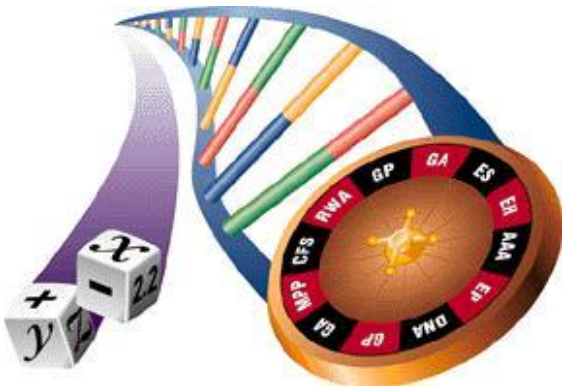
Universität Bayreuth
Rechts- und Wirtschafts-
wissenschaftliche Fakultät
Prof. Dr. A. Heinzl



Seminararbeit zur Speziellen Betriebswirtschaftslehre

„Wirtschaftsinformatik“

Thema:

Ansätze zur Entwicklung von leistungsfähigeren Genetischen Algorithmen

- Extended Compact Genetic Algorithm -

Vorgelegt von:

Andreas Feulner

XXX
XXX

6. Semester BWL

XXX
XXX

Abgabetermin:

30.09.1999

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

ABKÜRZUNGEN

BOA *Bayes Optimization Algorithm*

cGA *Compact Genetic Algorithm*

ecGA *Extended Compact Genetic Algorithm*

GA Genetischer Algorithmus

mGA *messy Genetic Algorithm*

PBIL *Population Based Incremental Learning*

sGA *simple Genetic Algorithm*, einfacher Genetischer Algorithmus

u.a. und andere

vgl. vergleiche

Wk Wahrscheinlichkeit

1 Einleitung

In einer Zeit, in der Ressourcen knapp sind, Engpässe den betrieblichen Ablauf bestimmen und Zeit als vielleicht wichtigster Wettbewerbsfaktor gilt, gewinnt die Optimierungsproblematik immer mehr an Bedeutung. Um im Wettbewerb bestehen und gegenüber der Konkurrenz potentielle Wettbewerbsvorteile generieren zu können, werden Abläufe im Unternehmen, die Vertriebs- und Beschaffungswege wie auch die Fakturierung, heutzutage rechnerunterstützt optimiert. Nicht zuletzt werden auch bei der Produktentwicklung Optimierungswerkzeuge eingesetzt, die eine schnellere Marktreife ermöglichen können.[1]

Mit wachsender Komplexität der Umweltbedingungen sind traditionelle Verfahren, wie zum Beispiel der Simplex-Algorithmus, mehr und mehr überfordert, da die Prämissen in einem Maße idealisiert sind, daß ihre Anwendung nicht oder nur begrenzt möglich wäre.[2]

Aus diesem Grund haben in jüngster Zeit Methoden an Bedeutung gewonnen, die auf der Evolutionstheorie nach Charles Darwin beruhen. Das Forschungsgebiet der Genetischen Algorithmen entwickelte sich in den letzten 20 Jahren zu einem standhaften Optimierungswerkzeug in der Umweltdynamik, das mittlerweile bereits in vielen Anwendungsgebieten implementiert worden ist.

Mit wachsender Umweltkomplexität entstanden zahlreiche Modifikationen, die mit den Problemen des ursprünglichen Genetischen Algorithmus immer besser umgehen konnten. Entscheidenden Einfluß auf die Fortentwicklung dieser Forschungsrichtung nahmen David Goldberg und Georges Harik, unter anderem auch mit seinem *Extended Compact Genetic Algorithm*.

Ziel dieser Arbeit ist es, diese Modifikation des einfachen Genetischen Algorithmus näher zu untersuchen. Dabei werde ich zum besseren Verständnis zuerst auf die Grundlagen der Genetischen Algorithmen eingehen. Im Hauptteil möchte ich schrittweise die Entwicklung vom einfachen Genetischen Algorithmus (*simple GA, sGA*), über den *Compact Genetic Algorithm* (cGA) zum *Extended Compact Genetic Algorithm* (ecGA) und die damit verbundenen Überlegungen aufzeigen. Ein Überblick über die dargestellten Algorithmen und ihrer grundlegenden Intentionen sowie ein Ausblick auf zukünftige Entwicklungen werden diese Arbeit abrunden.

---

[1] BMW zum Beispiel setzt evolutionäre Algorithmen zur Optimierung von Karosserieparametern ein (Steifigkeit, cw-Wert, Gewicht,...)

[2] vgl. dazu KINNEBROCK, WERNER 1994 S.11

2 Grundlagen der Genetischen Algorithmen

Wie bereits oben erwähnt, beruhen sämtliche Überlegungen der Evolutionären Algorithmen auf der Evolutionstheorie von Charles Darwin. Um die Funktionsweise dieser Optimierungsmethode besser erklären zu können, möchte ich die Kernaussagen der Evolutionstheorie vor der eigentlichen Thematik kurz aufgreifen.

2.1 Darwins Evolutionstheorie

Charles Darwin (1809 - 1882) beschrieb in seinem epochalem Werk *"On the origin of natural species by Means of Natural Selection*" die Evolution als einen "stufenweisen Prozeß".[3]
Er erklärte den Evolutionsprozeß mit folgenden Worten:

> "*As many more individuals of each species are born than can possibly survive; and as, consequently, there is a frequently recurring struggle for existence, it follows that any being, if it vary however slightly in any manner profitable to itself, under the complex and sometimes varying conditions of life, will have a better chance of surviving, and thus being naturally selected. From the strong principle of inheritance, any selected variety will tend to propagate its new and modified form.*"[4]

Darwin stellte fest, daß Lebewesen mehr Nachkommen erzeugen, als zum Erhalt der Rasse notwendig wären. Demnach können nicht alle überleben. Weiterhin erkannte er, daß die Nachkommen eines Elternpaares sich in Fähigkeiten und Eigenschaften unterscheiden. Die Lebewesen ringen miteinander um Nahrung, Lebensraum und Geschlechtspartner. In diesem Kampf ums Dasein überleben nur die geeignetsten, die weniger tauglichen sterben allmählich aus. Diese natürliche Selektion führt durch eine sich ständig wiederholende Anpassung an die Umweltverhältnisse zu einer Fortentwicklung der Arten. Durch die über Jahrhunderte hinweg vollzogene Selektion der Individuen geht die Veränderung und Anpassung der Tiere und Pflanzen an die Umgebung hervor. Auf dieser Erkenntnis begründete Darwin seine Selektionstheorie.[5]

Erfolgsfaktoren einer Selektion und Evolution waren demnach für ihn eine Population mit in ihren Eigenschaften verschiedenen Individuen. Die Fähigkeit die nächste Generation zu überleben, hängt von diesen Eigenschaften ab. Darüber hinaus müssen die Individuen die Fähigkeit besitzen, sich zu reproduzieren, das heißt sich fortzupflanzen.[6]

---

[3] NISSEN, VOLKER 1997 S.3

[4] DARWIN, CHARLES 1975 S.5

[5] vgl. zur Selektionstheorie DARWIN, CHARLES 1975 S.80ff

[6] vgl. KINNEBROCK, WERNER 1994 S.56

Zweifelsohne reduziert Darwins Evolutionstheorie die Wirklichkeit auf ein Modell. Jedoch ist dieses aussagekräftig genug, um die Erklärung der Entstehung der Arten begründen zu können. Durch den Versuch, die natürliche Evolution und damit die Entstehung der verschiedenen Arten auf dem Computer zu simulieren, entstanden die Genetischen Algorithmen.

John Holland veröffentlichte 1975 Algorithmen, die den Voraussetzungen für die Simulation der Evolution und später der Anwendung als Optimierungswerkzeug entsprachen. Er verstand unter dieser Art von Algorithmen ein *"tool to find solutions of optimization problems in poorly understood large spaces*".[7]

Seine Forschungsarbeiten befaßten sich mit der Theorie adaptiver Systeme, welche über die Biologie hinausgingen und auch Phänomene ganz anderer Bereiche, wie z.B. der Ökonometrie, einschloß. Seine zur Modellierung adaptiver Prozesse vorgeschlagenen "reproductive plans"[8] beruhen auf einer Abstraktion und Generalisierung des Populationskonzepts, der genetischen Codierung und genetischer Operatoren. Auf seine und von zahlreichen Forschern weiterverfolgten Überlegungen soll nun im folgenden eingegangen werden.

2.2 Einordnung der Genetischen Algorithmen

Den Anstoß zur Weiterentwicklung der evolutionären Algorithmen gaben Optimierungsprobleme, die mit bekannten mathematischen Verfahren nicht mehr zufriedenstellend gelöst werden konnten. Man erkannte, daß gewisse Problemstellungen mit auf der Evolution basierenden Systemen effizienter optimiert werden können.

Abbildung Nr.1 zeigt die Entwicklungspfade und gleichzeitig die heutigen drei Hauptströmungen der evolutionären Algorithmen.

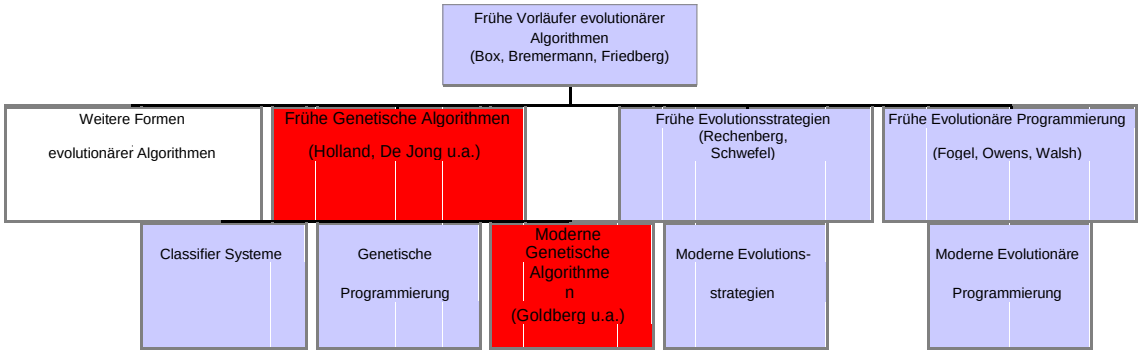


ABBILDUNG 1: Entwicklungspfade Evolutionärer Algorithmen[9]

Man unterscheidet dabei zwischen Genetischen Algorithmen, Evolutionsstrategien und Evolutionärer Programmierung. Während sich John H. Holland mit der Theorie adaptiver Systeme befaßte (Genetische Algorithmen), optimierte Ingo Rechenberg den Windwiderstand

[7] DAWID, HERBERT 1996 S.37

[8] HOLLAND, JOHN 1992 S.89ff

[9] vgl. NISSEN, VOLKER 1994 S.10

eines Stromlinienkörpers durch die Nachahmung evolutionärer Prinzipien, woraus er die Evolutionsstrategie konstatierte. Lawrence J. Fogel begründete die Evolutionäre Programmierung mit dem Entwurf eines künstlich intelligenten Automaten.[10][11]

Im Rahmen dieser Arbeit werde ich mich auf die Genetische Algorithmen konzentrieren, in der Übersicht wurden sie daher rot hinterlegt.

2.3 Struktur und Terminologie von Genetischen Algorithmen

Genetische Algorithmen basieren im Gegensatz zu traditionellen Verfahren auf dem Grundgedanken der Evolution. Evolution heißt in diesem Zusammenhang, daß eine Menge von bestehenden Lösungen durch Operatoren verändert und anschließend nach ihrer Güte bewertet und selektiert werden. Durch diesen Prozeß, der sich fortwährend wiederholt, werden die einzelnen Lösungen in ihrer Gesamtheit immer besser. Es stellt sich ein Optimum ein, wenn alle Lösungen eine einzige Lösung repräsentieren, die Menge konvergiert. Im folgenden soll nun die Terminologie und die Struktur Genetischer Algorithmen geklärt werden.

Ausgangsbasis für die Optimierung mit Genetischen Algorithmen ist eine gegebene Menge von Lösungen der Startpopulation (korrespondiert mit der Population in der Natur). Einzelne Lösungen dieser Menge werden Individuen genannt. Die Individuen sind aber nicht die Lösungen selbst, sondern liegen in einer codierten Form vor.

Abbildung 2 zeigt ein Beispiel der Codierung. Eine Lösung wird in Form einer Bit-Kette, eines Strings repräsentiert und korrespondiert mit den Chromosomen in der Natur. Dieser String kann noch einmal in seine Bestandteile, den Bits (vgl. Gene), heruntergebrochen werden. Die Werte dieser Bits (vgl. Allele) spiegeln in ihrer Gesamtheit die gegebene Lösung wieder.

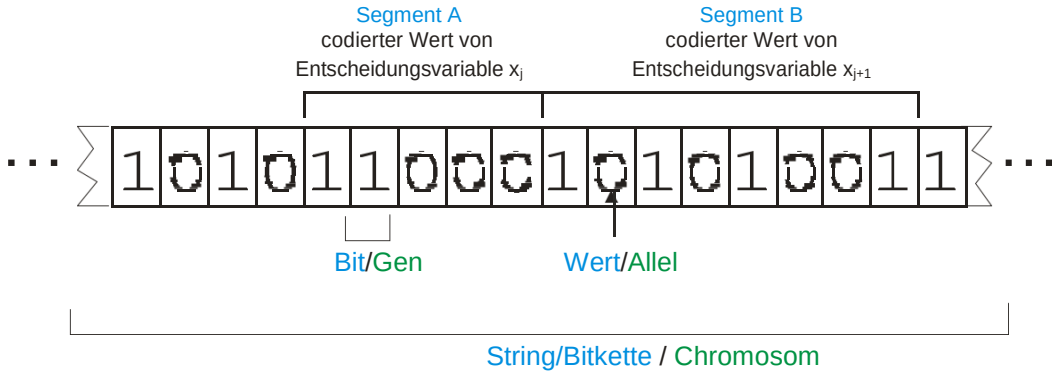


ABBILDUNG 2: Binäre Lösungscodierung des Genetischen Algorithmus[12]

[10] vgl. zu Evolutionsstrategien RECHENBERG, INGO 1973

[11] vgl. zu Evolutionärer Programmierung FOGEL, LAWRENCE 1996

[12] vgl. NISSEN, VOLKER 1994 S.35

Als Codierungsform wurde hier die Binärcodierung verwendet, da man diese in der Praxis am häufigsten antrifft. In wie viele Segmente ein String unterteilt wird, hängt von der Anzahl der Lösungsparameter ab, die zu optimieren sind. Dabei repräsentiert jedes Segment einen dieser Parameter.

Die Fitnessfunktion, die den objektiven Wert einer Lösung widerspiegelt, bewertet die Lösungen nach ihrer Güte. Je größer dieser Wert ist (bei der Suche nach einem Maximum), desto größer sind auch die Chancen, sich fortpflanzen zu dürfen. Fortpflanzen heißt hier, aus guten Lösungen werden neue, bessere generiert. Individuen der aktuellen Generation werden deshalb auch oft Eltern genannt, ihre Nachfolger Kinder oder "*offsprings*"[13].

Mit der codierten Startpopulation und der gegebenen Fitneßfunktion ist die Basis für einen Genetischen Algorithmus geschaffen. Für die Dynamik und den eigentlichen Optimierungsprozeß sorgen die genetischen Operatoren.

2.4 genetische Operatoren

Operatoren werden allgemein als Mittel und Verfahren zur Durchführung linguistischer, logischer und mathematischer Prozesse verstanden.[14] John Holland konnte 1975 die genetischen Operatoren erstmals erwähnen.[15] Er differenzierte dabei zwischen drei verschiedenen Operatoren, Selektion, *Crossover* und Mutation[16], die er aufbauend auf die Evolutionstheorie von Darwin aus der Natur ableiten konnte. Auf diese drei Operatoren möchte ich nun kurz eingehen, und dabei herausstellen, welchen Part sie im Optimierungsablauf einnehmen

In der Phase der Selektion werden aus einer Menge von Lösungen Individuen auserwählt, die sich vermehren dürfen. Um eine gute Performance zu erreichen, muß gewährleistet sein, daß Eltern mit einem höherem Fitneßwert größere Chancen haben, sich zu vermehren. Aus diesem Grund erhalten in vielen Selektionsverfahren die Individuen eine fitnessproportionale Wahrscheinlichkeit, sich fortpflanzen zu dürfen.

Da die Fitneßwerte der Startindividuen zwar untereinander verglichen werden, jedoch keine Aussage über die absolute Qualität getroffen werden kann, ist es auch möglich, daß man Schlechtes mit Schlechtem vergleicht. Deshalb, und um dem Algorithmus die Möglichkeit zu geben, so viele Individuen wie möglich zu testen, müssen weitere Operatoren greifen, die die Population verändern und neue Individuen generieren.

Diese Aufgabe erfüllen *Crossover* und Mutation. Die Performance eines Genetischen Algorithmus hängt entscheidend von der Art dieser beiden Operatoren und ihren Parametern ab.

---

[13] vgl. NISSEN, VOLKER 1997 S. 57 , GOLDBERG, DAVID 1999 S.11

[14] vgl. zur Definition "Operator" Duden 1989 S.1102

[15] HOLLAND, JOHN 1992

[16] vgl. dazu HOLLAND, JOHN 1992 S.89ff

Durch *Crossover* werden neue Lösungen generiert, indem aus der Menge der selektierten Individuen Paare gebildet werden, die dann bestimmte Ausschnitte aus ihrer Bit-Kette vertauschen. Durch diesen Vorgang entstehen immer neue Individuen, der Lösungsraum wird sozusagen durchforstet.

Der zweite Operator, Mutation, wird auf jedes Individuum der Kindergeneration angewendet, allerdings nur mit einer sehr geringen Wahrscheinlichkeit, oft nur bei 0.001. Er verändert bei seiner Anwendung die Information, die in einem Bit der Bit-Kette gespeichert ist. Bei binärer Codierung tauscht er deshalb eine Null gegen eine Eins, und umgekehrt. Der Grund für diesen Operator ist nicht offensichtlich, kann allerdings durch folgende Überlegung aufgezeigt werden. Durch zufällige Mutation wird vermieden, daß es Individuen gibt, die mit der Wahrscheinlichkeit 0 in den Optimierungsprozeß eingehen. Bei genügend Generationen kann deshalb sichergestellt werden, daß es sich bei dem gefundenen Optimum um ein globales und nicht ein lokales handelt.

Die bisherigen Ausführungen zeigten, welche Prinzipien allgemein hinter Genetischen Algorithmen stecken. GAs suchen das Optimum im Lösungsraum, indem sie Individuen der aktuellen Generation nach ihrer Fitneßfunktion bewerten und damit selektieren, wer sich fortpflanzen darf. Die Fortpflanzung selbst übernehmen Crossover und Mutation. Durch diesen kontinuierlichen Prozeß entstehen immer wieder neue Individuen, die dann bei der Selektion bewertet werden.[17]

Durch immer komplexer werdende Optimierungssituationen wurde der Ansatz der Genetischen Algorithmen weiterverfolgt und es entstanden zahlreiche Modifikationen.[18]

Im nun folgenden soll die Entwicklung vom sGA über den cGA zum extended compact genetic algorithm (ecGA) aufgezeigt werden. Dabei möchte ich besonders auf die Überlegungen eingehen, die zu diesen Modifikationen führten.

---

[17] Die einzelnen Arten der genetischen Operatoren und ihre Parameter werden in Kapitel 3.2.2 näher besprochen

[18] u.a. BOA, mGA, cGA, ecGA, PBIL

3 Ansätze zur Entwicklung von leistungsfähigeren Genetischen Algorithmen

3.1 Stärken des Genetischen Algorithmus

Traditionelle Optimierungsmethoden haben in der Vergangenheit bewiesen und beweisen auch heute noch, zu welchen Leistungen sie fähig sind. Nichtsdestotrotz werden immer häufiger Optimierungsverfahren herangezogen, die auf der naturanalogen Interpretation der Evolutionstheorie basieren.

Welche Gründe gibt es für diesen Siegeszug, wo liegen die Stärken eines Genetischen Algorithmus?

Die Stärken des einen sind die Schwächen des anderen. Welche Schwächen haben also traditionelle Verfahren?

Die Literatur differenziert drei Arten dieser Optimierungsverfahren. Es wird zwischen Verfahren unterschieden, die "*calculus-based, enumerativ and random*"[19] sind.

Verfahren, die direkt oder indirekt ein lokales Optimum berechnen, haben den entscheidenden Nachteil, daß sie dieses in der direkten Nachbarschaft des Ausgangspunktes suchen. Können diese Verfahren Funktionen mit einem Maximum noch optimieren, so treten bei Situationen, in denen mehrere lokale Optima zu finden sind, schon Probleme auf. Goldberg spricht dabei von einem "*lack of robustness*"[20], einem Genauigkeitsdefizit.

Mit der zweiten Methode können diese Probleme umgangen werden. Diese geht von einer Grundgesamtheit von Lösungen aus und berechnet für jede dieser den zugehörigen Funktionswert. Dadurch wird sichergestellt, daß das Optimum zwar gefunden wird, allerdings sehr ineffizient, da keine Parallelisierung möglich ist. Sukzessive werden alle Lösungen des Raumes abgearbeitet.

Die dritte Methode ist sehr artverwandt mit enumerativen Verfahren. Ein entscheidender Unterschied ist, daß diese den Lösungsraum nicht systematisch, sondern randomisiert durchforsten. Kein entscheidender Vorteil, wenn man bedenkt, daß auch hier keine Parallelisierung möglich ist. Goldberg spricht bei diesen beiden Verfahren von einem "*lack of efficiency*"[21], also einem Effizienzdefizit.

---

[19] GOLDBERG, DAVID 1999 S.2

[20] GOLDBERG, DAVID 1999 S.3

[21] GOLDBERG, DAVID 1999 S.5

Die Defizite traditioneller Verfahren korrelieren positiv mit der Komplexität des Problems. Das heißt, je komplexer ein Optimierungsproblem ist, desto gravierender können die Schwächen dieser Methoden in Erscheinung treten.
Worin unterscheiden sich jetzt genetische Algorithmen von diesen Verfahren?
Es gibt vier grundlegende Unterschiede:

- GAs arbeiten mit codierten Parametern und nicht mit den Parametern selbst.
- GAs suchen parallel im Lösungsraum, nicht Punkt für Punkt.
- GAs verwenden lediglich objektive Fitnessfunktionen, arbeiten blind.
- GAs navigieren die Suche im Lösungsraum randomisiert.

Mit Hilfe dieser Merkmale ist ein genetischer Algorithmus in der Lage, die Probleme bisheriger Verfahren zu bewältigen. Durch die Suche im Lösungsraum kommt es zu keinem Genauigkeitsdefizit und unter Inanspruchnahme der Parallelisierung der Lösungssuche können genetische Algorithmen sehr viel effizienter arbeiten, als dies bei traditionellen Verfahren der Fall ist.
Im folgenden möchte ich den einfachen Genetischen Algorithmus vorstellen und seine Funktionalität erklären.

3.2 einfacher Genetischer Algorithmus (simple Genetic Algorithm – sGA)

Der einfache Genetische Algorithmus geht auf John Holland und seine Forschungsergebnisse zurück. Dieser Ausgangspunkt wurde gewählt, da sich darauf aufbauend die Entwicklung der Modifikationen sehr gut darstellen läßt.

3.2.1 Optimierung mit dem sGA

Bei der ursprünglichen Form handelt es sich um einen Algorithmus mit den drei genetischen Hauptoperatoren Selektion, *Crossover* und Mutation.
Um die Arbeitsweise besser erklären zu können, möchte ich nachfolgend den Optimierungsablauf an einem Beispiel erläutern.
Die Problemstellung lautet folgendermaßen:

"Suche für die Funktion x^2 das Maximum für den x-Wertebereich [0..31]."[22]

Natürlich liegt die Vermutung nahe, daß argumentiert wird, dieses Problem könne viel einfacher gelöst werden. Ist natürlich hier nicht zu verneinen, es handelt sich um eine stetig steigende Funktion, die durch die Gleichsetzung der ersten Ableitung mit Null maximiert werden kann.
Dieses Beispiel soll allerdings nicht als Bewertungskriterium für Optimierungsmethoden gelten, sondern die Funktionalität genetischer Algorithmen erklären.
Um einen Genetischen Algorithmus verwenden zu können, müssen die Entscheidungsparameter

[22] GOLDBERG, DAVID 1999 S.15

codiert werden. Der Einfachheit halber und gemäß Goldbergs *principle of minimal alphabets*[23] wird hier und in den folgenden Ausführungen binäre Codierung eingesetzt.[24]
Der Entscheidungsparameter in diesem Beispiel ist die Variable x, die Werte von 0 bis 31 annehmen kann. Durch diesen Wertebereich wird die Länge der codierten Bit-Kette auf 5 konstatiert. Die Zahl 27 wird zum Beispiel durch folgende Bit-Kette repräsentiert:

$$27=1*2^4+1*2^3+0*2^2+1*2^1+1*2^0 \quad 11011$$

Mit Bit-Ketten der Länge 5 können demnach die Zahlen 0 (00000) bis 31 (11111) abgebildet werden.
Nachdem die Codierung festgelegt wurde und die zu optimierende Funktion x^2 die Fitnessfunktion definiert, kann jetzt eine einfache Generation mit den Operatoren Selektion, Crossover und Mutation generiert werden.
Da Genetische Algorithmen im Lösungsraum suchen, muß anfangs eine Startpopulation gebildet werden, hier bei diesem Algorithmus wird sie randomisiert erstellt. Abbildung 3 zeigt den schematischen Ablauf des einfachen Genetischen Algorithmus.

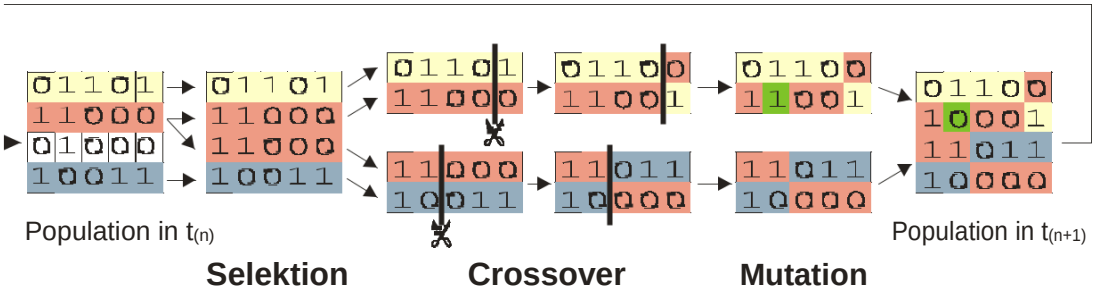


ABBILDUNG 3: Schematischer Ablauf des einfachen Genetischen Algorithmus
(eigene Darstellung)

Nachdem die Startpopulation generiert wurde, müssen sich die vier Individuen der Selektion unterziehen. Die Selektion entscheidet, welche Individuen sich vermehren dürfen. Der Fitnesswert, also der Funktionswert, den dieses Individuum annimmt, dient als Entscheidungskriterium. Das Individuum [01101] beispielsweise repräsentiert den x-Wert 13, hat damit den Fitnesswert 169. Der Anteil des individuellen Fitnesswertes am kumulierten Gesamtfitnesswert bestimmt die Wahrscheinlichkeit, selektiert zu werden. Diese Methode nennt sich Roulette-Selektionsverfahren.[25] Da Individuum [01000] einen niedrigen Fitnesswert besitzt, wird es ausgesondert.
Die nun resultierenden vier selektierten Individuen werden paarweise zusammengewürfelt und

[23] GOLDBERG, DAVID 1999 S.27ff
[24] vgl. dazu Kapitel 3.2.2 Parameter eines Genetischen Algorithmus
[25] weitere Selektionsverfahren werden in Kapitel 3.2.2 vorgestellt

pro Paar wird wiederum randomisiert die *crossover site* bestimmt. An dieser Stelle werden pro Paar die abgeschnittenen Bit-Ketten gekreuzt. Diese genetische Operation nennt sich *single point Crossover*.[26]

Der letzte Operator Mutation besitzt eine untergeordnete Rolle. Mit einer Wahrscheinlichkeit von ca. 0,001[27] ändert er willkürlich irgendein Bit eines Strings.

Die Population der nächsten Generation ist nun bestimmt.

Der fortwährende Prozeß, indem immer neue Generationen durch Crossover und Mutation geschaffen und durch Selektion bewertet werden, erzielt eine immer besser werdende Qualität der Individuen.

Der Algorithmus kann durch verschiedene Abbruchkriterien gestoppt werden. Das häufigste Kriterium ist, daß in einer Population nur noch übereinstimmende Individuen vorkommen. Man spricht dann von "*the population has converged*".[28]

Dieses einfache Beispiel hat gezeigt, auf welche Weise ein Genetischer Algorithmus die Entscheidungsparameter optimiert.

Wie schnell und wie gut diese Optimierung verläuft, hängt allerdings von zahlreichen Faktoren ab, auf die im folgenden näher eingegangen werden soll.

3.2.2 Parameter des Genetischen Algorithmus

Die Effektivität eines Genetischen Algorithmus hängt im wesentlichen von folgenden Komponenten ab:

- Lösungs-Codierung
- Populationskonzept
- Selektionsmechanismus
- Crossover-Varianten
- Crossover- und Mutation-Wahrscheinlichkeit

Es wurde bereits erwähnt, daß Genetische Algorithmen nicht mit den Entscheidungsparametern arbeiten, sondern mit ihren codierten Bit-Ketten. Die Art der Codierung ist deshalb von großer Bedeutung, da hiermit festgelegt wird, wie der Algorithmus das Optimierungsproblem sieht.

Grundsätzlich ist jede Art der Codierung möglich. Als mögliche Ansätze der Codierung sind *permutation encoding*, *value encoding* und *tree encoding* zu nennen, auf die allerdings im Rahmen dieser Arbeit nicht weiter eingegangen werden soll. Die einzelnen Bits können Zahlen beinhalten, ja sogar Buchstaben oder reelle Zahlen wären möglich. Die Möglichkeiten,

---

[26] weitere Crossover Varianten vgl. Kapitel 3.2.2

[27] vgl. zur Mutationswahrscheinlichkeit Kapitel 3.2.2

[28] HARIK, GEORGES 1999 S.5

komplexe Probleme zu codieren, steigt mit Alphabeten höherer Kardinalitäten. Der Nachteil ist allerdings, daß dann der codierte Lösungsraum komplex wird, und zudem vielleicht dadurch der eigentliche Grundgedanke Genetischer Algorithmen, den Lösungsraum ohne "*auxiliary knowledge*"[29] abzubilden – also blindes Suchen[30], verlorengeht. Aus diesem Grund hat sich die Binärcodierung gegenüber anderen Methoden in vielen Ansätzen durchgesetzt.

Darüber hinaus ist bei der Codierung der Entscheidungsparameter sehr wichtig, daß auf die Struktur der Information geachtet wird.[31] Eine falsche oder ungenügende Codierung kann aus einem einfachen, unimodalen Problem im Lösungsraum eine komplexe, multimodale Aufgabenstellung im Suchraum der codierten Lösungen machen.[32]

Die Größe der Startpopulation trägt ebenfalls einen entscheidenden Beitrag zum Erfolg eines Genetischen Algorithmus bei. Es mag überraschen, daß sehr große Populationen gewöhnlich die Performance nicht positiv beeinflussen. "Grundsätzlich besteht ein *tradeoff* zwischen Gefahr vorzeitiger Konvergenz auf suboptimale Lösungen bei kleinen Populationen und exzessiven Rechenaufwand bei großen Populationen."[33] Empirische Ergebnisse haben gezeigt, daß dieser *tradeoff* bei einer Populationsgröße von 30 bis 200 Individuen minimal wird. Bei sehr komplexen Problemen kommen jedoch auch größerer Populationen in Frage.

Auf welche Weise die Startpopulation erzeugt wird, hängt davon ab, welche Informationen über die zu suchenden Lösungen bestehen. Sind bereits gute Lösungen vorhanden, können diese die Startpopulation bilden. Aus Gründen der Heterogenität der Population, um die Suche im gesamten Lösungsraum zu ermöglichen, wird in der Praxis allerdings die zufällige Bestimmung vorgezogen.

Ein weiterer wichtiger Faktor für die Effizienz eines Genetischen Algorithmus ist die Art des Selektionsverfahren, das entscheidet, ob Individuen aussterben, oder nicht.

Sehr weit verbreitet ist das Roulette-Selektionsverfahren, das aufgrund seiner fitneßproportionalen Selektion oft eingesetzt wird. Daneben gibt es aber auch noch andere, auf die nun kurz eingegangen werden soll.

Bei sehr heterogenen Populationen konvergiert eine Population bei fitneßproportionaler Selektion sehr schnell. Eine rangbasierte Selektion ordnet aus diesem Grund die Individuen nach ihrem Fitnesswert und vergibt einen Rang. Das Individuum mit dem schlechtesten Fitnesswert erhält den Rang 1. Die Selektionswahrscheinlichkeit ist proportional zur Fitness, allerdings wird dieses Kriterium durch die Rangbildung etwas weicher betrachtet. Damit

---

[29] GOLDBERG, DAVID 1999 S.7

[30] vgl. Kapitel 3.1.1

[31] in Kapitel 3.2.3 wird diese Thematik speziell behandelt

[32] vgl. hierzu die Ergebnisse von BAECK, THOMAS 1996 S.109ff

[33] NISSEN, VOLKER 1994 S.38

erhöhen sich die Chancen der schlechteren Individuen, in die nächste Generation zu gelangen. Ein Nachteil dieses Verfahrens ist die Effizienz des Algorithmus, die Konvergenz stellt sich langsamer ein, es wird somit eine größere Anzahl an Generationen benötigt.

Das *Steady-State*-Verfahren hingegen differenziert sich von der Zuordnung von Wahrscheinlichkeiten auf die einzelnen Individuen. Ein festgelegter Parameter bestimmt, wie viele gute Individuen sich fortpflanzen dürfen. Der Rest wird ausgesondert. Die durch die selektierten Individuen entstandenen *Offsprings* füllen die Population auf und überleben gemeinsam mit ihren Eltern.

Das letzte Verfahren, das angesprochen werden soll, ist das *tournament*-Verfahren.

Bei dieser Methode werden aus der aktuellen Population S Individuen ausgewählt (mit oder ohne Zurücklegen). Das Individuum mit dem höchsten Wert kommt in die Folgepopulation. Dieser Prozeß wiederholt sich N (Populationsgröße) mal, bis die neue Population vollständig ist. Über den Parameter S läßt sich der Selektionsdruck steuern.

In einem analytischem Vergleich verschiedener Selektionsmechanismen ist Goldberg zu dem Ergebnis gekommen, daß bei proportionaler Selektion die Qualitätssteigerungsraten der Individuen am Anfang sehr hoch sind, bei rangbasierter oder *tournament* Selektion dagegen relativ konstant.

Schließlich entscheidet auch das Crossover-Verfahren über die Effizienz eines Genetischen Algorithmus. Unter der Prämisse binärer Lösungscodierung kann zwischen folgenden Methoden unterschieden werden.

- Single Point Crossover: Eine Stelle wird bestimmt, an der die Bit-Ketten beider Eltern gekreuzt werden
- Multi Point Crossover: Mehrere Stellen entscheiden darüber, wo Bit-Ketten gekreuzt werden und wo sie bestehen bleiben
- Uniform Crossover: Die Bits werden zufällig zwischen den beiden Strings vertauscht (*multi point crossover* mit N-1 Trennstellen)

Die Wahl der Methode ist unter anderen auch abhängig von der gewählten Selektionsstrategie. Nach Volker Nissen "sind stark vermischende *Crossover* dann besonders erfolgreich, wenn gleichzeitig vergleichsweise konservative Selektion betrieben wird, bei der Individuen auch viele Generationen überleben können, bis sie durch bessere Nachkommen ersetzt werden."[34]

Ein letzter, aber nicht weniger bedeutender Faktor für die Effektivität der Optimierung ist die Wahl der Wahrscheinlichkeiten, mit der die beiden genetischen Operatoren *Crossover* und Mutation in Erscheinung treten.[35] Die Höhe der *Crossover*-Rate entscheidet, welcher Anteil der aktuellen Generation direkt in die neue Generation übernommen wird. Der Grund für *Crossover*

---

[34] NISSEN, VOLKER 1994 S.59

[35] vgl. dazu OBITKO, MAREK 1999 Parameters of Genetic Algorithms

ist, wie weiter oben schon erwähnt, augenscheinlich, aus guten Individuen sollen durch Kreuzung bessere entstehen. Muß sich allerdings die Gesamtheit aller Individuen in jeder Generation dem *Crossover* unterziehen, so werden gute auch wieder durch schlechtere ersetzt. Aus diesem Grund verwendet man oft eine von 100% abweichende Rate der *Crossover*-Wahrscheinlichkeit. Zur Bestimmung der Mutationsrate liegen unterschiedliche Meinungen vor. Nissen beziffert sie zwischen 0,001 und 0,01 und stellt eine Kausalität zur Populationsgröße und Stringlänge fest.[36] Jedoch sollte sie nicht zu klein gewählt werden, da "Mutationen eine Irreversibilität im Entwicklungsprozeß der Generationen garantieren".[37]

Die Darstellung der verschiedenen Parameter zeigt, daß bei der Optimierung mit Genetischen Algorithmen viele unterschiedliche Entscheidungen getroffen werden müssen. Die richtige Wahl begründet die Effektivität des Systems.[38]

Ein weiterer wichtiger Faktor für die Performance eines Genetischen Algorithmus ist die Aufdeckung von bestimmten Bitmustern, die für hohe Fitnesswerte verantwortlich sind. Diese Überlegung wird im folgenden Kapitel aufgegriffen.

3.2.3 Schemata – Building-Block-Hypothese

Um Genetische Algorithmen in ihrem Verhalten analysieren und ihre Leistungsfähigkeit als Suchverfahren verstehen zu können, führte Holland den Begriff des Schemas ein.[39] Unter diesem Begriff versteht er "generalization of an interacting, coadapted set of genes".[40] Ein Schema entspricht einem Ähnlichkeitsmuster über dem Alphabet {0,1,*} und beschreibt eine Menge von Strings im Lösungsraum $\{0,1\}^L$, die an definierten Positionen übereinstimmen. Für jedes der L binären Gene eines Strings verwendet das Schema entweder eine 1, eine 0 oder es läßt den Wert offen durch den Platzhalter "*".

Das Schema-Konzept wird im Rahmen dieser Arbeit am Beispiel binärer Strings vorgestellt, gilt jedoch auch für Codierungen anderer Kardinalitäten.

Beispiel: Das Schema H_1=1*0* beschreibt alle Strings der Länge L=4, die eine 1 an erster Position haben und eine 0 an dritter. Das ist folgende Menge von Strings:

{1000, 1100, 1001, 1101}.

Diese einzelnen Individuen sind Instanzen des Schemas H_1, können aber auch dem Schema H_2=1*** angehören. H_2 ist gegenüber H_1 allgemeiner, man spricht deshalb von der Ordnung

---

[36] vgl. zur Bandbreite der Mutations-Wahrscheinlichkeit NISSEN,VOLKER 1994 S.54

[37] KINNEBROCK, WERNER 1994 S.78

[38] ein gute Zusammenfassung der Parameter findet sich bei NISSEN,VOLKER, 1997, S. 29ff KINNEBROCK, WERNER 1994 S.70ff

[39] vgl. HOLLAND, JOHN 1992 S.66ff

[40] HOLLAND, JOHN 1992 S.XIII

o(H) eines Schemas. Die Ordnung ist definiert als die Anzahl fixierter Positionen im String. Die Ordnung $o(H_1)$ ist demnach 2. Je niedriger die Ordnung eines Schemas ist, desto allgemeiner ist dieses. Den Abstand zwischen der ersten und letzten fixierten Position eines Schemas bezeichnet man als seine definierte Länge $\delta(H)$. Schema H_1 hat eine definierte Länge $\delta(H_1)=3-1=2$.

Während ein Genetischer Algorithmus explizit nur mit einer Population von N Individuen operiert, betrachtet er implizit eine wesentlich größere Anzahl von Schemata. Die Anzahl der bearbeiteten Schemata liegt zwischen 2^L und $N*2^L$, je nach Heterogenität der Population. Dieses Merkmal des GA bezeichnet man als "implicit parallelism".[41]

Die Anzahl unterschiedlicher Schemata der Länge L beträgt bei binärer Codierung 3^L. Während in der Startpopulation noch alle Schemata gleichberechtigt sind, so kommt es im Laufe der Optimierung zu einer Konzentration auf diejenigen mit einem hohen Fitnesswert.

Bei den wesentlichen Strategien eines Genetischen Algorithmus, dem Ausnutzen schon bekannter Strukturen (*exploitation*) und der Suche nach neuen, noch besseren Strukturen (*exploration*)[42], gilt es, zwischen diesen eine optimale Balance zu finden. Bekannte Strukturen unterlaufen der Gefahr, durch *Crossover* getrennt zu werden. Die Wahrscheinlichkeit, daß ein Schemata mit gutem Fitnesswert getrennt wird, ist proportional zu der *defining length*, also dem Abstand zwischen ersten und letzten fixem Gen des Schematas. Das heißt, je größer der Abstand ist, desto wahrscheinlicher ist es, dieses Schema zu zerstören.

An einem Beispiel läßt sich dies verdeutlichen. Schema A=11** wird nur dann zerstört, wenn die *crossing site* 1 gewählt wird. Ein Schema B=1**1 wird dagegen sehr viel wahrscheinlicher getrennt, da hier *crossing site* 1, 2 und 3 dazu führen.

Holland erkannte darüberhinaus, daß Schemata niedrigerer Ordnung mit größerer Wahrscheinlichkeit selektiert werden. Auf diesen beiden Erkenntnissen konstatierte er das Schema-Theorem.[43]

David Goldberg leitete aus diesem Theorem die Building-Block-Hypothese ab. Unter *building-blocks* versteht er "*highly fit schemata of low defining length and low order play*"[44]. Aufgrund der Erkenntnis, daß Schemata kleiner Ordnung und kleiner *defining length* für den Erfolg Genetischer Algorithmen eine große Rolle spielen, läßt sich eine gute Lösung sukzessive aus Partiallösungen zusammensetzen. Kritik übt Nissen an dieser These, da sie nichtlineare

---

[41] vgl. zum "implicit parallelism" GOLDBERG, DAVID 1999 S.20
HOLLAND, JOHN 1992 S.71ff,
BÄECK, THOMAS 1996 S.128ff

[42] aus NISSEN, VOLKER 1997 S.87

[43] für eine gut verständliche Herleitung der Schema-Theorems vgl. GOLDBERG, DAVID 1999 S.28ff

[44] GOLDBERG, DAVID 1999 S.41

Abhängigkeiten zwischen Lösungselementen außer acht läßt.[45]
Aufbauend auf der Theorie der Genetischen Algorithmen möchte ich nun zu den Überlegungen übergehen, die zu verschiedenen Modifikationen des Grundansatzes führten.

3.3 compact Genetic Algorithm – cGA

Mehr als andere Optimierungsverfahren ist das Gebiet der Genetischen Algorithmen durch eine hohe Entwicklungsdynamik gekennzeichnet. Permanent entstehen neue Verbesserungsvorschläge, neue Modifikationen. Nissen nennt für diese Dynamik drei Hauptmotive:

- Allgemeine Maßnahmen zur Steigerung der Leistungsfähigkeit
- Effizientere Lösungen spezieller Anwendungsprobleme
- Erweiterung des Anwendungsbereiches[46]

Um die Effektivität des Genetischen Algorithmus zu steigern, forschte man lange Zeit, welche Parameter vielleicht verändert werden könnten.
Die bisherigen Erkenntnisse zeigen, daß die Population Informationen über bereits erkundete Teile des Lösungsraum beinhaltet. *Crossover* und Selektion entscheiden darüber, wohin der Weg der Suche geht. Die Effekte von *Crossover* und Selektion greifen an verschiedenen Stellen in das Optimierungsgeschehen ein. Durch den Operator *Crossover* besteht die Gefahr, daß gute Schemata willkürlich getrennt werden und damit die Korrelation zwischen einzelnen Bits aufgehoben wird. Selektion dagegen verändert die Zusammensetzung der Population. Aufgrund der Tatsache, daß beide Operatoren direkt in die Generation eingreifen, versuchte man lange Zeit einen passenden Ansatz zu finden, der diese Operatoren ersetzen kann. Man stieß dabei auf die Idee, die Population als eine Wahrscheinlichkeitsverteilung über die einzelnen Bitpositionen darzustellen. Durch diese Art der Darstellung müssen nicht alle Individuen berücksichtigt werden, der Algorithmus arbeitet lediglich mit Wahrscheinlichkeiten, mit denen an bestimmten Genpositionen Einsen auftreten.
Georges Harik u.a. vom Illinois Genetic Algorithms Laboratory entwickelten unter diesen Prämissen den Compact Genetic Algorithm (cGA).[47]
Ähnlich wie beim einfachen Genetischen Algorithmus muß auch ein cGA initialisiert werden. Durch die Überlegung, ein Wahrscheinlichkeitsmodell zu verwenden, wird allerdings ein L-dimensionaler Wahrscheinlichkeitsvektor P[] mit $p_{1\text{-}L}(1)=0{,}5$ für jede Genposition initialisiert, wobei L die Länge des Strings widerspiegelt. Diese Initialisierung ist äquivalent zur randomisierten Startpopulation des sGA. Der Parameter S gibt die Anzahl der zu generierenden

[45] vgl. zur Kritik an der Building-Block-Hypothese NISSEN, VOLKER 1994 S.111
[46] zu den Motiven der Weiterentwicklung Genetischer Algorithmen vgl. NISSEN, VOLKER 1994 S.29
[47] HARIK, GEORGES u.a. 1997

Lösungen an. Jedes Gen erhält den Wert 1, wenn eine Zufallszahl zwischen 0 und 1 kleiner als der Wert des Wahrscheinlichkeitsvektors dieser Bitposition ist.

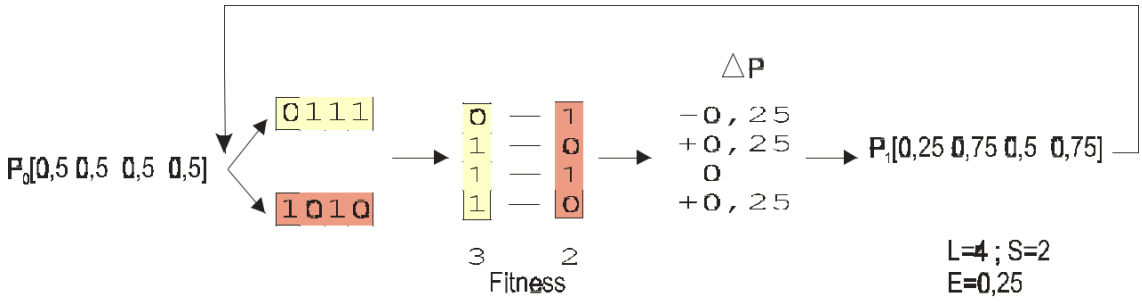


ABBILDUNG 4: Schematischer Ablauf des *Compact Genetic Algorithm* (eigene Darstellung)

Abbildung 4 zeigt den schematischen Ablauf des cGA. Ein vierdimensionaler Wk-Vektor wird initialisiert, indem für jede Bitposition die Wk für eine 1 auf 0,5 gesetzt wird. Aus diesem Vektor werden jetzt S=2 Lösungen generiert, indem für jede Genposition eine Zufallszahl zwischen 0 und 1 gewählt wird. Ist diese kleiner als der für diese Position geltende Wert des Vektors, erhält dieses Gen den Wert 1. Da bei diesem Problem die Fitnessfunktion die Anzahl der Einsen in dem String zählt, wird Individuum [0111] bevorzugt. Allerdings wird bei traditioneller Selektion auf der Bitebene ein Fehler unterlaufen, da Individuum [1010] bei Bitposition 1 einen besseren Beitrag zur Fitneß liefert als Individuum [0111]. Aus diesem Grund vergleicht man beim cGA Bit für Bit miteinander. Sind beide gleich, verändert sich der Wert des Vektors für diese Position nicht. Hat ein Bit des fitteren Individuums eine 1 (0) als Wert, das schlechtere Individuum dagegen eine 0 (1), so wird die Wk des Vektors um einen vorher definierten Wert E=0,25 erhöht (vermindert). In unserem Beispiel führt dies zu dem neuen Wk-Vektor P_1[0,25 0,75 0,5 0,75]. Dieser Prozeß wird solange wiederholt, bis der Vektor konvergiert, das heißt als Vektorelemente nur noch 1 oder 0 vorkommen.

Der Compact Genetic Algorithm kann mit der Funktionsweise eines einfachen Genetischen Algorithmus verglichen werden. Ein sGA mit N Individuen der Länge L, der *tournament selection* mit der Selektionsrate S und *uniform crossover* verwendet, kann durch einen cGA mit S Lösungen und E=1/N ersetzt werden.[48]

Experimente mit beiden Algorithmen haben gezeigt, daß sie fast auf die gleiche Weise verfahren. Jedoch benötigt der Compact Genetic Algorithm nur $\log_2 N*L$ Bits zur Speicherung, während der einfache Genetische Algorithmus N*L Bits in Anspruch nimmt.[49]

Aufgrund der geringeren Speicheranforderungen gelingt es dem cGA schneller zu besseren

[48] HARIK, GEORGES 1999 S.8

[49] HARIK, GEORGES u.a. 1997 S.4

Ergebnissen zu gelangen.[50]
Harik sieht den cGA als "*a quick way to check if a problem is easy or not.*". Weiterhin klassifiziert er diese Art von Algorithmen als *order-1 probabilistic algorithms*, also als Algorithmen, die einen Wahrscheinlichkeitsvektor P über alle Genpositionen bilden. Hiermit lassen sich Probleme mit *trivial building blocks* (Gene) lösen.
Im nächsten Kapitel soll nun ein Ansatz gezeigt werden, der auch *higher building blocks* berücksichtigt.

3.4 extended compact Genetic Algorithm – ecGA

Die Schule um Goldberg und speziell Georges Harik befassen sich mit sogenannten *deceptive* (irreführenden) Problemen und der Thematik, daß gute Schemata mit großer *defining length* mit großer Wahrscheinlichkeit nicht in die nächste Generation gelangen, da *Crossover* ihre Struktur zerstören kann.

3.4.1 *deceptive problems* und *marginal product models*

Unter *deceptive problems* versteht man Optimierungssituationen, denen traditionelle Genetische Algorithmen nicht gerecht werden.
Ein Beispiel kann die Problematik verdeutlichen.
Angenommen es handelt sich um ein Problem der Länge L=4 mit einer Population von N=8 verschiedenen Individuen.(vgl. Abbildung 5)

1000	1101	0111
1100	0010	0111
1000	1001	

ABBILDUNG 5: Optimierungsproblem mit L=4 und N=8

Wie in Kapitel 3.3 schon gezeigt wurde, kann mit dem cGA die Population auch in Form einer Wahrscheinlichkeitsverteilung repräsentiert werden. Der Vektor P[5/8 4/8 3/8 4/8] könnte diese Population repräsentieren. Jedoch ist es weder für den sGA noch den cGA möglich, Abhängigkeiten zwischen den einzelnen Genpositionen darzustellen. Gerade dies ist aber bei komplexen Problemen unabdingbar. In diesem Zusammenhang findet man in der Literatur den Begriff der *higher-order building blocks*[51]. Darunter sind Gene zu verstehen, zwischen denen Abhängigkeiten bestehen, die also nur gemeinsam einen Beitrag zur Fitness leisten können, und die nicht benachbart sind.

---

[50] experimentelle Ergebnisse in HARIK, GEORGES u.a. 1997 Anhang

[51] N.N. "Computer experiments with the ecGA" 1999 S.1

Holland erkannte schon in den Anfängen, daß "*mathematical models of genetic adaptation are based on very simple reproductive plans, where each individual allele is assigned a fitness and the fitness of any set of alleles is taken to be the sum of the fitnesses of the alleles in the set.*
...However...the fitness of an allele depends critically upon the influence of other alleles. ...The genetic operators provide for the preservation of coadapted sets by inducing a linkage between adjacent alleles, the closer together a set of alleles is on a chromosome, the more immune it is to separation by the genetic operators". [52] Folglich suchen bisher dargestellte GAs bei komplexen Problemen in die falsche Richtung, da sie jedes Gen unabhängig, separat betrachten. Die weitere Betrachtung des Beispiels aus Abbildung 5 soll dies verdeutlichen.
Angenommen zwischen Genposition 1 und 3 würden Interdependenzen bestehen, so könnte man ein zu bisherigen Methoden verschiedenes Wahrscheinlichkeitsmodell aufstellen. Abbildung 6 zeigt das von Harik vorgestellte *marginal product model*(MPM), das in der Lage ist, Abhängigkeiten zwischen den Genen zu berücksichtigen.

[1,3]	2	4
11 0	1 4/8	1 4/8
10 5/8	0 4/8	0 4/8
01 3/8		
00 0		

ABBILDUNG 6: *marginal product model*(MPM)

Betrachtet man die linke Spalte, so läßt sich unschwer erkennen, daß die Bits korreliert sind, da die Kombinationen [00] und [11] nie vorkommen. Ein entscheidender Vorteil des *marginal product models* gegenüber dem Wahrscheinlichkeitsmodell des cGA oder der Population des sGA ist, daß Abhängigkeiten zwischen einzelnen Bits aufgedeckt werden können. Diese Information würde man bei den beiden anderen Ansätzen nicht erkennen oder modellieren können.
Mit Hilfe der Entropie läßt sich die Stärke der Korrelation bzw. der Ungleichheit der Verteilung bemessen und damit kann beurteilt werden, wie stark die jeweilige Korrelation ist. Der Entropiebegriff entstammt ursprünglich der Informationstheorie und wird auch zur Bemessung des Einkommens-Ungleichgewichts herangezogen. [53]
Die Entropie ist die Summe der Informationsgehalts jeder Bit-Kombination gewichtet mit den jeweiligen Eintrittswahrscheinlichkeiten. Ist p die Wahrscheinlichkeit, daß eine bestimmte Kombination auftritt, so muß der Informationsgehalt h(p) eine abnehmende Funktion von p sein, da je unwahrscheinlicher die Kombination bestimmter Allele ist, desto interessanter ist es, wenn diese eintritt.

---

[52] HOLLAND, JOHN 1992 S.33f

[53] Gablers Wirtschaftslexikon 1995 S.966

Eine Formel, die diesen Prämissen entspricht, ist $h(p)=-\log_2(p)$. Somit ist die Entropie der Verteilung durch die Formel $H(p) = -\log_2(p_i) * p_i$ definiert.
Welche Aussagen liefert uns nun dieses Maß?
Die Entropie beträgt in diesem Fall $-(5/8)\log_2(5/8)-(3/8)\log_2(3/8)=0{,}954$. Würde jedes Paar mit p= ¼ in Erscheinung treten, so würde die Entropie H(p)=2 betragen, was dem Maximalwert in diesem Beispiel entspricht. Hätte ein Paar die Wk p=1, so berechnet sich eine Entropie von 0.

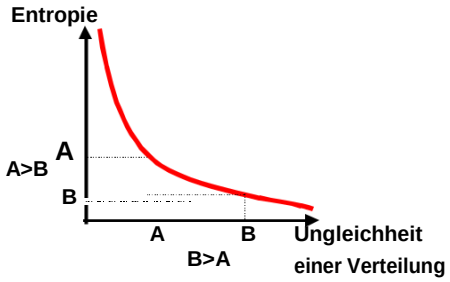


ABBILDUNG 7: Entropie und die Ungleichheit in einer Verteilung

In Abbildung 7 wird deutlich, zu welchen Aussagen wir jetzt fähig sind. Zwei Verteilungen A und B haben unterschiedliche Entropiewerte. Da Modell B einen niedrigeren Wert einnimmt, kann auf eine größere Ungleichheit der Verteilung geschlossen werden. Daraus kann gefolgert werden, daß bei Modell B stärkere Abhängigkeiten zwischen den gruppierten Genen bestehen als bei A. Durch diese Erkenntnis lassen sich sukzessive alle Building-Blocks finden und somit kann dem Problem der Trennung guter Schemata entgegengewirkt werden.
Kommen wir nun zu der Frage, welches Modell die Ausgangspopulation generiert haben könnte.
Die Informationstheorie sagt, ein Wahrscheinlichkeitsmodell ist desto besser, je weniger Speicherplatz, je weniger Informationen es benötigt, um die Population zu repräsentieren. Harik nahm diesen Gedanken in seine Überlegungen auf und stellte fest, "the particular distribution is defined as the sum representation size of the model itself and the population compressed under the model"[54]. Damit konstatiert er, welche Parameter für die Speicheranforderungen eines Wk-Modells verantwortlich sind. Diese Parameter sollen im folgenden Kapitel näher untersucht werden.

3.4.2 Das Combined Complexity Criterion

Wie bereits festgestellt, entscheidet die Informationskomprimierung über die Güte eines Wahrscheinlichkeitsmodells. Je kleiner der Speicherbedarf ist, desto besser ist das Modell. Dabei muß erwähnt werden, daß die Modelle, die beurteilt werden, die Population repräsentieren müssen, also im Grunde dieselben Informationen beinhalten.

---

[54] HARIK, GEORGES 1999 S.13

Der Speicherbedarf setzt sich nach Harik aus zwei Bestandteilen zusammen:

a) Speicherbedarf für das Wahrscheinlichkeitsmodell

b) Speicherbedarf für die komprimierte Population

Ein Wahrscheinlichkeitsmodell muß im Grunde jede Wahrscheinlichkeit speichern, da sie einem bestimmten Ereignis zugeordnet ist. Allerdings summieren sich die Wk pro betrachteter Bitposition zu 1. Das Wk-Modell des cGA benötigt für das Beispiel aus Abbildung 5 4Bits, im Vergleich dazu würde der sGA aufgrund seiner Individuenbetrachtung 4*8=32Bits in Anspruch nehmen. Das MPM aus Abbildung 6 müßte dagegen 5Bits speichern.

Die komprimierte Population kann jedoch effizienter im Modell [0,2][1][3] gespeichert werden, da die gemeinsame Betrachtung der Gen [0] und[2] eine geringere Entropie hat als die Summe der Entropien der einzelnen Gene.

Folglich ist das Modell das beste, das die Summe der beiden Speicheranforderungen minimiert.[55]

Formal heißt dies:

a) Speicherbedarf für das Wahrscheinlichkeitsmodell

$$\text{model complexity} = \log(N+1)_i \quad (2^{S_i} - 1)$$

b) Speicherbedarf für die komprimierte Population

$$\text{compressed population complexity} = N \sum_i Entropy(M_i)$$

Die Summe der beiden Speicheranforderungen bezeichnete Harik als *combined complexity* [56]

3.4.3 linkage learning mit dem ecGA

Es wurde bereits in Kapitel 3.4.1 gezeigt, daß es wichtig ist, abhängige Gene, sogenannte Building-Blocks, zu identifizieren. Durch die Erkenntnisse, die mit Hilfe der Komplexitätskriterien gewonnen wurden, konnte Harik einen Genetischen Algorithmus kreieren, der in jeder Generation nach *Building-Blocks* Ausschau hält, im Fachjargon *linkage learning* genannt. Er sucht ein Wahrscheinlichkeitsmodell, das die Population maximal komprimiert, das heißt die Summe der beiden Komplexitätskriterien, die *Combined Complexity*, minimiert.

---

[55] vgl. dazu HARIK, GEORGES 1999 S. 13

[56] vgl. zur formalen Herleitung der Komplexitätskriterien HARIK, GEORGES 1999 S.11ff

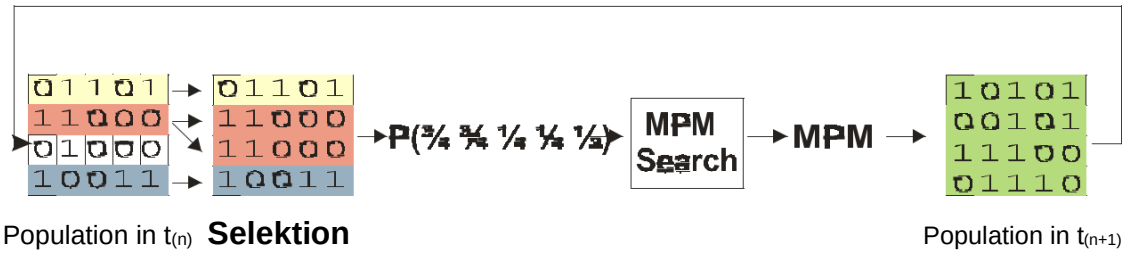


ABBILDUNG 8: Schematischer Ablauf des Extended Compact Genetic Algorithm
(eigene Darstellung)

Abbildung 8 zeigt den schematischen Ablauf der Optimierung.
Der ecGA startet wie der einfache GA mit einer randomisierten Startpopulation. In einem zweiten Schritt unterzieht sich diese Population der *tournament*-Selektion. Die entstehenden Individuen werden nun in einem Wahrscheinlichkeitsmodell transferiert, das jedes Gen separat betrachtet. Hier setzt die eigentliche Stärke des ecGA ein. Es werden nach und nach alle möglichen *marginal product models* generiert und auf Basis ihrer *combined complexity* bewertet. Das MPM mit der niedrigsten Rate wird ausgewählt und generiert die neue Generation. Diese muß sich wieder dem zweiten Schritt, der Selektion unterziehen.
Im Vergleich zum compact GA benötigt der ecGA eine Population und den genetischen Operator Selektion, da durch das gewählte *marginal product model* die Population jede Generation auf Basis eines unterschiedlichen Modells gebildet wird.
Untersuchungen haben ergeben, daß der Extended Compact Genetic Algorithm bei der Lösung von *deceptive problems* signifikant schneller ist als der einfache GA.[57]

4 Fazit und Ausblick

In der vorliegenden Arbeit wurde mit den Grundlagen der Genetischen Algorithmen begonnen. Der Ablauf des einfachen GA konnte zeigen, welche Funktionalität hinter dieser Art von evolutionärer Algorithmen steckt. Darauf aufbauend wurde Hollands Schema-Theorem und die Building-Block-Hypothese vorgestellt. Mit dem compact GA konnte daraufhin ein Ansatz vorgestellt werden, der auf die Population und die genetischen Operatoren in ihrer Gesamtheit verzichtet, indem er die Information in einem Wahrscheinlichkeitsmodell repräsentiert. Aufgrund der Building-Block Problematik und irreführender (*deceptive)* Probleme, wurde mit Hariks Komplexitätskriterium *combined complexity* ein Ansatz gefunden, der in der Lage ist, mit diesen umzugehen. Die Darstellung der Implementierung dieses Ansatzes in das Grundgerüst eines Genetischen Algorithmus vervollständigte die Arbeit.
Experimentelle Ergebnisse zeigen, daß der Extended Compact Genetic Algorithm bei

[57] experimentelle Ergebnisse in HARIK, GEORGES 1999 S.14ff

komplexen Problemen weitaus schneller zu guten Ergebnissen kommt, als beispielsweise der sGA oder der cGA. Durch *linkage learning* und der Identifikation von *building-blocks* ist er in der Lage komplexe Sachverhalte in einem MPM zu repräsentieren, und damit die Grundlage zu schaffen, für die Generierung der nächsten Population. Harik beschreibt den Erfolg mit seinen eigenen Worten als "the long-sought solution to the linkage learning problem."[58]

Die vorgestellten Methoden sind nur ein Ausschnitt aus dem Forschungsgebiet der Genetischen Algorithmen. In der Zukunft wird es das Ziel sein, evolutionäre Algorithmen und andere künstliche Intelligenz wie Neuronale Netze oder Fuzzy-Systeme zu Hybridsystemen zu verschmelzen, um die Schwächen des einen durch die Stärken des anderen ausgleichen zu können. Metasysteme, die verschiedene Funktionalitäten und Features unterschiedlicher Formen künstlicher Intelligenz sammeln können, würden eine heute nicht einschätzbare Performance und Leistungsfähigkeit in sich bergen.

---

[58] HARIK, GEORGES 1999 S.17

LITERATURVERZEICHNIS

BÄCK, THOMAS (1996), Evolutionary algorithms in theory and practice, Oxford Univ. Verlag, New York, 1996

DARWIN, CHARLES (1975), On the origin of species, Harvard Univ. Press, Cambridge, 1975

DAWID, HERBERT (1996), Adaptive Learning by genetic algorithms, Springer Verlag, Berlin, 1996

DUDEN (1989), Dudenverlag, Mannheim, 1989

FOGEL, LAWRENCE J. (1999), Intelligence through simulated evolution: forty years of Evolutionary Programming, John Wiley&sons, USA, 1999

GOLDBERG, DAVID (1999), Genetic algorithms in search, optimization and machine learning, Addison Wesley Longman ,Canada 1989, S. 186-184

HARIK, GEORGES/LOBO, FERNANDO/GOLDBERG, DAVID (1997), The Compact Genetic Algorithm, IlliGAL Report No. 97006, Illinois, 1997

HARIK, GEORGES (1999), Linkage Learning via Probabilistic Modeling in the ECGA, IlliGAL Report No. 99010, Illinois, 1999

HOLLAND, JOHN (1992), Adaption in natural and artificial systems, MIT-Press, Cambridge, 1992

HTTP://CS.FELK.CVUT.CZ/~XOBITKO/GA (1999)

KINNEBROCK, WERNER (1994), Optimierung mit genetischen und selektiven Algorithmen, Oldenbourg Verlag, München, 1994

NISSEN, VOLKER (1994), Evolutionäre Algorithmen, Dt. Univ. Verlag, Wiesbaden, 1994

NISSEN, VOLKER (1997), Einführung in evolutionäre Algorithmen, Vieweg Verlag, Braunschweig, 1997

N.N. (1999), Computer experiments with the ECGA, IlliGAL internal Report, Illinois, 1999

RECHENBERG, INGO (1973), Evolutionsstrategie, Frommann Verlag, Stuttgart, 1973

Ehrenwörtliche Erklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.
Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nichtveröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht.
Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Bayreuth, den 30.09.1999 .....
Andreas Feulner