

Einführung in die  
Angewandte Optimierung

# **Genetische Algorithmen**

Alexander Schatten

29. November 2004

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
<b>2</b>	<b>Optimierung?</b>	<b>4</b>
2.1	Einleitung – Beispiele . . . . .	4
2.2	Definitionen . . . . .	5
2.3	Topographische Veranschaulichung des zwei-dimensionalen Falles . . . . .	6
2.4	Veranschaulichung des Optimierungsproblem als Blackbox . . . . .	7
2.5	Größe des Suchraumes . . . . .	7
<b>3</b>	<b>Biologische Prinzipien</b>	<b>9</b>
3.1	Einleitung . . . . .	9
3.2	Mechanismen . . . . .	11
3.3	Selektion . . . . .	11
3.4	Paarung, Crossover . . . . .	11
3.5	Mutation . . . . .	11
<b>4</b>	<b>Hill Climbing, Monte-Carlo: Der Weg zu Evolutionären Methoden</b>	<b>12</b>
4.1	Einleitung . . . . .	12
4.2	Hill Climbing . . . . .	12
4.3	Monte-Carlo Zufalls-Suche . . . . .	15
4.4	Analogien zu den biologischen Prinzipien — Der Weg zu Genetischen Algorithmen . . . . .	15
<b>5</b>	<b>Implementation</b>	<b>15</b>
5.1	Codierung . . . . .	15
5.1.1	NUM – Parameter Estimation Probleme . . . . .	16
5.1.2	SUB – Subset Selection . . . . .	18
5.1.3	SEQ – Sequencing . . . . .	19
5.2	Algorithmus . . . . .	19
5.2.1	Initialpopulation . . . . .	19
5.2.2	„Bewertung“ der Population – Selektion . . . . .	19
5.2.3	Paarung (Mating) . . . . .	21
5.2.4	Crossover . . . . .	21
5.2.5	Mutation . . . . .	23
5.2.6	Abbruchkriterium . . . . .	23
5.3	Bewertung . . . . .	23
5.4	Kritische Anmerkungen . . . . .	24
5.5	Beispiel . . . . .	24
<b>6</b>	<b>Evolutionäre Algorithmen</b>	<b>27</b>
<b>7</b>	<b>Anmerkungen</b>	<b>28</b>

# 1 Einleitung

*As I wrote many years ago at the very beginning of the debate about computers, a computer is just a glorified pencil. Einstein once said “my pencil is cleverer than I”. What he meant could perhaps be put thus: armed with a pencil, we can be more than twice as clever as we are without. Armed with a computer (a typical World 3 object), we can perhaps be more than a hundred times as clever as we are without; and with improving computers there need not be an upper limit to this.*

*Karl Popper [17]*

Bei Genetischen Algorithmen (GAs) handelt es sich um eine Klasse von Algorithmen, die sich (wie im nächsten Abschnitt verdeutlicht werden soll) Optimierungsstrategien von biologischen Systemen „abschauen“, diese abstrahieren und auf die jeweilige Problemstellung anpassen. So ergeben sich neue Ansätze für viele konventionell schwierig oder gar nicht handhabbare Optimierungsprobleme (z.B. nicht lineare Modelle), die so in akzeptabler Zeit lösbar werden (z.B. [11, 9]).

Die Leistungsfähigkeit der GAs läßt sich durch folgende Überlegung verstehen: Jede parametrisierbare Optimierungsaufgabe, bei der jedem Parametersatz ein Wert zugewiesen werden kann, der für die Qualität der Optimierung<sup>1</sup> steht ist (prinzipiell) mittels GA lösbar (siehe auch Abb.3).

Im Prinzip deshalb, weil bei aufwendigen Optimierungs-Problemen selbst so leistungsfähige Algorithmen wie GAs an ihre Grenzen stoßen können. Die Rechenzeiten können bei vielen Ausgangsdaten, vielen Parametern und/oder komplizierten Modellen Tage dauern! Zunächst soll ein kurzer Überblick über den Inhalt dieses Skriptums vermittelt werden:

Im nächsten Abschnitt wird kurz zusammengefaßt, was unter dem Terminus *Optimierung* zu verstehen ist, und auch der für die Veranschaulichung wichtige Vergleich mit topographischen Karten hergestellt.

Dann wird ein kurzer Abstecher in die Biologie, genauer gesagt in die Molekulargenetik notwendig sein, da man sich bei der Programmierung derartiger evolutionärer Algorithmen<sup>2</sup> einer Vorgangsweise bedient, die sich auch andere Bereiche z.B. die Bionik<sup>3</sup> angeeignet hat. Man versucht also von der Vorgehensweise der Natur zu lernen, und deren Strategie den eigenen Problemen anzupassen. Darum ist es für das Verständnis wichtig, sich die biologischen Vorgänge zumindest rudimentär zu verinnerlichen.

Der zweite Ansatz, aus denen man sich die Entwicklung genetischer Algorithmen herleiten

---

<sup>1</sup>Unter Bewertung versteht man beispielsweise die Summe der quadrierten Residuen, mit deren Hilfe es möglich ist, jedem beliebigen Parametersatz eines Modells eine Wertung, also eine Qualität zuzuweisen (siehe auch Abb. 1).

<sup>2</sup> Ich stecke hier in einem gewissen „Dilemma“ mit der Nomenklatur dieser Art von Algorithmen. An sich sollte strikt zwischen *genetischen Algorithmen* und *evolutionären Algorithmen*, wie in Abschnitt 6 erwähnt, unterschieden werden. Trotzdem hat sich der Begriff evolutionäre Algorithmen aus naheliegenden Gründen für diese ganze Klasse eingebürgert.

<sup>3</sup> *Bionik* ist ein Kunstwort, das sich aus den Begriffen *Biologie* und *Technik* zusammensetzt. Bionik ist eine knapp 30 Jahre alte Wissenschaft, die sich zur Aufgabe gemacht hat Technologie und Technik aus Bauprinzipien der Natur abzuleiten, also Technik nach dem Vorbild der Natur.

kann ist aus der Kombination konventioneller Optimierungs-Strategien. Diese Idee wird (auch im Vergleich zum „biologischen“ Ansatz) im nächsten Abschnitt beschrieben.

In den weiteren Abschnitten wird kurz die Abstraktion — also der verwendete Algorithmus — und die Software-Implementation dargestellt. Außerdem werden die wichtigsten Anwendungen erwähnt und kurz beschrieben.

Schließlich wird anhand eines einfachen Beispiels ein „konventioneller“ Ansatz der Lösung mithilfe eines genetischen Algorithmus gegenübergestellt.

## 2 Optimierung?

### 2.1 Einleitung – Beispiele

Optimierungsprobleme treten in nahezu allen Bereichen der Naturwissenschaften, Technologie und Logistik auf. Allerdings ist die Bandbreite der Probleme sehr groß, und die Anzahl der mittlerweile existierenden Verfahren bzw. Algorithmen nahezu unüberschaubar. Die folgenden Beispiele sollen einen Eindruck der Problemvielfalt geben, die man unter dem Überbegriff *Optimierung* zusammenfaßt:

1. Ein Epidemiologe versucht einen Zusammenhang zwischen der Konzentration von Benzol in der Luft und dem Auftreten von Krebs zu finden. Das Problem, das sich stellt, ist eine möglichst gute (optimale) Anpassung eines exponentiellen Modells an die Daten zu finden.
2. Eine Firma, die Leiterplatten produziert steht vor der Aufgabe, die Qualitätssicherung der Produktion zu optimieren. Gegensätzliche Ziele in diesem Prozeß sind: Die Verbesserung der Qualitätskontrollen und die Kosten, da eine Verbesserung der Qualitätskontrolle mit einer exponentiellen Zunahme der Kosten verbunden sei. Andererseits sind die Folgekosten im Falle von in Umlauf geratenen defekten Produkten mit ins Kalkül zu ziehen und ein Parameter, der beachtet werden muß. (Nicht nur die reinen Kosten an Reklamationen und Schadenersatz-Zahlungen sollten berücksichtigt werden, sondern ebenfalls der mögliche Vertrauensverlust im Falle zu hoher Fehleraten. Dies ist zweifellos eine Größe, die nicht leicht zu quantifizieren sein wird.)
3. Ein Mathematiker erhält den Auftrag den Busfahrplan eines neuen Unternehmers zu gestalten und mit den Fahrzeiten von Schnellbahn und Straßenbahn der Stadt abzustimmen.
4. Ein analytischer Chemiker versucht die Auflösung von NMR (Kernresonanz) Spektren zu verbessern. Es stellt sich die Aufgabe, einen Satz von Geräteparametern so einzustellen, daß maximale Auflösung des Meßgerätes erreicht wird.
5. Ein Wissenschaftler in der pharmazeutischen Industrie ist im Besitz eines großen medizinischen Datensatzes einer Gruppe von Patienten, die mit einem bestimmten neuen Medikament behandelt worden sind. Dieser Datenbestand besteht aus etwa 20 Variablen wie Blutdruck, Puls, Blutwerte, usw. Er möchte aus diesen 20 Variablen nur die

für die Untersuchung signifikanten (er schätzt etwa fünf) herausfinden. Es ist eine Methode zu entwickeln, diese *Parameterselktion* durchzuführen.

## 2.2 Definitionen

Zusammenfassend kann **Optimierung** definiert werden als *den Versuch Eigenschaften eines Systemes in geschickter Weise so zu verändern, daß eine gestellte Aufgabe besser oder optimal bewältigt oder sogar gelöst werden kann.*

Prinzipiell lassen sich Optimierungsprobleme in drei Klassen einteilen [14, 15]:

**Parameteroptimierung:** Probleme bei denen bspw. *Parameter* eines mathematischen Modells zu optimieren sind, vgl. Beispiel 1.

**Subset-Selection:** Probleme, bei denen aus einer *Menge* eine *Untermenge* ausgewählt werden muß, vgl. Beispiel 5.

**Kombinatorische Probleme:** Probleme, bei denen die *Anordnung* von Elementen eine Rolle spielt. Ein gängiges Beispiel ist das sogenannte *traveling salesman* Problem: Es soll die optimale Anordnung von Städten auf einer Reiseroute bestimmt werden. Als „optimal“ könnte bspw. die kürzeste Strecke definiert werden, oder auch die Strecke, die die geringsten Kosten verursacht, etc., vgl. auch Beispiel 3. Probleme dieser Kategorie werden auch mit dem Begriff **Sequencing (SEQ)** bezeichnet (siehe auch 5.1.3 auf Seite 19)

In dieser Zusammenfassung wird in erster Linie die Parameteroptimierung besprochen, da es sich dabei neben der Subset-Selection um die wichtigste Klasse für den Chemiker handelt. In der Beschreibung der Implementation der GAs wird dann auch kurz auf die anderen Klassen eingegangen werden.

Unter einem **Parameter** (= Variable) wird eine *veränderliche Größe* des betrachteten Systems verstanden, dessen Veränderung in „irgendeiner“ Weise mit dem Ergebnis einer *Qualitätsfunktion* in Zusammenhang steht.

Eine **Qualitätsfunktion** ist ein Kriterium, das die qualitative Beurteilung des betrachteten Systemes in Abhängigkeit von den Parametern erlaubt.

Unter einem **Suchraum** oder auch **Phasenraum**<sup>4</sup> wird der Raum verstanden, den die Parameter aufspannen (zur Veranschaulichung siehe auch Abschnitt 2.3).

Im ersten Beispiel der Einleitung (Abschnitt 2.1) wären die *Parameter* des möglichen mathematischen Modelles

$$f(x) = a \cdot e^{bx} \quad (1)$$

---

<sup>4</sup> Im physikalischen Sinne wird der Terminus *Phasenraum* für den Satz von Parametern verwendet, der ausreicht um ein bestimmtes physikalisches System vollständig zu beschreiben. Bspw. wird der Phasenraum eines mathematischen Pendels durch die Parameter *Auslenkung des Pendels* und *Winkelgeschwindigkeit* bestimmt. Der Zustand des Systems zu einem bestimmten Zeitpunkt ist dann als *Punkt* im Phasenraum zu verstehen. Mit der zeitlichen Entwicklung komplexer Systeme im Phasenraum beschäftigt sich die relative junge Wissenschaft der Chaos-Forschung. Eine gute Einführung findet man in [1].

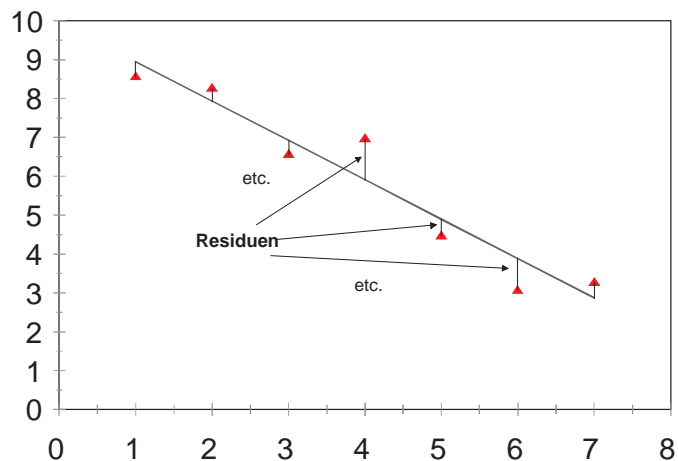


Abbildung 1: Residuen einer linearen Regression (Ausgleichsgerade): Die *Residuen* sind die *Abweichungen* der einzelnen Meßwerte von der Geraden. Die Gerade wird im Fall der linearen *least sum of squared residuals* Regression (= Ausgleichsgerade) als optimal verstanden, wenn die Summe der quadrierten Abweichungen (= Residuen) von der Geraden minimal wird.

$a$  und  $b$ . Die *Qualitätsfunktion* wäre in diesem Fall so zu definieren, daß sie ein Maß für die Qualität der Anpassung des mathematischen Modells in Gleichung 1 für beliebige Werte von  $a$  und  $b$  darstellt. Ein oft verwendetes Maß ist die Summe der quadrierten Residuen, d.h.: die Summe der Abweichungen vom Modell zu Originaldaten zum Quadrat. Diese Funktion liefert *sehr große* Werte im Falle *großer* Abweichungen — also schlechter Anpassung, und *sehr kleine* Werte im Falle *geringer* Abweichung — also guter Anpassung (siehe auch Abb.1).

Der *Suchraum* dieses Systems wäre die „Landschaft“ die die beiden Parameter  $a$  und  $b$  aufspannen (siehe auch Abschnitt 2.3).

### 2.3 Topographische Veranschaulichung des zwei-dimensionalen Falles

Den *Suchraum* kann man sich (zumindest im zwei-dimensionalen Fall) auch als topographische Landschaft vorstellen. Die beiden Parameter werden durch die x- und y-Achse repräsentiert (siehe Abb. 2). Für jeden Parametersatz, d.h. für jedes Wertepaar der beiden Parameter ist das Ergebnis der Qualitätsfunktion als Höhe dargestellt. Man kann sich also vorstellen, daß Gipfel in dieser Landschaft für gute Werte der entsprechenden Koordinaten stehen, und Täler dementsprechend für schlechte. (Oder auch umgekehrt — das ist natürlich Problemabhängig.)

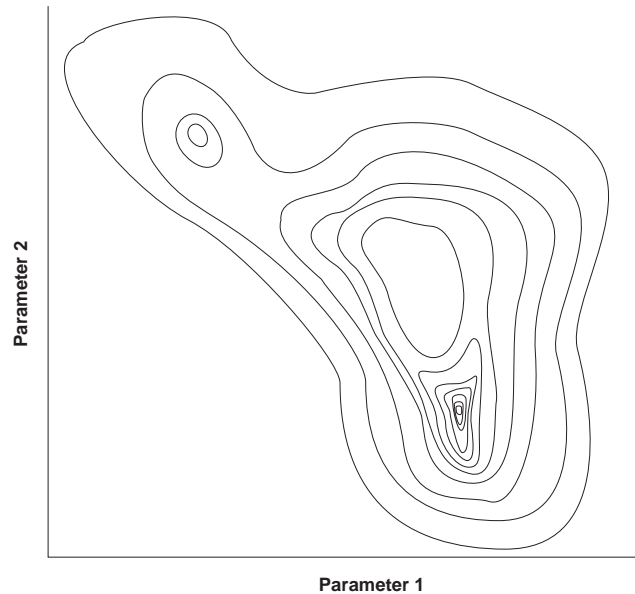


Abbildung 2: Topographische Veranschaulichung eines zwei-dimensionalen Suchraumes.

## 2.4 Veranschaulichung des Optimierungsproblem es als Blackbox

Eine andere Möglichkeit der Veranschaulichung in erster Linie der Parameter-Optimierung ist in Abb. 3 illustriert. Man stelle sich ein Gerät vor, bei dem eine Anzahl von Knöpfen (im Beispiel 3) vorhanden sind, die in einer bestimmten — aber evt. auch unbekannt — Art und Weise auf ein Meßgerät wirken. Die Optimierungsaufgabe wird nun als *möglichst günstige* Strategie verstanden an diesen Knöpfen so zu drehen, daß sich ein gewünschtes Ergebnis am Meßgerät einstellt. Das gewünschte Ergebniss könnte z.B. ein maximaler oder ein minimaler Wert sein.

Die Stellung der Knöpfe entspricht den Parametern, die Anzeige des Meßgerätes dem Ergebnis der Qualitäts- (= Fitness-) Funktion. Das interessante an diesem Vergleich ist, daß der Zusammenhang keinen Einfluß auf die Konstruktion des GA hat. Dieser Zusammenhang wird vielmehr problemabhängig von der Fitnessfunktion bestimmt.

## 2.5 Größe des Suchraumes

Ein essentielles Problem, das mit der Auswahl geeigneter Optimierungsstrategien in engem Zusammenhang steht, ist die *Größe des Suchraumes*. Zur Veranschaulichung der Problematik wird die Größe anhand eines einfachen Beispiels gezeigt:

Angenommen, es soll ein System mit zwei Parametern optimiert werden, wobei (der Einfachheit halber) beide Parameter ( $a_1, a_2$ ) im Bereich:

$$100 \geq a_n \geq 0.1, \quad \text{Schrittweite } 0.1 \quad (2)$$

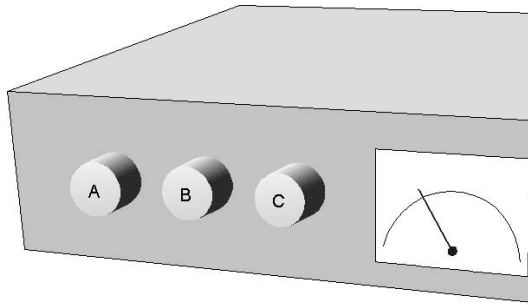


Abbildung 3: Das Optimierungsproblem kann als „Blackbox“ veranschaulicht werden. Die Parameter des zu optimierenden Systemes, sind als Drehknöpfe dargestellt. Der „Responsewert“, der maximiert oder auch minimiert werden soll ist durch das Meßinstrument charakterisiert. Die Aufgabe die sich im Zuge der Optimierung also stellt, ist die Knöpfe an der Front des Gerätes in eine Position zu bringen, sodaß das Instrument einen optimalen Wert annimmt.

zu variieren sind. D.h. für jeden Parameter gibt es in diesem Genauigkeitsbereich 1000 mögliche Werte die er annehmen kann. Der Suchraum, der durch die beiden Parameter aufgespannt wird, umfaßt somit  $1000 * 1000 = 1000000$  Möglichkeiten. Sollte noch ein dritter Parameter mit denselben Eigenschaften dazukommen, so würde die Anzahl der Möglichkeiten nicht etwa *doppelt* oder *dreimal* so groß, sondern vielmehr *1000* mal so groß. In Abb.4 ist qualitativ (!) dargestellt, wie sich die Größe des Suchraumes im Verhältnis zur Anzahl der Parameter (Dimension) verhält.

Ausgehend von diesen Überlegungen erscheint es einsichtig, daß ein „Durchprobieren“ aller möglichen Werte<sup>5</sup> nur in sehr niedrig-dimensionalen Räumen möglich ist. Sobald die Anzahl der Dimensionen, oder die erforderliche Präzision größer wird, scheitern die Methoden aufgrund der ins astronomische wachsende Rechenzeiten. Wolfgang Auer hat sich die Mühe gemacht ein interessantes Beispiel zu errechnen, seine Ausführungen daher als Zitat:

*Gegeben sei eine 100-Tasten-Tastatur (Standard-PC hat 102) und ein Affe. Der Affe drücke willkürlich in die Tasten, daher sind die Ereignisse „Taste drücken“ alle paarweise voneinander unabhängig<sup>6</sup>. Wenn Du nun ein Wort vorgibst, z.B. „toast“, so muß für jeden Buchstaben unabhängig in der richtigen Reihenfolge die richtige Taste gedrückt werden, das heißt, jeder Buchstabe wird mit der Wahrscheinlichkeit 1/100 „erraten“.*

<sup>5</sup> Solche Algorithmen werden auch *brute force* — also „rohe Gewalt“ — Algorithmen genannt. Der Grund ist, daß eben keinerlei „Intelligenz“ zur Lösung einer Problemstellung verwandt wird, sondern eben vielmehr alle erdenklichen Varianten ausprobiert werden. Der Vorteil dieses Ansatzes ist allerdings, daß man keine Möglichkeit übersehen kann. Somit werden auch in theoretischen Arbeiten oftmals brute-force Methoden als „Referenz“ verwendet.

<sup>6</sup> Unter *paarweiser Unabhängigkeit* versteht man, daß die Wahrscheinlichkeit, daß ein Ereignis eintritt *nicht* von vorhergegangenen Ereignissen abhängig ist.

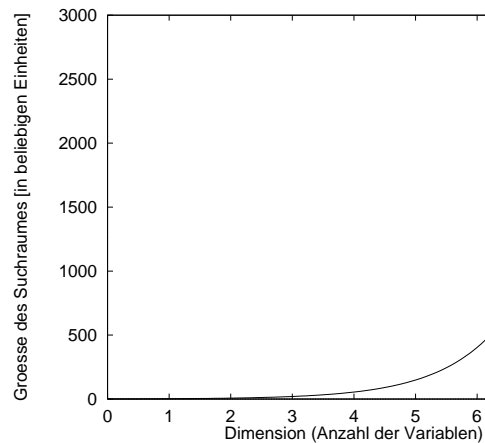


Abbildung 4: Zunahme der Größe des Suchraumes, mit Anzahl der Parameter (Dimension).

*Fazit: Schon bei 5 Buchstaben haben wir Wahrscheinlichkeit  $1/10^{10}$ ; soll vorne und hinten ein Leerzeichen stehen, ist die Wahrscheinlichkeit schon  $1/10^{14}$  (!) Was das bedeutet, kann man sich leicht ausrechnen. Drückt der Affe eine Tasten pro Sekunde (gar nicht einmal so unwahrscheinlich, wenn man ihm dafür eine Banane verspricht), so ist die wahrscheinlich benötigte Zeit  $1 \cdot 10^{14}$  Sekunden. 86400 Sekunden sind ein Tag, nicht ganz 365.25 Tage ein Jahr, das heißt (grübel), wir haben  $3.17 \cdot 10^7$  Jahre Zeit. Eine ganze Menge, der Affe wird wohl seine Banane nie in Empfang nehmen können (wenn er, sagen wir, 30 wird, ist die Wahrscheinlichkeit dafür  $30/(3.17 \cdot 10^7)$  entspricht fast genau 1:100000)*

Bei einfachen Problemstellungen kann also die Brute-Force Methode durchaus eine probate Strategie sein, zumal die Garantie besteht, die optimale Lösung zu finden, bei größeren Systemen jedoch stößt dieser Ansatz sehr schnell an die Grenzen des berechenbaren, und andere Verfahren müssen angewandt werden.

### 3 Biologische Prinzipien

#### 3.1 Einleitung

„Die Bedürfnisse der Individuen einer Art sind mitverantwortlich, daß sie bestimmte Eigen-

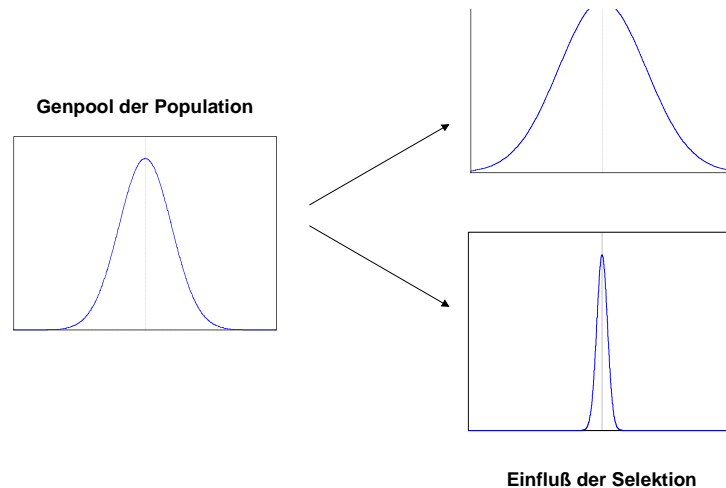


Abbildung 5: Die Mutation wirkt quasi als Gegenpol zur Selektion. Während die **Selektion** die „Breite“ des Genpools einer Art einschränkt, wird sie durch den „ziello- sen“ Charakter der **Mutation** verbreitert.

*schaften entwickeln*“ — Lamarck vermutet bereits im Jahr 1801 [13], also noch deutlich vor Charles Darwin, daß, wenn Bedürfnisse einer Art über eine bestimmte Zeit konstant bleiben, entsprechende Eigenschaften vererbt werden. Mit anderen Worten: wenn sich die Lebensumstände einer Spezies ändern, z.B. wenn über längere Zeit Trockenheit vorherrscht, und Pflanzen nur mehr in größerer Höhe vorhanden sind, so würde sich diese „Information“ im Erbmaterial niederschlagen und entsprechend Tiere mit langem Hals „erzeugen“. Lamarck lehnte den *Zufall* als Motor der Evolution ab. So ist aus heutiger Sicht die Interpretation der „Lamarckisten“ problematisch. Sie versuchten einen Informationsfluß vom *Phän* zum *Gen* zu beweisen, d.h. von den biologischen Strukturen zu deren genetischer Codierung. Allerdings dürfte diese Problematik erst in diesem Jahrhundert den beiden „Antipoden“ *Lamarck* und *Charles Darwin* in den Mund gelegt worden sein.

In der Tat kann auch Darwin als Lamarckist bezeichnet werden [20], zumindest in dem Sinne, daß er die Entstehung der Artenvielfalt auf eine Entwicklung, eine Evolution — geprägt durch Selektion — postulierte [4, 3] und nicht eine a priori statische Existenz der Arten annahm (seit der Schöpfung ...). Auch widersprachen beide (wie auch Lyell) den damals auch modernen Katastrophentheorien [21]. (Die Konflikte und Polarisierungen scheinen später durch das Aufkommen der Genetik geprägt worden zu sein, im besonderen durch das Zentraldogma (oder auch Zentralschema) der Molekulargenetik [10], das eben diesen Informationsfluß vom Phän zum Gen untersagt. Allerdings ist ein Dogma als Basis einer Naturwissenschaft meiner Meinung nach als einigermaßen problematisch anzusehen und wurde auch vielfach kritisiert [19].)

## 3.2 Mechanismen

Was sind nun die (hier stark vereinfacht wiedergegebenen) Prinzipien biologischer Evolution auf der Basis des Darwinismus und der weiteren Entdeckungen in der Physik [22] und der Molekulargenetik [8]:

Die Codierung der Erbinformation durch ein Makromolekül [24] in der Hierarchie: Gen, Chromosom, und den Prinzipien **Selektion**, **Mutation**, **Paarung** und damit im Zusammenhang stehend: **Crossover**<sup>7</sup>. Auf dieser Grundlage ist nach der Ansicht der meisten führenden Experten auf diesem Gebiet die Entwicklung der Arten bis zum Menschen hin, auch unter Verzicht auf den Rückgriff auf *vitalistische* oder — allgemeiner — auf *teleologische* Konzepte, erklärbar (siehe auch [8, 16]), wobei letztere immer noch heftig diskutiert werden.

## 3.3 Selektion

Unter *Selektion* versteht man die Bewertung der Fähigkeiten eines Individuums in Wechselwirkung mit seiner Umwelt(!)<sup>8</sup> und der daraus resultierenden Fortpflanzung der am besten angepaßten Individuen, womit auch der nächste Begriff, die **Paarung** erklärt ist. Mit anderen Worten, diejenigen Individuen, die durch ihre Eigenschaften über eine bessere Überlebensfähigkeit in der jeweiligen Umwelt verfügen, werden sich stärker vermehren und somit auch ihre genetische Information.

## 3.4 Paarung, Crossover

Paarung und Fortpflanzung laufen immer unter Informationsaustausch zwischen den Chromosomen/Genen der Eltern-Individuen ab. Dieser Informationsaustausch findet in dem genetischen Mechanismus der *Crossover* genannt wird statt, wobei der „Informationsträger“ in der Natur Makromoleküle, nämlich die *Desoxyribonucleinsäure* (DNS, DNA) sind.

In der weiteren Folge sollten wieder unter dem Selektionsdruck die fähigsten Individuen überleben.

## 3.5 Mutation

Nun ist noch eine Frage offen: wie können neue Strukturen entstehen? Eigenschaften also, die *noch nicht* in der Erbsubstanz der Art zu finden sind! Aufgrund der Eigenschaften der Codierung der Erbinformation in einem Makromolekül, der DNS<sup>9</sup>, kann es zu Fehlern beim Kopieren, durch Einwirkung von Strahlung, chemischen Substanzen, etc. kommen. Diese

---

<sup>7</sup> in der Biologie wird der Terminus *crossing over* bevorzugt

<sup>8</sup> „Survival of the fittest becomes a tautology, with no objective definition of fitness“, Farmer (Santa Fe)

<sup>9</sup> Die Abkürzung DNS steht für *Desoxyribonucleinsäure*: Auf einer Zucker-Phosphat Kette wird die Erbinformation mit von vier Basen codiert: Guanin, Adenin, Cytosin und Thymin. Bereits bei Bakterien hat sie eine Länge von etwa zweihunderttausend, beim Menschen zwei Milliarden „Zeichen“! Aufgrund der Tatsache, daß es sich bei der Abschrift um biochemische Vorgänge handelt sind Kopierfehler natürlich nicht ausgeschlossen und treten mit einer gewissen Häufigkeit auf [20].

Fehler werden *Mutationen* genannt<sup>10</sup>. Die meisten Mutationen werden von der Selektion wieder ausgemerzt, weil sie entweder keine entscheidende Verbesserung gebracht haben, oder sogar lebensunfähig sind. Einige mutierte Individuen können jedoch überleben und für neue (bisher unter Umständen noch nicht dagewesene) Eigenschaften der Art sorgen. Der Unterschiedliche Charakter von Mutation und Selektion auf den Genpool einer Spezies wird in Abb.5 illustriert.

Auf eine detailliertere Diskussion dieser Prozesse muß — um den Rahmen nicht zu sprengen — verzichtet werden, i.B. auch aufgrund der Tatsache, daß die Konsequenzen dieser Ansätze heiß umstritten sind, und guten Gewissens hier keine einheitliche Meinung wiedergegeben werden könnte. Es sei an dieser Stelle auf die oben zitierten Literaturstellen verwiesen.

## 4 Hill Climbing, Monte-Carlo: Der Weg zu Evolutionären Methoden

### 4.1 Einleitung

Der zweite Gesichtspunkt unter dem man die Entwicklung bzw. die Bedeutung genetischer Algorithmen betrachten kann, ist als Versuch „klassische“ Optimierungsverfahren zu kombinieren und damit die Nachteile verschiedener Methoden auszugleichen. Während „brute force“ Methoden *jede* Möglichkeit eines Systems durchrechnen oder dies zumindest mit einer gewissen Genauigkeit versuchen uns somit — wie in den vorigen Abschnitten dargelegt, nur in wenigen (einfachen) Fällen praktikabel sind — stehen ansonsten auf der einen Seite *streng deterministische Methoden*, die nach klaren Konzepten lokale Maxima suchen. Deren Schwäche ist aber in großen oder komplexen Suchräumen offensichtlich, da nicht immer klar ist, in welchem Bereich auch nur *ungefähr* ein Optimum zu finden ist.

Auf der anderen Seite sind Verfahren, die auf *reinem Zufall* basieren zwar in der Suche eher umfassend, ihnen mangelt es aber klarerweise an der „Hartnäckigkeit“, eine lokalisierte, gute Region „auszubeuten“, also den Gipfel zu ermitteln.

Während deterministische Verfahren sich in die Suche des *nächstliegenden* Gipfels „verbeißen“, leiden diese quasi unter dem Problem am halben Weg stehen zu bleiben. Die Idee diese beiden Strategien zu verbinden und synergistische Effekte auszunutzen liegt also auf der Hand.

So werden zunächst kurz die Prinzipien der sogenannten „Hill Climbing“ Methoden beschrieben, und auf die Idee der Monte-Carlo Zufalls-Suche eingegangen.

### 4.2 Hill Climbing

Eine sofort einsichtige Strategie ein — zumindest lokales — Optimum in einem *Suchraum* (siehe auch Abschnitt 2 auf Seite 4) zu finden, ist, sich von einem beliebig gewählten Punkt

---

<sup>10</sup> Es ist wichtig sich an dieser Stelle klar zu machen, daß Zufall nicht gleichzusetzen ist mit Willkür. Es ist möglich, wie es hier auch anhand der GAs demonstriert wird, Zufall sehr systematisch und effizient einzusetzen.

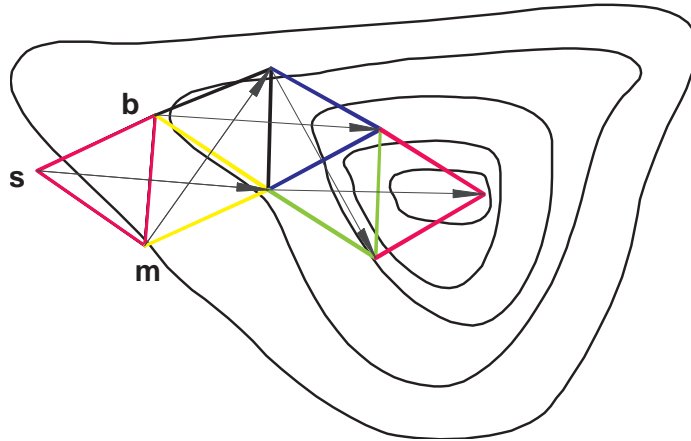


Abbildung 6: SIMPLEX Optimierung: Für das erste SIMPLEX ist die „Ordnung“ der Eckpunkte angegeben, wobei  $s$  für den schlechtesten,  $m$  für den mittleren und  $b$  für den besten Punkt steht. Der Pfeil symbolisiert die Richtung der „Spiegelung“. Die weiteren Simplexes werden analog behandelt.

immer in *Richtung der größten Steigung* zu orientieren. So ist garantiert, zumindest das nächstliegende *lokale* Optimum zu finden. Nun stellt sich die Frage, wie man diese Strategie möglichst effizient umsetzt.

Der SIMPLEX Algorithmus [6, 5] verwendet zu diesem Zweck die kleinste geschlossene geometrische Figur, die ausreicht um die Steigung in einem Gebiet zu ermitteln, eben den sogenannten SIMPLEX: Im Falle einer zwei-dimensionalen Suchlandschaft (= zwei Parameter) ist dies ein Dreieck, im drei-dimensionalen Fall eine Pyramide . . .

Wird nun bspw. ein Dreieck in eine zwei-dimensionale Landschaft „gelegt“ so kann man anhand der „Höhe“ der drei Eckpunkte des Dreiecks leicht die annähernde Richtung des nächsten Gipfels abschätzen.

Die Suche nach dem „Gipfel“ also dem nächstliegenden Maximum oder Minimum erfolgt prinzipiell nach dem folgenden Schema<sup>11</sup>:

1. Ein initiales SIMPLEX (also im zwei-dimensionalen Fall ein Dreieck) wird an zufälliger Position in den Suchraum „gelegt“.
2. Die „Qualität“ der Eckpunkte (topographisch betrachtet — die Höhe) wird berechnet, gemessen oder auf irgend eine andere (problemabhängige) Art ermittelt.

<sup>11</sup> In diesem Schema wird die „einfachste“ Implementation beschrieben. Ein offensichtliches Problem in dieser einfachen Implementation ist die *Größe* des SIMPLEX. Wird das initiale SIMPLEX *zu klein* angesetzt, sind u.U. sehr viele Punkte notwendig um das Optimum zu finden. Wird es aber *zu groß* angenommen, ist zwar die Annäherung an das Optimum sehr schnell, dafür besteht die Gefahr das Optimum zu verfehlen, bzw. eben nur mit geringer Präzision zu finden. Um dieses Dilemma zu entschärfen werden üblicherweise *dynamische SIMPLEX Algorithmen* verwendet. D.h. in der „Anfangsphase“ der Suche wird das SIMPLEX eher groß angenommen, und die Größe entsprechend dem Suchverlauf angepaßt, also bspw. in der Nähe des Optimums sehr klein definiert.

3. Aus den „Höhen“ der Eckpunkte kann nun leicht die ungefähre Richtung des Optimums (des Gipfels in topographischer Veranschaulichung) ermittelt werden. Ein neues SIMPLEX, das sich näher dem Optimum befindet wird nun ermittelt, wie im nächsten Punkt beschrieben wird:
4. Der Eckpunkt mit der schlechtesten Bewertung wird entlang der Verbindungslinie der „besten“ Punkte gespiegelt. Der schlechteste Punkt im SIMPLEX wird durch den neuen (gespiegelten) Punkt ersetzt, wodurch ein *neues* SIMPLEX entsteht.
5. goto 2 (d.h. Wiederaufnahme der Bewertung, Ordnen, Spiegeln, ...) Sobald das Optimum erreicht ist<sup>12</sup>, wird diese Prozedur abgebrochen.

Eine Illustration zu diesem Schema findet sich in Abb.6. Im höherdimensionalen Suchraum, d.h. für den Fall, daß mehr als zwei Parameter zu optimieren sind, ist der SIMPLEX natürlich — wie schon erwähnt — **kein** Dreieck mehr. Im *drei-dimensionalen* Fall ist der SIMPLEX bspw. eine *Pyramide*.

Die prinzipielle Einschränkung der Hill-Climbing Algorithmen wurden schon weiter oben angedeutet: **Es besteht keine Garantie ein globales Optimum zu finden, da prinzipbedingt immer nur das nächst gelegene Optimum gefunden wird.**

Eine Strategie, die man wählen sollte um einen Einblick in den Suchraum zu bekommen ist, mehrere Suchläufe mit unterschiedlichen Ausgangspunkten durchzuführen. Wird immer dasselbe Optimum gefunden, so kann man mit gewisser Wahrscheinlichkeit davon ausgehen, das globale Optimum gefunden zu haben — Sicherheit kann allerdings auch so nicht erzielt werden! Bei komplizierten Suchräumen wird aus verständlichen Gründen (die Wahrscheinlichkeit das globale Optimum zu erreichen sinkt mit der Komplexität des Suchraumes dramatisch) allerdings auch diese Suchstrategie mit hoher Wahrscheinlichkeit scheitern. Trotzdem sind Hill-Climbing Strategien in vielen Fällen eine günstige Wahl, z.B. wenn

- die Bestimmung einzelner Werte sehr aufwendig oder teuer ist
- die „Beschaffenheit“ der Suchlandschaft und ein günstiger Ausgangspunkt bekannt sind, bzw. die Suchlandschaft so einfach ist (nur ein Optimum), daß dieses Verfahren als ausreichend betrachtet wird
- man sich sicher ist, bereits nahe am gesuchten (globalen) Optimum zu sein, und letztlich nur die letzten Schritte zum Optimum fehlen (bspw. Suche von Nullstellen komplexer Funktionen, nachdem bereits durch andere Verfahren die ungefähre Lage bestimmt wurde)
- auch aus obigen Gründen das Suchen des globalen Optimums nicht möglich oder vielleicht auch gar nicht unbedingt notwendig erscheint (z.B. im Falle der Optimierung von Ausbeuten chemischer Prozesse, wo schon eine deutliche Verbesserung — zum nächsten lokalen Optimum — als Erfolg gewertet werden kann).

<sup>12</sup> Das Definieren eines Abbruchkriteriums kann in vielen Fällen ein Problem darstellen. Im Falle des statischen SIMPLEX-Algorithmus ist dies einfacher, da das Optimum mit der gewählten Genauigkeit (Größe des SIMPLEX) quasi „umkreist“ wird.

In allen anderen Fällen sollte die Verwendung anderer Optimierungsstrategien überlegt werden, wie bspw. die Anwendung *Genetischer Algorithmen*.

### 4.3 Monte-Carlo Zufalls-Suche

Eine Strategie der ganz anderen Art, die zur Suche in von Parametern „aufgespannten“ Suchräumen denkbar ist, ist eine Suche, die auf reinem Zufall basiert. Die Vorgangsweise ist eine sehr einfache: Es werden ein gewisse Anzahl von Punkten *zufällig* im Suchraum ausgewählt und die „Höhe“ an diesen Punkten gemessen. Der beste Wert wird ausgewählt und als Optimum verwendet.

Es ist klar, daß die Wahrscheinlichkeit mit diesem Algorithmus (zumindest wenn relativ wenige Werte gemessen werden) ein globales Optimum zu finden äußerst gering ist, bzw. der Aufwand extrem hoch wird. Allerdings können solche Strategien für andere Zwecke verwendet werden. Beispielsweise ist es unter gewissen Voraussetzungen möglich so die „Rauhigkeit“ oder auch „Steilheit“ in bestimmten Bereichen der Suchlandschaft abzuschätzen.

Der Grund, warum diese Strategie an dieser Stelle erwähnt wird, ist allerdings ein anderer: so schlecht die Leistung als Optimierungsmethode auch sein mag, jedenfalls weist diese Strategie einen Vorteil gegenüber den *Hill-Climbing* Algorithmen auf: Es besteht nicht die Gefahr, sich an einem lokalen Optimum „festzufressen“, vielmehr wird ohne Einschränkung der gesamte Suchraum betrachtet.

### 4.4 Analogien zu den biologischen Prinzipien — Der Weg zu Genetischen Algorithmen

Vergleicht man die in den Abschnitten 4.2 und 4.3 beschriebenen Suchstrategien mit den in Abschnitt 3 angerissenen biologischen Prinzipien, so kann man — die Biologen mögen diese grobschlächtigen Vergleiche verzeihen — das Prinzip des Crossover mit der mathematischen Strategie des Hill-Climbing vergleichen. Die Zufallssuche könnte als Analogie zur zufällige Komponente die die Mutation einbringt interpretiert werden.

Somit scheint der Weg zu genetischen Algorithmen klar: es handelt sich um den Versuch die Vorteile von Hill-Climbing Methoden im Sinne lokaler Optimierung mit dem Vorteil breiter Suche der Monte-Carlo Methoden zu vereinigen.

## 5 Implementation

### 5.1 Codierung

Nun stellt sich nach den vorigen Abschnitten unmittelbar die Frage, wie man die evolutionäre Strategie verwenden kann um die beschriebenen Optimierungsprobleme zu lösen. Evolution im biologischen Sinne bedeutet ja etwa, daß die Eigenschaften einer Art sich so verändern (verbessern), daß eben diese Art über bessere Chancen in der Umwelt verfügt. Durch die Evolution wird letztlich die Erbinformation der Individuen verändert. Diese wir-

ken ja direkt auf die Ausprägung des Individuums. Der Analogieschluß zu den hier erwähnten Optimierungsproblemen wirft also zunächst folgende Frage auf:

Wie kann man den *genetischen Code* im Computer nachbilden und in weiterer Folge, wie kann man die biologischen Vorgänge der *Selektion*, der *Mutation* und des *Crossover* am besten auf bestehende Probleme anpaßt<sup>13</sup>

Die Art der Repräsentation des Gens die wir für die Umsetzung im Computer wählen ist zweifellos willkürlich — allerdings nicht bedingungslos! Immerhin sollte eine Codierung gewählt werden, auf deren Basis sich die aus der Genetik kopierten Strategien möglichst gut verwirklichen lassen. In der Praxis hat sich im wesentlichen eine Möglichkeit durchgesetzt, die auch von John Holland mathematisch fundiert wurde (*schema theorem*, beschrieben z.B. in [14]):

An Stelle des genetischen Codes im biologischen Sinne, tritt eine *Binärzahl*. D.h. wir definieren unser „Computergen“ als eine Folge einer bestimmten Anzahl von 0 und 1. Mit diesem Gen werden dann, wie im nächsten Kapitel beschrieben wird, die Operationen *Crossover*, *Mutation* und *Selektion* durchgeführt. Da das binäre Gen einen „realen“ Sachverhalt, also z.B. Variablen repräsentiert, ist die Länge dieses binären Gens abhängig von der Problemstellung.

Selbstverständlich ist auch die *Art* der Codierung stark von der Art des Problem es abhängig. Entsprechend der in Abschnitt 2.2 auf Seite 5 erwähnten Kategorien, die auch mit den Abkürzungen **NUM** für numerical parameter estimation also Parameter Optimierung, **SUB** für Subset-Selection (Probleme, bei denen aus einer Gruppe von Elementen ausgewählt werden soll) und **SEQ** für Sequencing Probleme (also Anordnungsprobleme, bzw. auch allgemeiner *kombinatorische Probleme* fallen in diese Gruppe), nach [14] bezeichnet werden.

### 5.1.1 NUM – Parameter Estimation Probleme

Im Modell liegen üblicherweise mehrere Parameter, im Definitionsbereich der *reellen Zahlen* vor, deren Optimum gefunden werden soll. Innerhalb des Algorithmus arbeiten wir aber, wie eben beschrieben, nur mit dem *binären* Gen. Das bedeutet, das diese Parameter in irgendeiner eindeutigen Form in ein binäres Gen umgewandelt werden müssen.

Diese Problematik ist in der Tat, wie viele Versuche zeigen [23] ein sehr fundamentales Problem. **Bei ungünstiger Wahl der Codierung kann die Leistungsfähigkeit des gesamten Algorithmus in Frage gestellt werden!** Ich will nun, ausgehend von mehreren reellen Parametern<sup>14</sup>, die zwei wichtigsten Möglichkeiten, diese Abbildung<sup>15</sup> vorzunehmen, beschreiben.

---

<sup>13</sup> Es sei an dieser Stelle nochmals erwähnt, daß hier nicht die Natur nachgeahmt werden soll, d.h. es sollen in diesen Anwendungen keine wie immer gearteten biologischen Vorgänge simuliert werden. Vielmehr wird versucht, die Strategie die die Natur selbst anwendet um Anpassungen vorzunehmen, zu übernehmen und auf eigene Probleme anzuwenden. Der Gedanke, der dahinter steckt, ist: Prinzipien, die in der Natur Optimierungsarbeit leisten, sollten dies auch im mathematischen Sinne tun.

<sup>14</sup> sollten Parameter aus dem Definitionsbereich der natürlichen Zahlen stammen, so vereinfacht sich die Vorgangsweise, da der erste Schritt entfällt

<sup>15</sup> Die Umwandlung, also Transformation der Parameter in das binäre Gen kann mathematisch als *Abbildung* verstanden werden.

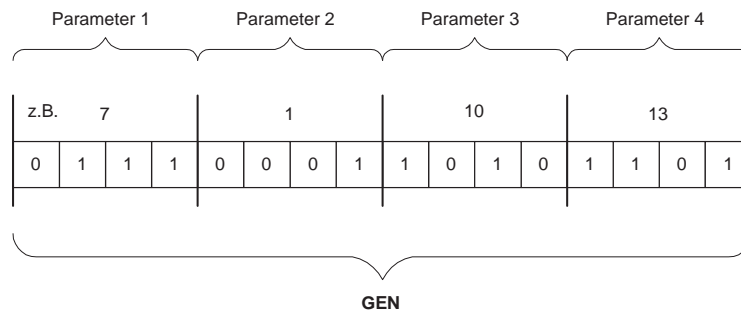


Abbildung 7: Binäre Transformation eines Parametersatzes eines mathematischen Modelles in ein Gen

**„Einfache“ Binäre Codierung** Die einfachste Möglichkeit der Codierung ist es, die Parameter zu natürlichen Zahlen<sup>16</sup>, diese in Binärzahlen umzuwandeln und aneinanderzuhängen:

1. Zunächst muß die reelle Zahl, die durch den Parameter ausgedrückt wird, in eine natürliche Zahl umgeformt werden, das geschieht üblicherweise durch folgende Transformation

$$P_T = \text{int}(10^n \cdot P) \tag{3}$$

wobei  $P_T$  den transformierten Parameter darstellt und  $n$  die Anzahl der zu berücksichtigenden Dezimalstellen. In Worten: Die Kommastelle des reellen Parameters wird je nach gewünschter Genauigkeit nach rechts verschoben und dann auf eine Ganzzahl gerundet.

**Beispiel:** 35.23543 soll mit einer Genauigkeit von 2 Nachkommastellen transformiert werden, so ergibt dies: 3524.

Dieser transformierte Parameter ist aus dem Definitionsbereich der natürlichen Zahlen, kann also ohne weiteres in eine binäre Zahl umgerechnet werden. Dieser Vorgang wird für alle zu optimierenden Parameter wiederholt<sup>17</sup>.

2. Die binären Parameter werden nun in einer Folge aneinandergereiht. Diese Reihe wird jetzt als Gen definiert, der Repräsentant des ganzen Parametersatzes. Diese Vorgangsweise ist in Abb.7 illustriert.

<sup>16</sup> Als natürliche Zahlen  $N$  versteht man Zahlen der Folge: 0, 1, 2, 3, ...

<sup>17</sup> Soll auch das Vorzeichen berücksichtigt werden, so wird ein weiteres Bit verwendet. Ist der Wert dieses Bits bspw. 1 so ist die Zahl negativ.

Binärer Code	Gray Code	Dezimalzahl
000	000	0
001	001	1
010	011	2
011	010	3
100	110	4
101	111	5
110	101	6
111	100	7

Tabelle 1: **Vergleich: Binäre und Gray Codierung.** Ein wichtiger Unterschied zwischen Binär- und Graycode besteht darin, daß sich bei Addition oder Subtraktion des Wertes 1 im Graycode jeweils nur *ein* Bit ändert, während die Änderung beim Binärcode auch *mehrere* Bits gleichzeitig betreffen kann.

**Binäre Codierung mit Gray-Code** Allerdings zeigt sich in der Praxis, daß diese Form der Codierung bedingt durch die Stelligkeit der binären Codierung nicht optimal ist, da jedem Bit ein anderes Gewicht zukommt. Ein Invertieren — also Umdrehen — des niederwertigsten Bits verursacht eine Veränderung des Parameters um den Wert 1, des zweiten Bits um 2, der dritten um 4, des  $n$ -ten um  $2^{n-1}$ .<sup>18</sup>

Das ist normalerweise *nicht* wünschenswert.

So findet oft anstelle des einfachen binären Codes der *Graycode*<sup>19</sup> Verwendung (siehe auch Tab.1), der zwar auch über eine *Stelligkeit* — wie oben erwähnt — verfügt, die sich aber bei weitem nicht so unangenehm äußert, wie die des einfachen binären Codes.

Das Prinzip der Transformation bleibt allerdings das gleiche. Auch in den folgenden Operationen ändert sich nichts, das Gen liegt auch hier in Form von 0 und 1 vor. Weitere Details zum Graycode finden sich unter anderem in [23].

Selbstverständlich müssen die Parameter, bevor sie wieder in das Modell eingesetzt werden, rücktransformiert werden. D.h. innerhalb des GA wird nur mit den binären oder gray codierten Genen (= Bitstrings) gearbeitet, extern — also in den Modellen — natürlich mit den reellen Parametern!

### 5.1.2 SUB – Subset Selection

Unter *Subset Selection* versteht man das Problem, aus einer *Menge von Elementen* nach bestimmten Kriterien eine *Untermenge* auszuwählen. Beispielsweise könnten eine Menge von Umweltmeßdaten zur Verfügung stehen. Aus dieser Menge von Variablen sollen nun

<sup>18</sup> Im Falle *dezimaler* Zahlen kann die Stelligkeit vielleicht einsichtiger veranschaulicht werden: Bei der Zahl 3742 führt eine Änderung der ersten Stelle (von links aus betrachtet) um den Wert 1 zu einer Änderung des Wertes der Zahl um 1000, der zweiten Stelle um 100, der dritten Stelle um 10 und der letzten um 1.

<sup>19</sup> Der Vollständigkeit halber sollte erwähnt werden, daß es auch verschiedene Arten der Gray-Codierung gibt, die allerdings über ähnliche Eigenschaften verfügen. Die hier vorgestellte Art ist allerdings eine, die im Rahmen von genetischen Algorithmen sehr häufig eingesetzt wird.

nur *diejenigen* für eine Modellierung verwendet werden, die für das betrachtete Problem relevant sind.

Verschiedene Codierungen für diese Problemstellung sind denkbar:

1. Die Anzahl der Bits im Bitstring ist so groß wie die Anzahl der Elemente der Menge aus der gewählt wird. Jede Stelle im Gen entspricht dem Vorhandensein (1) oder der Abwesenheit (0) des entsprechenden Elementes in der ausgewählten Menge. (01001001) beispielsweise bedeutet, daß das 2., das 5. und das 8. Element ausgewählt sind.
2. Die Anzahl der Elemente die ausgewählt werden sollen ist bekannt, bzw. wird festgesetzt: Bezeichnen wir die Anzahl der Elemente die gewählt werden sollen mit  $n$ :
  - Das Gen besteht aus  $n$  Variablen.
  - Jede dieser  $n$  Variablen codiert das Vorhandensein eines Elementes. Hätte das binäre Gen beispielsweise den Wert 001011101111, so bedeutet daß Element 1 (001), 3 (011), 5 (101) und 7 (111) ausgewählt sind, wenn binär codiert wird.

Das Problem bei Variante 1 ist, daß die Anzahl der gewählten Elemente mit einer sinnvollen Bewertungsfunktion begrenzt werden muß, der Nachteil der Variante 2 liegt klarerweise darin, daß die Anzahl der selektierten Elemente im vorhinein bestimmt werden muß und daß im Algorithmus verhindert werden sollte, daß zwei oder mehrere gleiche Elemente ausgewählt werden.

### 5.1.3 SEQ – Sequencing

Eine Implementierung von Sequencing-Problemen (also bspw. *traveling salesman* Problem) könnte ähnlich angesetzt werden, wie die Variante 2 im Abschnitt 5.1.2 beschrieben. Allerdings müßte natürlich im Algorithmus garantiert werden, daß es zu keinen Wiederholungen der Elemente kommt.

## 5.2 Algorithmus

### 5.2.1 Initialpopulation

Der erste Schritt einer GA-Optimierung besteht darin, eine Population zu generieren. Die Populationsgröße liegt im allgemeinen zwischen 50 und 1000 Individuen, wobei man unter einem Individuum ein einzelnes Gen (also einen Parametersatz) versteht. Die Werte der Ausgangspopulation werden in den meisten Fällen mit einem Zufallsgenerator bestimmt. Natürlich können auch *spezielle* Werte gewählt werden, allerdings besteht dann auch wieder die Gefahr, den Algorithmus „auf die falsche Fährte“ zu bringen.

### 5.2.2 „Bewertung“ der Population – Selektion

Der nächste Schritt ist die Bewertung der Population. Es werden also die Gene zu den Parametern rücktransformiert und in das jeweilige Modell eingesetzt. Je nach der Güte der

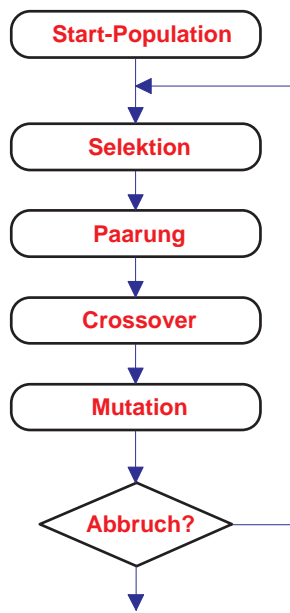


Abbildung 8: Struktogramm des genetischen Algorithmus

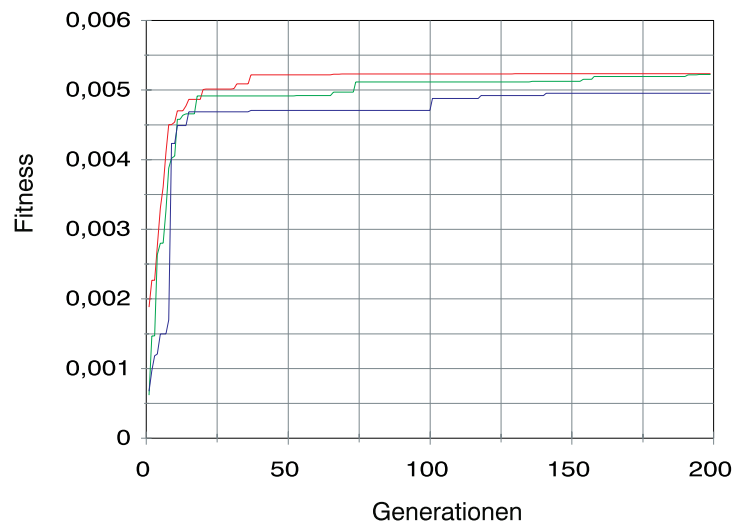


Abbildung 9: Fitness Entwicklung des besten Individuums jeder Generation dreier GA-Simulationsläufe.

Anpassung erhält jedes Gen eine Bewertung (z.B. den reziproken Wert<sup>20</sup> der *least squares*, siehe auch Beispiel in Abschnitt 5.5 auf Seite 24). Der numerische Wert dieser Bewertung wird auch als *Fitness* des Gens bezeichnet, wobei zu beachten ist, daß eine gute Bewertung sich in hohen Werten für die Fitness niederschlagen müssen<sup>21</sup>. An dieser Stelle folgt die *Selektion* :

Auch für die Selektion wurde eine große Anzahl unterschiedlicher Methoden publiziert. Oft verwendete Implementationen sind bspw.:

- Die neue Population ersetzt die alte (Eltern-) Population vollständig
- Die  $n$  besten Individuen werden in die neue Generation übernommen
- Die Vererbung erfolgt proportional zur Fitness der Individuen, d.h. die Besten werden mehrfach vererbt, die schlechteren weniger oft, diejenigen, deren Fitness niedriger als der Durchschnitt ist, werden verworfen.
- Weiters wurden Schemata mit Alterung vorgeschlagen, ...

Problematisch bei vielen Implementationen ist die Tatsache, daß das besten Gen verloren gehen kann, was üblicherweise nicht wünschenswert ist. So wird oft ein „Seitenweg“ für das beste Individuum gewählt, indem es gespeichert, und unverändert in die neue Population wieder eingesetzt wird. Durch diese Vorgangsweise ist garantiert, daß die Fitness des jeweils besten Gens der Population im Laufe der Optimierung nur Ansteigen oder wenigstens gleich bleiben, jedenfalls aber nicht schlechter werden kann.

Abb. 9 zeigt beispielhaft die Entwicklungen der Fitness-Werte des jeweils besten Gens jeder Generation. In diesem Beispiel wurde drei Läufe durchgeführt. Als nächstes folgen die Schritte *Paarung*, *Crossover* und *Mutation*.

### 5.2.3 Paarung (Mating)

Bei der *Paarung* werden aus Individuen aus der Population Paare gebildet, die dann dem Prozeß des Crossover unterworfen werde. Die Paare werden üblicherweise zufällig ausgewählt. Die Anzahl der Paare die gebildet werden, können über den Parameter *probability of mating* (also Wahrscheinlichkeit, daß eine Paarung zweier Gene erfolgt) gesteuert werden<sup>22</sup>.

### 5.2.4 Crossover

Unter *Crossover* versteht man — analog zur Genetik — einen Informationsaustausch zwischen den beiden gepaarten Individuen. Algorithmisch bedeutet das nichts anderes, als den Austausch von Bits zwischen den beiden Bitstrings.

<sup>20</sup> Unter dem reziproke Wert der Zahl  $a$  versteht man den Wert  $1/a$ . Der reziproke Wert der Zahl 3 ist also  $1/3$ .

<sup>21</sup> GAs *Maximieren* prinzipiell, da in diesem Fall aber möglichst niedrige Werte erwünscht sind, wird der reziproke Wert als Qualität herangezogen

<sup>22</sup> An dieser Stelle sollte erwähnt werden, daß unzählige Variationen der einzelnen Schritte des Algorithmus vorgeschlagen wurden. In diesem Rahmen können nur einige oft verwendete besprochen werden.

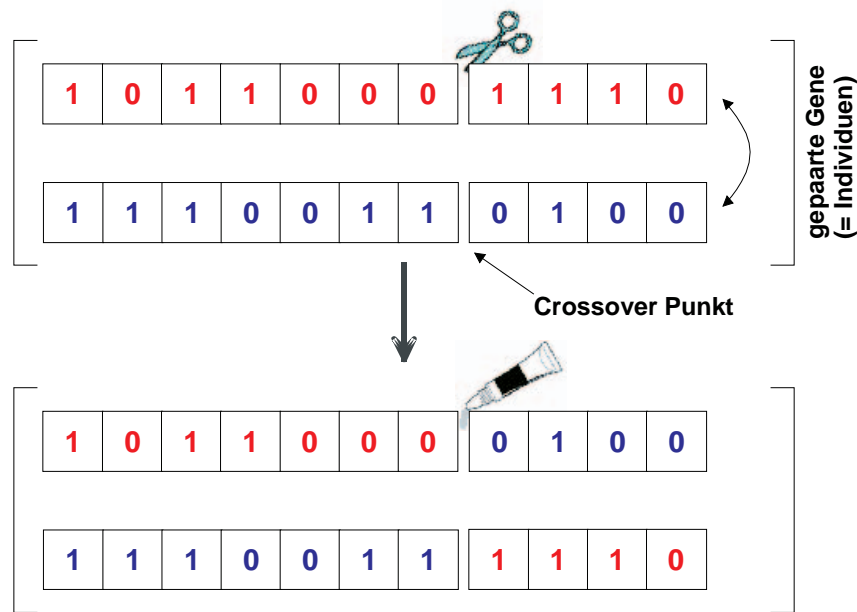


Abbildung 10: One-Point Crossover

Die einfachste Implementation (die üblicherweise nicht verwendet wird) ist das sogenannte *one-point Crossover*, das in Abb.10 dargestellt ist. Diese Variante weist noch die größte Parallelität zum natürlichen „Vorbild“ auf. Im allgemeinen werden *two-point* oder *uniform Crossover* verwendet. Two-point Crossover ist vergleichbar mit one-point crossover, nur mit dem Unterschied, daß das Gen eben nicht nur an einer Stelle, sondern an zwei Stellen „geschnitten“ und Information getauscht wird. Die meist günstigste Variante ist das *uniform Crossover*. Dabei besteht eine definierte Austauschwahrscheinlichkeit zwischen allen Bits der beiden Bitstrings. In der Abbildung würde das bedeuten, daß von links nach rechts mit einer bestimmten Wahrscheinlichkeit die Bits an gleicher Position ausgetauscht werden.

Die Bedeutung des Crossover liegt darin, daß Individuen Information austauschen und somit mit einer gewissen Wahrscheinlichkeit das „Tochterindividuum“ über bessere Eigenschaften verfügt als die Eltern. Speziell für den Fall, daß die gepaarten Individuen sich in der Nähe desselbe Optimums liegen, ist mit einer Verbesserung zu rechnen. So wird Crossover auch als die „Hill-Climbing“ Komponente der GA beschrieben. Man sollte es allerdings mit diesen Vergleichen nicht zu weit treiben, da die Struktur der GAs sich doch sehr von der konventioneller Algorithmen unterscheidet.

Spezielles Augenmerk auf diese Problematik wird auch im *schema theorem* von John Holland gelegt, beschrieben in [14,9].

### 5.2.5 Mutation

Der letzte Schritt des Algorithmus ist die *Mutation*: An dieser Stelle werden mit einer bestimmten Wahrscheinlichkeit einzelne Bits in den Genen invertiert. Diese Wahrscheinlichkeit sollte üblicherweise eher niedrig angesetzt werden (im Prozentbereich), ansonsten besteht die Gefahr, daß zuviele gute Gene zerstört werden.

### 5.2.6 Abbruchkriterium

Jetzt muß natürlich die neu entstandene Population wieder bewertet werden (s.o.) und der oben beschriebene Ablauf wird wiederholt: Paarung, Crossover, Mutation, Bewertung, ... – solange bis entweder ein *Konvergenzkriterium*<sup>23</sup> erfüllt wird (das oftmals schwer zu definieren ist, da ja die Suchlandschaft meist nicht bekannt ist), oder eine bestimmte Anzahl von Generationen erreicht wird. Ein Überblick über den Algorithmus wird in Abb.8 gegeben. Es empfiehlt sich, sich nicht nur auf einen Optimierungs-Lauf zu verlassen, sondern mindestens 2-3 Läufe zu rechnen um die Qualität der Ergebnisse abschätzen zu können.

## 5.3 Bewertung

Wie schon im vorigen Abschnitt angedeutet, ist die Bewertung die Grundlage der Selektion — und somit der Schritt, indem das jeweilige Modell zum Zug kommt. Als Bewertung versteht man also das Einsetzen der Parameter in das jeweilige Modell und die Ermittlung eines Qualitätskriteriums. Wie schon erwähnt, werden als Bewertungskriterien oft die reziproken Werte der *sum of squared residuals* und verwendet.

Prinzipiell ist aber *jede* beliebige Bewertungsfunktion (=Qualitätsfunktion) vorstellbar — womit die vielfältigen Anwendungsmöglichkeiten der GAs erklärt wären (siehe auch Abb.3 auf Seite 8)! Auch *Modifikationen* bestehender Modelle stellen den Algorithmus vor keine Probleme, da ja *nur in die Bewertungsfunktion* eingegriffen werden muß, im Gegenteil, gerade für diesen Fall sind GAs prädestiniert. Im Falle geschlossener mathematischer Lösungen, muß für jede Änderung des Modells eine neue Lösung abgeleitet werden (soweit überhaupt möglich). Im Falle des Einsatzes genetischer Algorithmen muß nur die Bewertungsfunktion geändert werden, was üblicherweise nur wenig Arbeit benötigt.

Allerdings ist immer zu bedenken, daß die Anzahl der Bewertungen im Laufe einer Optimierung in die Millionen gehen kann! Die Zeit, die die Einzelbewertung dauert, muß unter Umständen mit einem Faktor der Größenordnung  $10^6$  multipliziert werden!! Soll beispielsweise die gesamte Optimierung maximal eine Stunde dauern, so darf bei 500 Individuen und 2000 Generationen ( $500 \cdot 2000 = 1\,000\,000$ ) die Zeit, die **eine** Bewertung<sup>24</sup> dauert **4ms**

<sup>23</sup> Einfach ausgedrückt, versteht man unter einem Konvergenzkriterium eine „Entscheidungshilfe“, nach der entschieden werden kann, ob bspw. Übereinstimmung zwischen einer Variablen und einem bestimmten Wert gegeben ist oder nicht. In Fall von Optimierungsproblemen kann das Konvergenzkriterium als ein Kriterium beschrieben werden, nach dem klar entschieden werden kann, ob weitere Suche notwendig scheint, oder ob das Optimum hinreichend genau gefunden ist. Mathematisch ausgedrückt definiert man die Konvergenz einer Folge üblicherweise wie folgt: Für den Fall von  $a \in \mathbb{R}$  sei  $a = \lim_{n \rightarrow \infty} a_n$ , so konvergiert  $(a_n)$  für jedes  $a$  mit  $-\infty \leq a \leq +\infty$  [2]

<sup>24</sup> eine Bewertung ist z.B die Berechnung der Summe der Residuen **eines** Parametersatzes!

nicht überschreiten!

Für den Fall, daß die Komplexität der Aufgabe zu hoch ist, muß man unter Umständen versuche den Suchraum wieder einzuschränken, und evt. von näherungsweise oder geschätzten Lösungen mit anderen Verfahren wie in Abschnitt 4.2 beschrieben vorlieb nehmen.

## 5.4 Kritische Anmerkungen

Im Anschluß an die Erklärung der Implementation genetischer Algorithmen sollte an dieser Stelle nochmals klargestellt werden, daß genetische Algorithmen in der hier beschriebenen Form weder den Anspruch erheben, noch über die Fähigkeiten verfügen, natürliche evolutionäre Prozesse zu simulieren.

Es soll also nicht die „Natur“ und ebenso keine wie auch immer gearteten biologischen Vorgänge nachgeahmt werden. Vielmehr wird versucht, die Strategie, die die Natur selbst anwendet um Anpassungen vorzunehmen, zu übernehmen, und auf eigene Probleme anzuwenden. Der Gedanke der dahinter steckt ist ein mit der Bionik vergleichbarer: Prinzipien, die in der Natur Optimierungsarbeit leisten, sollten dies auch im mathematischen Sinne tun. Wie schematisch und vereinfacht die Prinzipien der Evolution hier angewandt werden läßt sich auch anhand des folgenden Zitates aus [12] und [25] zeigen:

*[...] Da alle Organismen in irgendeiner Form von symbiotischen Beziehungen leben, ist Evolution immer Koevolution. Evolution ist niemals völlige Anpassung an etwas, das von der Evolution eines Phylums, einer Art oder eines Individuums getrennt werden kann. [...] Wie von Weizsäcker (1975) es formulierte, spielen „koevolvierende Systeme ... zwischen Angepaßtheit und Nichtangepaßtheit. Völlige Angepaßtheit und völlige Nischenangepaßtheit sind tödlich. [...]“*

Genetische Algorithmen sind (zumindest in dieser Form) in keiner Weise koevolvierend. Es ist auch eine völlige Anpassung an „etwas“ erwünscht, dieses „etwas“ ist eben die durch das Problem definierte Fitness (= Objective) Funktion. Diese Aspekte sollten in der Betrachtung und Interpretation nicht vergessen werden.

## 5.5 Beispiel

Anhand einer einfachen linearen Regression wird der Ansatz der genetischen Algorithmen der mathematisch-geschlossenen Lösung gegenübergestellt. In der Praxis würde wohl niemand auf die Idee kommen, ein solches Problem mit genetischen Algorithmen zu lösen, da die mathematische Ableitung bekannt ist und in jeder statistischen Software eine Option zur Lösung dieses Problems existiert. Allerdings ermöglicht dieses einfache Beispiel aus diesem Grund auch eine anschauliche Gegenüberstellung der beiden Ansätze:

**Problemstellung** Es liegt eine Anzahl von Meßwerten in zwei Variablen (x, y) vor. Beispielsweise acht Werte wie in Abb.11. Für diese Werte soll ein lineares Modell der Form  $y = kx + d$  bestimmt werden. D.h. die Parameter  $k$  und  $d$  sollen so ermittelt werden, daß die

x	y
3	10,11
6	-4,19
7	16,72
9	27,58
15	52,96
21	48,75
22	42,98
28	74,64

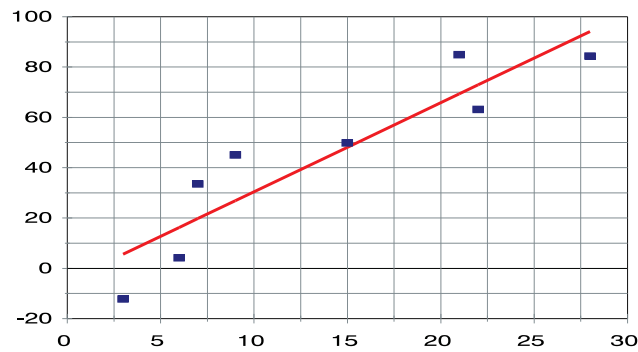


Abbildung 11: Meßwerte und lineare Regression (Ausgleichsgerade).

Gerade die Werte möglichst gut repräsentiert. Als Bewertungsfunktion werden die *Summe quadrierten Residuen* herangezogen, die zu minimieren sind (siehe auch Abb.1 auf Seite 6).

**Geschlossene mathematische Lösung** Die Berechnung der linearen Regression des linearen Modells  $y = ax + b$  durch *Minimierung der Fehlerquadrate* erfolgt nach dem folgenden Schema<sup>25</sup> [7]:

$$e_i = kx_i + d - y_i, \quad i = 1, 2, \dots, n \quad (4)$$

sind die Residuen, also die vertikale Abweichungen der Meßpunkte von der Geraden (= Residuen). Sind die *Residuen* normalverteilt, so kann man zeigen, daß die Gerade sich den Meßpunkten am besten anpaßt, wenn eben die Summe der quadrierten Residuen ein *Minimum* annimmt. Dies scheint auch intuitiv einleuchtend, daß die Anpassung der Geraden als gut angesehen wird, wenn die Absände der Meßwerte gering ist.

Gesucht ist also das Minimum der Funktion  $f$ :

$$f(k, d) = \sum_{i=1}^n (kx_i + d - y_i)^2 \quad k, d \in \mathbb{R} \quad (5)$$

Zunächst werden die relativen Extrema der Funktion  $f$  durch partielle Ableitung nach  $a$  und  $b$  bestimmt

<sup>25</sup> Zum Verständnis des Beispiels ist es nicht unbedingt erforderlich die mathematische Ableitung im Detail nachvollziehen zu können. Der wichtige Schluß für den an mathematischen Ableitungen weniger Interessierten ist, daß es in diesem Fall eine Möglichkeit gibt, eine exakte Formel abzuleiten. Nach einsetzen der Meßwerte lassen sich damit die Parameter des linearen Modelles berechnen.

$$\frac{\partial f}{\partial k} = 2 \sum_{i=1}^n ((kx_i + d - y_i)x_i) = 0 \quad \text{und} \quad (6)$$

$$\frac{\partial f}{\partial d} = 2 \sum_{i=1}^n (kx_i + d - y_i) = 0 \quad (7)$$

Um das Gleichungssystem mithilfe der *Cramerschen* Regel zu bestimmen, muß zunächst die Determinante errechnet werden:

$$D = n \sum_{i=1}^n x_i - \left( \sum_{i=1}^n x_i \right)^2 = \frac{1}{2} \sum_{i,k=1}^n (x_i - x_k)^2 \quad (8)$$

Das Gleichungssystem besitzt (logischerweise) genau eine Lösung, wenn mindestens zwei verschiedene Meßwerte von  $x_i$  vorliegen. Diese Lösung wird mit  $k_{min}$  und  $d_{min}$  bezeichnet:

$$k_{min} = \frac{1}{D} \cdot \begin{vmatrix} \sum_{i=1}^n x_i y_i & \sum_{i=1}^n x_i \\ \sum_{i=1}^n y_i & n \end{vmatrix} \quad (9)$$

$$d_{min} = \frac{1}{D} \cdot \begin{vmatrix} \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n y_i \end{vmatrix} \quad (10)$$

Ein relatives Minimum liegt vor da  $f_{aa} \cdot f_{bb} - f_{ab}^2 > 0$  und  $f_{aa}(k_{min}, d_{min}) > 0$ . Die Regressionsgerade (auch „Ausgleichsgerade“ genannt), also das gesuchte lineare Modell, hat dann die Form:

$$y = k_{min}x + d_{min} \quad (11)$$

**Ansatz des genetischen Algorithmus** Zunächst muß die notwendige Genauigkeit der beiden Parameter  $k$  und  $d$  festgelegt werden. Jeder Parameter wird mit einer Genauigkeit von 15 Bit angenommen, plus 1 Bit pro Parameter für das Vorzeichen. Mit 15 Bit plus Vorzeichen kann von -32768 bis +32768 gezählt werden kann. Da wir aber reelle Parameter benötigen und eine Genauigkeit von drei Nachkommastellen für ausreichend halten, werden die Werte noch durch 1000 dividiert. D.h. mit den 15 Bit Parametern wird der Wertebereich von -32.768 bis +32.768 abgedeckt.

Das gesamte Gen setzt sich dann aus den beiden binär kodierten 15 Bit Parametern zusammen und könnte bspw. so aussehen:

11101001001011011001111010010110

wobei die ersten 16 Stellen (1110100100101101) den Parameter  $k$  repräsentieren, die hinteren 16 Stellen (1001111010010110) den Parameter  $d$  (jeweils 15 Stellen plus Vorzeichenbit).

Der genetische Algorithmus wird nun wie beschrieben mit einer Startpopulation von bspw. 50 Individuen mit Genen, die mittels Zufallsgenerator bestimmt werden, angesetzt. Dann werden die Operationen Mating, Crossover, Mutation und Bewertung durchgeführt. Die

Bewertung muß für jedes Gen (=Individuum) durchgeführt indem das oben erwähnte 30 Bit lange Gen wieder in zwei Teile geteilt wird und die beiden Parameter  $k$  und  $d$  daraus ermittelt werden.

Diese Parameter werden dann in die Gleichung

$$resid^2 = (\sum(y - (kx + d))^2) \quad (12)$$

eingesetzt, die die Summe der quadrierten Residuen bestimmt. Da das lineare Modell aber umso besser passt, je kleiner die Abweichungen der Meßpunkte vom Modell sind, so wird auch  $resid^2$  umso kleiner wird, je besser das Modell ist. Da aber jedem Individuum ein Fitnesswert zugewiesen werden muß der, je besser das Modell paßt umso größer werden soll, wird die Fitness wie folgt definiert:

$$Fitness = 1/resid^2 \quad (13)$$

Nachdem für alle Individuen eine Fitness ermittelt wurde, wird die Selektion durchgeführt. Für den Fall, daß das „beste“ Individuum bereits ausreichend gut ist wird der Algorithmus abgebrochen. Andernfalls wird wieder von vorne begonnen, also Mating, Crossover, Mutation, Bewertung, Selektion, ...

Der Parametersatz des besten Individuums nach dem Abbruch wird als Ergebnis angenommen. Um zu überprüfen ob der Algorithmus sich nicht „verirrt“ hat, sollte mehr als ein Lauf durchgeführt werden. Kommt bei zwei odere mehreren Berechnungen dieselben Werte (oder sehr ähnliche Werte — je nach Komplexität des Problems) heraus, so kann man das Ergebnis akzeptieren.

## 6 Evolutionäre Algorithmen

Eine den genetischen Algorithmen sehr ähnliche Strategie ist die Klasse der *evolutionären Algorithmen* (EA). Sie wurde etwa zur selben Zeit wie die genetischen Algorithmen in den USA – in Deutschland von Ingo Rechenberg [18] entwickelt.

Der essentielle Unterschied zwischen EAs und GAs liegt in der **Codierung**. Während Parameter aus den reellen Zahlen im Falle von genetischen Algorithmen zu *Bitstrings* transformiert werden, werden bei evolutionären Algorithmen die Parameter *direkt* verwendet und bilden einen *Vektor*.

Dieser Vektor ist nun die Ausgangsbasis aller weiteren Schritte, die ähnlich definiert sind wie bei GAs. Eine gute Einführung findet sich in [21].

*Persönliche Anmerkung: Der Streit welcher Algorithmus „besser“ ist, scheint nicht geklärt. Tatsächlich aber haben genetische Algorithmen eine weitere Verbreitung gefunden — was vielleicht auch daran liegen mag, daß die Implementation der evolutionären Prinzipien natürlicher scheinen.*

## 7 Anmerkungen

Da es sich bei diesem Skriptum um eine in Arbeit befindliche Version handelt, würde ich mich sehr über **Rückmeldungen** freuen. In erster Linie interessiert mich natürlich, ob die Aufbereitung des Themas verständlich ist, und falls nicht, welche Stellen umgeschrieben werden sollten.

Für positive Rückmeldungen, bzw. korrigierte Skripten und „Fehlermeldungen“ bin ich dankbar (sollten korrigierte Skripten retourniert werden, so wird selbstverständlich „Ersatz“ geleistet, bzw. überarbeitete Versionen zur Verfügung gestellt).

**Aktuelle Versionen** dieses Textes findet man auf meiner Homepage (s.u.).

An dieser Stelle vielen Dank an alle „Testleser“ in erster Linie an **Petra Gruber**, auf deren Anregung hin substantielle Änderungen an Inhalt und Abbildungen vorgenommen wurden und **David Bolius**.

Rückmeldungen bitte mit Verweis auf das Datum des Textes (siehe Titelseite) **direkt** an den Autor unter:

**DR. ALEXANDER SCHATTEN**

**Institut für Softwaretechnik und  
Interaktive Systeme**

TU-Wien

Favoritenstr.. 9-11/188

URL: <http://www.schatten.info>

## Literatur

- [1] BRIGGS, JOHN und DAVID F. PEAT: *Die Entdeckung des Chaos*. dtv, München, mar 1990.
- [2] BRONSTEIN, I. N., K. A. SEMENDJAJEW, G. GROSCHE, V. ZIEGLER und D. ZIEGLER: *Teubner Taschenbuch der Mathematik*. B. G. Teubner, Stuttgart, Leipzig, 1996.
- [3] DARWIN, CHARLES: *The Origin of Species by means of natural selection, or the preservation of favoured races in the struggle for life*. J. Murray, London, 1859.
- [4] DARWIN, CHARLES: *Gesammelte Werke, Übersetzung von J. Carus*. Schweitzerbarth, Stuttgart, 1878.
- [5] DEMING, S. N. und S. L. MORGAN: *Simplex Optimization of Variables in Analytical Chemistry*. Analytical Chemistry, 45(3):278a–283a, mar 1973.
- [6] DEMING, S. N. und L. R. PARKER JR.: *A Review of SIMPLEX Optimization in Analytical Chemistry*. Analytical Chemistry, Seiten 187–202, sep 1978.

- [7] DORNINGER, D., G. EIGENTHALER und H. KAISER: *Mathematische Grundlagen für Chemiker II*. Prugg, Eisenstadt, 1981.
- [8] EIGEN, MANFRED: *Selforganisation of Matter and the Evolution of Biological Macromolecules*. *Naturwissenschaften*, 58:465–523, 1973.
- [9] GOLDBERG, D. E.: *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, Bonn, Paris, Reading Massachusetts, 1989.
- [10] HAGEMANN, RUDOLF: *Allgemeine Genetik*. Gustav Fischer, Jena, Dritte Auflage, 1991.
- [11] HOLLAND, J. H.: *Genetic Algorithms*. *Scientific American*, Seiten 44–50, jul 1992.
- [12] JANTSCH, ERICH: *Erkenntnistheoretische Aspekte der Selbstorganisation natürlicher Systeme*. In: SCHMIDT, SIEGFRIED J. (Herausgeber): *Der Diskurs des radikalen Konstruktivismus*, Seiten 159–191. Suhrkamp, Frankfurt am Main, Siebente Auflage, 1996.
- [13] *Jean-Baptiste Lamarck*. <http://www.ucmp.berkeley.edu/history/lamarck.html>.
- [14] LUCASIUS, C. B. und G. KATEMAN: *Understanding and Using Genetic Algorithms, Part 1. Concepts, Properties and Context*. *Chemometrics and Intelligent Laboratory Systems*, 19:1–33, 1993.
- [15] LUCASIUS, C. B. und G. KATEMAN: *Understanding and Using Genetic Algorithms, Part 2. Representation, configuration and hybridization-*. *Chemometrics and Intelligent Laboratory Systems*, 25:99–145, 1994.
- [16] MONOD, JACQUES: *Zufall und Notwendigkeit, Philosophische Fragen der modernen Biologie*. dtv, München, Neunte Auflage, apr 1975.
- [17] POPPER, KARL RAIMUND und JOHN C. ECCLES: *The Self and Its Brain – An Argument for Interactionism*. Springer, Heidelberg, Berlin, London, New York, 1977. dt. Ausgabe: *Das Ich und sein Gehirn*, Piper 1989.
- [18] RECHENBERG, INGO: *Evolutionsstrategie*. Friedrich Frommann, Stuttgart, 1973.
- [19] RIEDL, RUPERT: *Evolution und Erkenntnis*. Piper, München, 1982.
- [20] RIEDL, RUPERT: *Die Strategie der Genesis*. Piper, München, Siebente Auflage, 1984.
- [21] SCHÖNEBURG, EBERHARD, FRANK HEINZMANN und SVEN FEDDERSEN: *Genetische Algorithmen und Evolutionsstrategien*. Addison-Wesley, Bonn, Paris, Reading Massachusetts, 1994. ISBN 3-89319-493-2.
- [22] SCHRÖDINGER, ERWIN: *Was ist Leben?* Piper, München, 1987. Erstauflage: Cambridge University Press 1944.

- [23] VANKEERBERGHEN, P., J. SMEYERS-VERBEKE, R. LEARDI, C. L. KARR und D. L. MASSART: *Robust Regression and Outlier Detection for non-linear Models using Genetic Algorithms*. *Chemometrics and Intelligent Laboratory Systems*, 28:73–87, 1995.
- [24] WATSON, J.: *A double helix. A personal account of the discovery of the structure of DNA*. Athenum, New York, 1968.
- [25] WEIZSÄCKER, CHR. VON: *Die umweltfreundliche Emanzipation*. In: *Humanökologie*, Wien, 1975.