hants., UK, August 1965.

[Rec71] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Dissertation, Technical University of Berlin, Germany, 1971.

[Rec94] I. Rechenberg. *Evolutionsstrategie '94*, Band 1 der Reihe *Werkstatt Bionik und Evolutionstechnik*. Frommann–Holzboog, Stuttgart, 1994.

[Ron97] S. Ronald. Robust Encodings in Genetic Algorithms. In Dasgupta und Michalewicz [DM97], Seiten 29–44.

[Ros97] M. A. Rosenman. The Generation of Form Using an Evolutionary Algorithm. In Dasgupta und Michalewicz [DM97], Seiten 69–85.

[Rud91] G. Rudolph. Global Optimization by Means of Distributed Evolution Strategies. In H.-P. Schwefel und R. Männer, Hrsg., *Parallel Problem Solving from Nature — Proceedings 1st Workshop PPSN I*, Band 496 der Reihe *Lecture Notes in Computer Science*, Seiten 209–213. Springer, Berlin, 1991.

[Rud92] G. Rudolph. On Correlated Mutations in Evolution Strategies. In Männer und Manderick [MM92], Seiten 105–114.

[Rud94a] G. Rudolph. Convergence of Non-Elitist Strategies. In *Proceedings of the First IEEE Conference on Evolutionary Computation, Orlando, FL*, Seiten 63–66. IEEE Press, Piscataway, NJ, 1994.

[Rud94b] G. Rudolph. An Evolutionary Algorithm for Integer Programming. In Y. Davidor, H.-P. Schwefel, und R. Männer, Hrsg., *Parallel Problem Solving from Nature — PPSN III International Conference on Evolutionary Computation*, Band 866 der Reihe *Lecture Notes in Computer Science*, Seiten 139–148. Springer, Berlin, 1994.

[Rud96] G. Rudolph. Convergence of Evolutionary Algorithms in General Search Spaces. In *Proceedings of the Third IEEE Conference on Evolutionary Computation*, Seiten 50–54. IEEE Press, Piscataway, NJ, 1996.

[Sch68] H.-P. Schwefel. Projekt MHD-Staustrahlrohr: Experimentelle Optimierung einer Zweiphasendüse, Teil I. Technischer Bericht 11.034/68, 35, AEG Forschungsinstitut, Berlin, Oktober 1968.

[Sch75] H.-P. Schwefel. *Evolutionsstrategie und numerische Optimierung.* Dissertation, Technical University of Berlin, Germany, 1975.

[Sch92] H.-P. Schwefel. Imitating Evolution: Collective, Two-Level Learning Processes. In U. Witt, Hrsg., *Explaining Process and Change — Approaches to Evolutionary Economics*, Seiten 49–63. The University of Michigan Press, Ann Arbor, MI, 1992.

[Sch94] J. D. Schaffer. Combinations of Genetic Algorithms with Neural Networks or Fuzzy Systems. In Zurada et al. [ZMR94], Seiten 371–395.

[Sch95] H.-P. Schwefel. *Evolution and Optimum Seeking.* Sixth-Generation Computer Technology Series. Wiley, New York, 1995.

[Sch97] H.-P. Schwefel. Evolutionary Computation — A Study on Collective Learning. In N. Callaos, C. M. Khoong, und E. Cohen, Hrsg., *Proc. World Multiconference on Systemics, Cybernetics and Informatics (SCI '97)*, Band 2, Seiten 198–205, Caracas, Venezuela, 7.–11. Juli 1997. Int'l Inst. of Informatics and Systemics (IIIS), Orlando FL.

[SHMM96] M. Schütz, U. Hammel, A. Meyer, und P. Maldaner. Gesamtoptimierung verfahrenstechnischer Anlagen mit naturanalogen Methoden, Teil B: Der Einsatz evolutionärer Algorithmen. In *Kolloquium der Volkswagen-Stiftung zum Schwerpunkt „Modellierung komplexer Systeme der Verfahrenstechnik", 22.-23. Feb., RWTH Aachen*, 1996.

[SJ97] J. Sarma und K. De Jong. C 2.7 Generation Gap Methods. In Bäck et al. [BFM97], Seiten C2.7.1–C2.7.5.

[SM87] J. D. Schaffer und A. Morishima. An Adaptive Crossover Distribution Mechanism for Genetic Algorithms. In J. J. Grefenstette, Hrsg., *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, Seiten 36–40. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[Spr94] J. Sprave. Linear Neighborhood Evolution Strategy. In A. V. Sebald und L. J. Fogel, Hrsg., *Proceedings of the Third Annual Conference on Evolutionary Programming*, Seiten 42–51. World Scientific, Singapore, 1994.

[Suz95] J. Suzuki. A Markov Chain Analysis on Simple Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(4):655–659, April 1995.

[Vos95] M. D. Vose. Modeling Simple Genetic Algorithms. *Evolutionary Computation*, 3(4):453–472, 1995.

[WHB98] D. Wiesmann, U. Hammel, und Th. Bäck. Robust Design by Evolution Strategies. In *Proceedings of the Fifth IEEE Conference on Evolutionary Computation.* IEEE Press, Piscataway, NJ, 1998.

[Whi95] L. D. Whitley. An Executable Model of a Simple Genetic Algorithm. In M. D. Vose L. D. Whitley, Hrsg., *Foundations of Genetic Algorithms 3*, Seiten 45–62. Morgan Kaufmann Publishers, San Francisco, CA, 1995.

[Whi97] D. Whitley. C 1.4 Permutations. In Bäck et al. [BFM97], Seiten C1.4:1 – C1.4:8.

[WM97] D. H. Wolpert und W. G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[ZMR94] J. M. Zurada, R. J. Marks, und C. J. Robinson, Hrsg. *Computational Intelligence: Imitating Life*. IEEE Press, New York, 1994.